Ingenieurfakultät Bau Geo Umwelt
Methodik der Fernerkundung

Joint Information Augmentation of Road Maps, Aerial Images and Ground Images

Gellért Máttyus, M.Sc.

Vollständiger Abdruck
der von der Ingenieurfakultät Bau Geo Umwelt
der Technischen Universität München
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigten Dissertation.

Die Dissertation wurde am 9. Juni 2016 bei der Technischen Universität München eingereicht und durch die Ingenieurfakultät Bau Geo Umwelt am 18. Juli 2016 angenommen.

Abstract

Extracting information about roads is important for many applications, such as infrastructure monitoring, traffic management, urban planning, vehicle navigation, realistic driving simulations, and it will be essential in the future for autonomous driving cars. The most straightforward way to express the road information is through a detailed map. Collecting road information on the spot (the ground) for a larger area is labor and time intensive as the surveyor has to visit the whole area of interest. Aerial images provide a rich information source to survey and map a larger area remotely, but if the images are interpreted manually, this process typically needs long, tedious work.

Analyzing the aerial images automatically can make the analysis of remote sensing images much more efficient. In the ideal case a complete map could be created from an aerial image without any human intervention. However, this is a very difficult task, for many scenes the aerial view does not provide enough information and even humans can only hardly interpret the image. The majority of available maps were created by also incorporating local surveys or other information sources to resolve uncertainties present in the remote sensing images. This makes the maps a great tool to include ground measurements and prior knowledge into the analysis of aerial images. *The goal of this thesis is to apply the already existing road maps jointly with aerial and ground images in fully automatic workflows. The information missing in the map is augmented with information present in the aerial and ground images and vice versa.* Three problems are investigated for the information exchange between aerial imagery and road maps and one problem where ground images are also included. This is a *cummulative* dissertation, the four problems are addressed by four peer-reviewed papers:

Fast multiclass vehicle detection on aerial images: Conventional maps describe the static characteristic of the roads, however, the dynamic traffic conditions on the roads are an important input for navigation, as well as planning and managing the infrastructure. A single aerial image with appropriate resolution enables to count the vehicles and extract their location and orientation. This already allows to estimate the utilization of the roads and parking spaces. By using multiple image frames, the speed of the vehicles can also be measured to estimate the traffic flow over the area covered by the aerial imagery. The key problem to solve is the reliable and quick detection of the vehicles in the images. Toward this goal, a fast and high-performing vehicle detector is proposed for aerial images. In contrast to previous methods, this estimates also the direction and the category class of the vehicle. The method does not need an orthorectified image, it can work on an original image frame without a prior of the region of interest, e.g. a road mask. The performance of the method is examined on a new dataset containing several thousand vehicles. The fast speed of this detector makes it suitable for real-time airborne road traffic extraction.

Large scale aerial image sequence geolocalization with road traffic as invariant feature: Aerial images are the most practicable if their geolocation is known and a pixel coordinate in the image can be transformed to a world coordinate. A novel approach is proposed to extract the geolocation of aerial images by using only a road map and the traffic visible in the images. In contrast to the three other tasks, here the aerial image is augmented with extra information from the map, i.e. the geolocation. The road network pattern over a larger area tends to be so unique that it can be used to recognize the geolocation. It is not even needed to extract the complete road network in the images, already a fraction is enough. Instead of detecting the roads directly, the traffic in the scene is extracted in form of tracks of moving vehicles. Using these tracks the images can be successfully geolocalized in a search area of 22500 km^2 containing 32000 km of streets in the Munich metropolitan area. This method could replace the expensive and heavy Global Positioning System (GPS) + Inertial Measurement Unit (IMU) systems used for creating geolocalized aerial images. Such systems will be particularly important for Unmanned Aerial Vehicles (UAVs), where the weight and cost of the system is more critical. This method localizes the images similar as humans localize themselves, based only on visual information and a road map.

Enhancing road maps with the street width by parsing aerial images: Currently used road maps are intended for the navigation of humans. The roads as stored as centerlines with connections to other roads without providing detailed information about the physical dimensions of the road. Having access to the width attributes of the roads would be important for infrastructure planning, creating realistic simulations over roads and they can improve the automatic scene understanding of autonomous vehicles. To address these demands, a method is presented to automatically extract the road width while also considering misalignments between the road network and the aerial image. Instead of formulating the problem as a pixelwise semantic segmentation, the problem is defined as one of inference in a Markov Random Field (MRF) reasoning about the road parameters directly. This makes it very fast, more robust and the topology of the road network is preserved. Experiments are conducted on three datasets, over Bavaria, Germany, over the city of Karlsruhe, Germany and Google Earth images over various locations around the world. The proposed method outperforms the state of the art in both speed and accuracy. The ability of the detailed maps to improve the scene understanding of ground images is demonstrated on the KITTI autonomous driving dataset.

Fine-grained road segmentation by parsing ground and aerial images: The estimation of the road width is extended to extract the fine-grained road layout by estimating the number and width of lanes plus the presence and width of parking spots and sidewalks. Importantly, the proposed approach applies existing road maps, aerial images and ground images jointly. The problem is formulated as one of inference in an MRF reasoning about the road layout as well as the alignment between the aerial image, the map and the ground image sequence in a joint energy function. The MRF takes features extracted from the images by deep learning as data terms and formulates the constraints on the lane sizes and the road layout as pairwise potentials. This allows robust estimation also in case when the image evidence (e.g. lane markings) is not visible or is missing. The alignment of ground and aerial images is necessary as even when applying sophisticated GPS-IMU systems, registration errors can still occur. The registration of the ground images to the map can also function as precise self-localization of the vehicle within the road, an important task for path planning and safe driving. Experiments are performed on a dataset including annotated aerial and ground images over the same area.

Zusammenfassung

Die Erfassung von Straßeninformationen ist wichtig für eine Vielzahl von Anwendungen, wie Infrastrukturüberwachung, Verkehrsmanagement, Stadtplanung, Fahrzeugnavigation, realistische Fahrtsimulationen und in Zukunft auch für autonomes Fahren. Die gängigste Methode um Straßeninformationen darzustellen sind detaillierte Straßenkarten. Die Erfassung von detaillierten Straßeninformationen vor Ort (an der Straße) erfordert viel Aufwand, da Vermessungstrupps das ganze Gebiet besuchen müssen. Luftbilder sind eine reiche Informationsquelle um größere Gebiete zu messen und aus der Ferne zu kartieren, auch wenn für die manuelle Bearbeitung und Interpretation der Bilder trotzdem noch viel Aufwand benötigt wird. Die automatische Verarbeitung von Luftbildern könnte die Analyse von Luftbildern viel effizienter machen. Im Idealfall würde man sogar eine vollständige Karte ohne manuelle Arbeitsschritte erstellen können. Aber die vollautomatische Bildverarbeitung ist eine sehr anspruchsvolle Aufgabe, weil auch die Luftbilder in vielen Szenen nicht genügend Informationen anbieten, sodass auch Menschen einige Bilder nur schwierig interpretieren und kartieren können. Der Großteil der Straßenkarten, die heute zu Verfügung stehen, wird durch Bodenmessungen und mit anderen Zusatzinformationen erstellt, welche auch die Unsicherheiten in den Luftbildern lösen können. Dadurch sind Karten mit den integrierten Bodenmessungen und zusätzlichem a-priori Wissen sehr geeignet, um die automatische Analyse von Luftbildern zu stützen. Ziel dieser Dissertation ist die automatische Verknüpfung von existierenden Straßenkarten mit Luftbildern und mit terrestrischen Aufnahmen. Fehlende Informationen in den Karten werden durch Informationen in den Bildern ergänzt, fehlende Luftbildinformationen wiederum werden durch Informationen in den Karten ergänzt. Insgesamt drei Probleme über die Verarbeitung von Straßenkarten und Luftbildern werden untersucht, wobei bei einem Problem auch zusätzlich terrestrische Aufnahmen hinzukommen. Diese Dissertation ist kumulativ geschrieben; die vier nachfolgend beschriebenen Probleme sind durch vier Peer-Reviewed Veröffentlichungen abgedeckt:

Schnelle Multiklassen-Fahrzeugdetektion in Luftbildern: Die konventionellen Straßenkarten beschreiben meist nur die statischen Eigenschaften von Straßen, obwohl dynamische Verkehrsinformationen sehr wichtig für die Fahrzeugnavigation als auch für Infrastrukturplanung und -verwaltung sind. Mit Hilfe eines einzelnen Luftbilds ist es möglich, die Straßen- und Parkplatzbelegung durch Zählung von Fahrzeugen mit ihren Positionen und Richtungen zu schätzen. Mit einer Bildsequenz kann man sogar die Geschwindigkeit von Fahrzeugen messen, um daraus die Verkehrslage zu bestimmen. Um diese Grundaufgabe optimal zu lösen, wird ein Fahrzeugdetektor präsentiert, der eine verbesserte Fahrzeugdetektion mit kurzen Rechenzeiten liefert. Die vorgeschlagene Methode bestimmt im Gegensatz zu anderen Methoden sowohl die jeweilige Fahrzeugklasse als auch die Orientierung der Fahrzeuge. Dabei wird nur ein einzelnes georeferenziertes, nicht orthorektifiziertes Luftbild ohne Einschränkung des Suchgebiets durch eine Straßenmaske benötigt. Die Leistung von der Methode ist in einen aktuellen Datensatz mit mehreren tausend Fahrzeugen untersucht worden. Die schnelle Geschwindigkeit dieser Methode erlaubt die Anwendung für Echtzeit Verkehrserfassung aus Luftbildern.

Großflächige Luftbildsequenz Georeferenzierung durch Verwendung von Straßenkarten als invariante Merkmale: Luftbilder sind dann am besten geeignet für Anwendungen, wenn eine Transformation der Bildkoordinaten zu Weltkoordinaten möglich ist. Ein neuer Ansatz wird präsentiert, der die Georeferenzierung von Luftbildsequenzen mit Hilfe einer Straßenkarte und mit Hilfe des in den Bildern sichtbaren Verkehrsstroms bestimmt. Im Gegensatz zu den drei anderen Problemen wird hier die Straßenkarte verwendet, um fehlende Informationen bei den Luftbildern zu ergänzen. Die Muster der Straßenkarte erscheinen genügend eindeutig zu sein, so dass man sie zur Georeferenzierung verwenden kann. Dabei muss nicht das komplette Straßennetz im Bild extrahiert werden, sondern es genügt ein Teil davon. Statt direkt die Straßen zu extrahieren, werden sich bewegende Fahrzeuge detektiert und verfolgt. Anhand der Trajektorien dieser Fahrzeuge können die Luftbildsequenzen in einem großen Suchgebiet, z.B. um die Großstadt München mit 22500 km^2 Fläche und 32000 Straßenkilometer, erfolgreich lokalisiert werden. Diese Methode kann die Messungen mit kostenintensiven GPS Inertialsysteme ersetzen, um Luftbilder zu georeferenzieren oder um mit Hilfe von Bildaufnahmen in der Luft zu navigieren. Das wird insbesondere bei unbemannten Luftfahrzeugen wichtig, bei denen Gewicht und Preis eine wichtige Rolle spielt. Diese Methode erlaubt die Positionierung basierend nur auf visueller Information und Straßenkarten und ähnelt so der Orientierung von Menschen.

Erweiterung von Straßenkarten mit Straßenbreiteninformation durch Luftbildanalyse: Straßenkarten sind im Prinzip für die Orientierung von Menschen entworfen. Die Abbildung von Straßen erfolgt als Mittellinie mit Kreuzungen zu anderen Straßen. Detaillierte Informationen über die physikalischen Größen der Straßen werden meist nicht angegeben, obwohl diese für verschiedene Anwendungen, wie Infrastrukturplanung, realistische Fahrtsimulationen und für die Interpretation in autonom fahrenden Fahrzeugen wichtig sein könnten. Eine Methode wird präsentiert, welche die Straßenbreite automatisch aus Luftbildern extrahiert und die Registrierungsfehler zwischen Luftbildern und Straßenkarten berücksichtigt. Statt wie üblich das Problem als semantische Segmentierung zu betrachten, wird das Problem als Inferenz in einem Markov Random Field (MRF) beschrieben, das direkt die gewünschten Straßenparameter schätzt. Dies führt zu sehr schnellen Laufzeiten, robusteren Ergebnisse und zur Bewahrung der Straßentopologie. Experimente wurden auf drei Datensätzen durchgeführt: Luftbilder über Bayern, Bilder aus Google Earth über Karlsruhe, und weltweite Bilder aus Google Earth. Diese Methode liefert bessere Ergebnisse als vergleichbare Referenzmethoden und hat deutlich geringere Laufzeiten. Anhand des KITTI Datensatzes wird demonstriert, dass die so verbesserten Straßenkarten die Bildverarbeitung für autonomes Fahren verbessert.

Hochauflösende Straßensegmentierung durch Luftbilder und terrestrischen Aufnahmen: Das Verfahren zur Erfassung der Straßenbreiten wurde zu einer hochauflösenden Straßensegmentierung durch die Extraktion von Anzahl und Breite die Fahrspuren, Gehsteigen und Parkspuren erweitert. Entscheidend bei diesem Verfahren ist, dass Straßenkarten, Luftbilder und terrestrische Aufnahmen in einem gemeinsamen Arbeitsschritt verarbeitet werden. Das Problem ist als Inferenz in einem MRF formuliert, welches die Straßenanordnung sowie die Registrierung zwischen Luftbildern, terrestrischen Aufnahmen und der Straßenkarte gemeinsam schätzt. Deep Learning wird angewendet, um semantische Bildmerkmale zu berechnen, und als Datenterm in dem MRF zu verwenden. Die Zwangsbedingungen der Breite der Spuren und die Struktur des Straßenlayouts werden durch paarweise Potentiale ausgedrückt. Das ermöglicht eine robuste Schätzung, auch wenn wenig Bildinformation vorhanden ist, z.B. wenn die Straßenmarkierungen fehlen oder nicht sichtbar sind. Die Registrierung zwischen die Luftbildern und terrestrischen Aufnahmen ist notwendig, da auch bei Verwendung von sehr präzisen GPS Inertialsystemen die geforderte Genauigkeit nicht immer eingehalten werden kann. Diese Registrierung kann man auch als präzise Lokalisierung betrachten, was für Routenplanung und autonomes Fahren sehr wichtig ist. Experimente wurden auf einem Datensatz ausgeführt, der annotierte Luftbilder und terrestrische Aufnahmen vom selben Gebiet beinhaltet.

Contents

Abstract i

Zusammenfassung iii

1 Introduction 1
1.1 Objectives 4
1.1.1 General goal 4
1.1.2 Methodological goals 4
1.2 Reader's Guide 5

2 Basics and state of the art 8
2.1 Image analysis fundamentals 8
2.1.1 Supervised machine learning 9
2.1.1.1 Support Vector Machines (SVMs) 11
2.1.1.2 Decision Trees 12
2.1.1.3 Boosting 13
2.1.1.4 Neural Networks 14
2.1.2 Deep Learning 14
2.1.2.1 Training the network by Backpropagation 16
2.1.2.2 Convolutional Neural Networks (CNNs) 16
2.1.3 Graphical Models 20
2.1.3.1 Markov Random Fields (MRFs) 20
2.1.3.2 Inference 22
2.1.3.3 Structured Support Vector Machines (SSVMs) 25
2.2 Image geolocalization 28
2.2.1 Map projections 28
2.2.1.1 Geographic Datum 28
2.2.1.2 Projections 29
2.2.2 Aerial image orthorectification 29
2.2.3 Image based localization 30
2.2.3.1 Image to image matching 30
2.2.3.2 Cross-view image matching 31
2.2.3.3 Image to map matching 31
2.3 Detecting objects in images 32
2.4 Semantic image segmentation 33

3 Fast multiclass vehicle detection on aerial images 36
3.1 Related Work 37
3.2 Multi-direction Vehicle Detection 38
3.2.1 Binary sliding window detector 38
3.2.1.1 Fast image features 39
3.2.1.2 AdaBoost classifier in Soft Cascade structure 39
3.2.1.3 Multi-direction detection 40
3.2.1.4 Single classifier method 40
3.2.1.5 Aggregated classifier method 40
3.3 Multiclass Vehicle Classification 40
3.3.1 Orientation estimation 41
3.3.2 Type classification 41
3.4 Experiments 42
3.4.1 Results on Munich images 43
3.4.1.1 Multi-direction vehicle detection 43
3.4.1.2 Multi-class vehicle classification 43
3.4.1.3 Baseline comparison 44

3.4.1.4 Computation time 45
3.4.2 Baseline comparison on UAV images 45

4 Large scale aerial image sequence geolocalization with road traffic as invariant feature 46
4.1 Related Work 47
4.2 Method 48
4.2.1 Track extraction 49
4.2.2 Matching the tracks to the road network 49
4.2.2.1 Geometric hashing 49
4.2.2.2 Verification of the shortlist 51
4.3 Experiments 52
4.3.1 Quantitative evaluation 53
4.3.2 Comparison to simple chamfer matching 54

5 Enhancing road maps with street width by parsing aerial images 56
5.1 Related Work 58
5.2 Augmenting Road Maps from Aerial Images 59
5.2.1 Energy Formulation 59
5.2.1.1 Road classifier: 61
5.2.1.2 Edge: 61
5.2.1.3 Object detector: 61
5.2.1.4 Homogeneity: 61
5.2.1.5 Context features: 62
5.2.1.6 Smoothness: 62
5.2.1.7 Overlap: 62
5.2.2 Inference 63
5.2.3 Learning 63
5.3 Experimental Evaluation 65
5.3.1 Datasets 65
5.3.1.1 Bavaria: 65
5.3.1.2 Aerial KITTI: 65
5.3.1.3 World: 65
5.3.2 Metrics: 65
5.3.2.1 Comparison to baselines: 66
5.3.2.2 Importance of the features: 66
5.3.2.3 Segmenting the world: 66
5.3.2.4 Domain Adaptation: 66
5.3.2.5 Processing time: 66
5.3.2.6 Ground-level Scene Understanding: 67
5.3.2.7 Failure modes and limitations: 67

6 Fine-grained road segmentation by parsing ground and aerial images 69
6.1 Related Work 69
6.1.1 Aerial image parsing: 69
6.1.2 Aerial parsing with maps: 69
6.1.3 Fine-grained road parsing: 70
6.1.4 Aerial-to-ground reasoning: 70
6.2 Fine-grained Semantic Parsing of Roads 70
6.2.1 Model Formulation 71
6.2.1.1 Aerial semantics: 72
6.2.1.2 Aerial edges: 73
6.2.1.3 Along the road smoothness: 73
6.2.1.4 Parallel roads: 73
6.2.1.5 Road collapse constraints: 73
6.2.1.6 Lane size constraint: 74
6.2.1.7 Centerline prior: 75

6.2.1.8 Ground semantics: 75
6.2.1.9 Ground alignment smoothness: 75
6.2.2 Inference via Block Coordinate Descent (BCD) 75
6.2.3 Training with SSVM 76
6.3 Experimental Evaluation 76
6.3.1 Comparison to the state-of-the-art: 79
6.3.2 Deep semantic features in aerial Images: 79
6.3.3 Alignment between aerial and ground images: 79
6.3.4 Qualitative Results: 79
6.3.5 Ablation studies: 79
6.3.6 Inference time: 79
6.3.7 Limitations: 80

7 Conclusion and Outlook 81
7.1 Summary 81
7.2 Improvements and future work 82

List of Abbreviations 84

List of Figures 85

References 91

Acknowledgments 97

Appendices 98

A Liu K., Mattyus, G., 2015. Fast Multiclass Vehicle Detection on Aerial Images. IEEE Geoscience and Remote Sensing Letters 12(9): 1938-1942 99

B Mattyus G., Fraundorfer, F., 2016. Aerial image sequence geolocalization with road traffic as invariant feature. Image and Vision Computing, Volume 52: 218–229. 105

C Máttyus, G., Wang, S., Fidler S. and Urtasun R., 2015. Enhancing Road Maps by Parsing Aerial Images Around the World. 2015 IEEE International Conference on Computer Vision (ICCV), pages 1689–1697. 122

D Máttyus, G., Wang, S., Fidler S. and Urtasun R., 2016. HD Maps: Fine-grained Road Segmentation by Parsing Ground and Aerial Images, Conference on Computer Vision and Pattern Recognition (CVPR) 2016. 132

1 Introduction

Maps are used to describe the spatial relations of the objects in the world. Without maps the modern life is hardly imaginable, they are used for designing and maintaining the infrastructure, planning constructions and investments, managing agriculture and the environment, etc. Particularly important are road maps helping people to navigate from one point to another. Mankind has been using maps since the ancient times and as the technology advanced, maps were enhanced and extended to meet new demands and cover new areas. This evolution is still ongoing. Especially important was the emergence of commercially available GPS navigation devices enabling the automatic localization of the vehicles. When connected to a digital road map, these GPS devices can navigate a vehicle on entire continents, but only to the extent of the reliability of the applied map. This poses a high demand on more accurate, more detailed and more complete road maps. This aspect will get more important for fully autonomously driving cars, which seem to be realized in the future. Autonomous vehicles have to understand their environment, interact with other traffic participants and be able to self-localize precisely under any circumstances. Highly detailed road maps can support these by providing priors about the environment, intentions of other vehicles or pedestrians, and most importantly, maps can enable path planning and self-localization similarly as humans were using maps to find their location before the time of GPS. Navigation and traffic are very important, but not the only applications requiring fine-grained maps. Detailed maps can be leveraged in virtual and augmented reality for both for the management and maintenance of the man-made and natural environment or for entertainment purposes. It seems inevitable to create more detailed and more complete maps for the demands of current and future technologies.

Existing maps were created by huge manual effort and they represent a type of human knowledge. The creation of high definition maps can be based on the map information already present plus additional sensors. Images are a rich information source easily understandable by humans, which makes them convenient for mapping and surveying. Aerial images provide full coverage over a large area, without the need of collecting data on the spot. This makes them an important source of geospatial information and maps since the dawn of aviation and photography. Airplanes are the most widespread way to carry cameras, however nowadays UAVs are becoming a more economical alternative. Remote sensing imagery is particularly important for public administrations (e.g. cadastre), for the military and the intelligence. Nowadays aerial images are abundant, almost all major cities have aerial imagery, often also publicly available (e.g. German cities) or even for free (e.g. the United States Geological Survey (USGS) in the USA). Manual mapping and surveying using remote sensing images is a laborious and often tedious work resulting in high costs. Analyzing the aerial images automatically could make remote sensing much more efficient. In the ideal case a complete map could be created from an aerial image without any human intervention. However, this is a very difficult task. There are partially automatic systems for certain tasks, but a general, fully automatic mapping system is currently not realized (at least to the knowledge of the author). For many scenes the aerial view does not provide enough information and even humans can only hardly interpret the scene. Existing maps and images can be applied jointly to augment new information present in one of the data but not in the other by incorporating the information of both. The collection of maps typically employed also ground surveys or other prior knowledge supporting the image interpretation when remote sensing data alone was not informative enough. This information can be leveraged when existing maps and aerial image are processed jointly.

For managing the traffic, planning an optimal route for the current road utilization and estimating the time of arrival, live information about the road traffic is required. Various methods exist to acquire traffic data on the ground. Special sensors can be deployed (e.g.

by induction loops, surveillance cameras) or a fleet of mobile positioning devices can be tracked. Such a fleet can be as large as all the users of a given mobile operating system platform. Alternatively, aerial images can be used to count, position and track the vehicles visible in the image. This is more expensive, since flying an aircraft involves higher costs. However, aerial images can cover a large area rapidly, the area of interest can be set on demand and such a system is independent from the terrestrial infrastructure. A system estimating the traffic from aerial images can provide an extensive picture of the state of the infrastructure and traffic even in case of disasters when electricity outages and interruptions in the communication occur.

The most dominant feature connecting images and maps is location. When an image is precisely georeferenced and the coordinate transformation between the image and the map is known, the image location containing an object in the map can be directly visualized. This also applies vice versa, the content of an image can be queried (interpreted) using the map, if it is detailed enough to contain the objects of interest. A good example is *Google Maps* enabling to switch between the image and the map view. Aerial images particularly need geographic information, otherwise the content of the image is far less informative. It would be ideal to have 100% correctly georeferenced images and accurate maps describing the world to every detail. However, in general this is not feasible, as the registration between the map and the image is not perfect neither the map is detailed enough and up-to-date. Both the map and the geolocation of the image can contain errors, which corrupt the registration between them. Furthermore, maps describe a certain moment of the physical world to a given extent. Changes occur, e.g. cars and people move, new roads and buildings are built, and the maps become outdated only partially describing the reality. The joint inference of inaccurately geolocalized images and imperfect maps can improve registration, extend the amount and detail of covered objects and update the content to the latest state of the physical world. Even if the location of an image is entirely missing, matching the features of the map and the image can recover the location, hence the position of the camera. Maps containing partial data can already assist the extraction of more categories, e.g. a road map containing only the road centerlines can support the extraction of the detailed road layout or help building detection, since most buildings are connected to roads. Performing this joint inference automatically could allow the frequent upgrade of maps in a cost efficient way.

Maps are created by both authorities, private companies and recently also by volunteers in the form of the OpenStreetMap (OSM) [1] project. OSM could be described as the ”Wikipedia” of maps. Volunteers edit the geospatial maps by doing infield local measurements using handheld GPS devices or by using aerial or satellite images provided by certain companies (e.g. Bing Maps) or local authorities (e.g. the City of Stuttgart) [2] . OSM maps are freely available, the coverage is very high and it is growing rapidly. Current road maps were intended to support the navigation of human drivers. They provide a topological network enabling routing across the road network. The road is depicted as nodes connected by directed edges (lines), while an intersection is described by a node with multiple edges. Basic maps define only the centerline of the road and the turning lanes at intersections. This is enough for human drivers capable of precisely localizing their vehicle on the road based on visual information. More information (e.g. the width of the road, number of lanes) would be probably more disturbing than useful. In contrast, the artificial intelligence of an autonomous car can be supported by much more information about the road. An accurate, more detailed map can be highly useful to enable safer driving. A human can also drive better on a known road which she/he has ”mapped” before and has a prior knowledge about. A detailed map can be considered such a prior. If the system knows that there is a sidewalk it can increase the likelihood of a pedestrian stepping down on the road, if a parking lane is present, a car

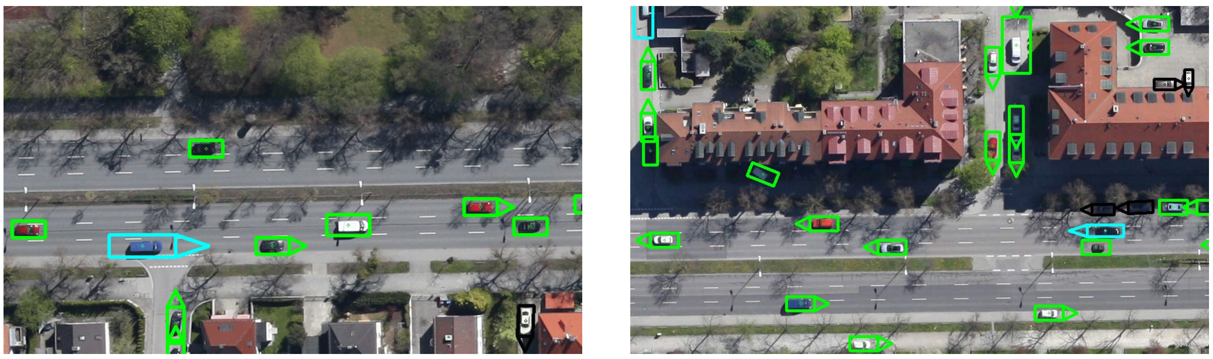
[1] `http://www.openstreetmap.org`
[2] Aerial and satellite images can only be used to edit OSM if the owner of the images allows this.

might turn out to the road.

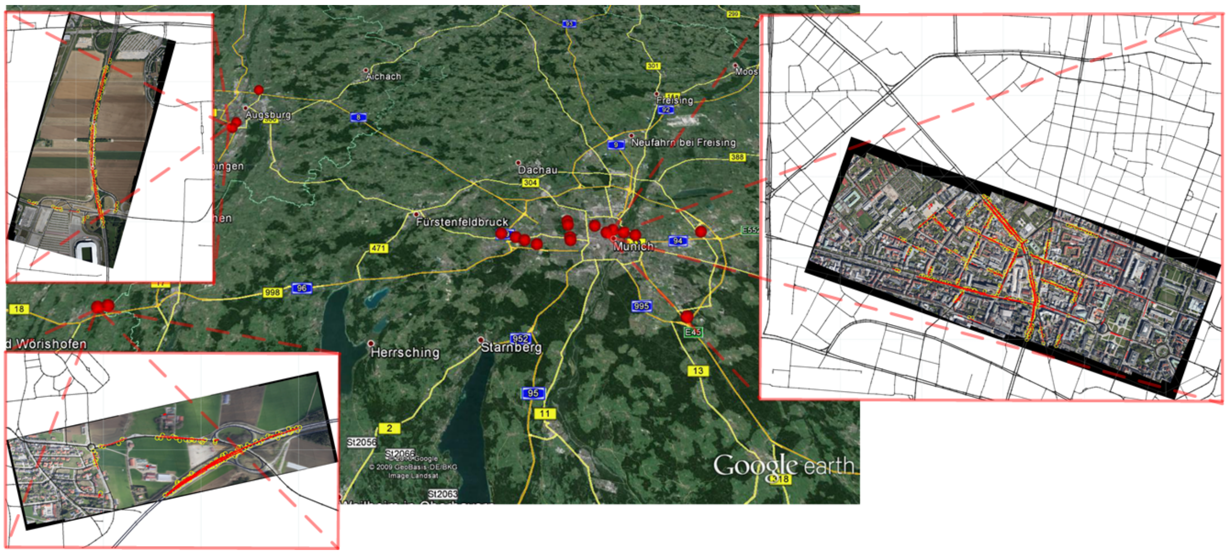
Detailed and reliable maps are needed also for the management and planning of the infrastructure. Local authorities typically have databases about the road infrastructure. However, these databases can origin from very different sources, from legacy measurements stored on paper to modern Light Detection And Ranging (LIDAR) Point Clouds (PCLs) acquired by survey vehicles. Therefore the completeness, the reliability and the grade of the details of these information can vary highly. An automatic system verifying the existing data based on aerial images can support the decision if new surveying is needed or the current data already fulfills the requirements. In certain cases an update using available imagery and maps could be enough instead of a new survey.

Modern cars are getting equipped with more and more sensors (e.g. cameras) for Advanced Driver Assistance Systems (ADAS) and ultimately fully autonomous driving. This makes the car fleet of the future a huge mobile sensor network driving the roads continuously. To extract coherent information the sensors of the individual cars have to be registered precisely. This task involves the alignment of possibly millions of sensors. These sensors can not be calibrated in a laboratory and their field of view will not cover the same area at the same time. Standard, image based image registration techniques, assuming overlap between the images, are hardly applicable. The appearance of a scene can change considerably between the visit of two vehicles, e.g. day vs. night, sunny weather vs. rainy, summer season vs. winter. Instead of registering the images to each other, the sensors can be registered to a common map. Maps contain higher level information: the spatial description of objects. These features are invariant to appearance changes, they do not depend on the lighting conditions or the applied sensor. Even different sensor modalities (e.g. image vs. radar) can be matched based on the spatial extent of the object they are capturing. By registering the sensors of a moving car to a precise map self-localization and sensor (camera) calibration becomes possible. This enables the reliable localization of the vehicle even if there is no GPS signal available, and allows to deploy consumer vehicles with lower cost (and lower precision) sensors. However, this can only work if detailed and up-to-date maps are available on a large scale. Therefore the augmentation of maps with detailed information and keeping them up-to-date will become an important aspect of the traffic infrastructure of the future. Alternatively to applying sophisticated surveying vehicles with calibrated sensors (e.g. GPS+IMU, LIDAR), lower cost sensors (i.e. cameras) can be applied to enhance the road maps. In this case the precise localization of the images is a more relevant problem. A possible solution is to apply aerial images jointly with the images acquired by the vehicles on the ground. The absolute location accuracy of aerial images is typically high, which can be leveraged to register them with the terrestrial images. However, matching images of different perspective and resolution is very challenging, for which standard image matching techniques are not suitable. The spatial extent of objects described by maps can provide the link between the two image types (i.e. air and ground) allowing registration. Inferring detailed road map information and augmenting the alignment of aerial and ground images jointly can give a more optimal result than doing these tasks individually.

The transportation systems of the future demand the augmentation of road maps with dynamic traffic information, more detailed categories (e.g. precise lane information), and the registration of images and maps by geolocalizing the images. This can be achieved by applying images and existing maps jointly.



Objective 1: Vehicle detection



Objective 2: Aerial image localization based on road maps

Fig. 1. Illustration of the objectives 1 and 2. The detected vehicles are shown with colored bounding boxes, different classes with different color. Black shows false-negative (missing) cars. The bottom image shows aerial images localized over a larger area. The image is transformed into the road map. The red color in the image shows the extracted car tracks used to match to the map. The red dots in the Google Earth image shows the location of the image.

1.1 Objectives

1.1.1 General goal

This thesis aims to present new scientific methods to automatically *augment information of road maps, aerial imagery and ground images by inferring jointly in these different data sources.*

1.1.2 Methodological goals

The general goal is reached by four methodological goals: providing new methods to solve four specific problems.

Objective 1:

Development of a more reliable, faster vehicle detector for extracting road traffic from aerial images in near real-time.

Aerial images can provide not only static information about the infrastructure but also dynamic properties, e.g. road utilization, traffic flow. To extract these dynamic properties the key is to detect the vehicles in the image reliably and automatically. This is a challenging task as the aerial images to be processed are large (e.g. 5000×4000 pixel), needing more computation but the size of the targets is still small (30×20 pixel) making it hard to distinguish objects from background. The German Aerospace Center (DLR) has developed a system in the project VABENE[3] to extract the road traffic in real-time based on an optical camera system mounted on an airplane. For the fast and accurate real-time traffic extraction in such a system it is crucial to improve the method of detecting the vehicles. See an illustration of the task in Fig. 1.

Objective 2:

Geolocalize aerial images without a GPS and IMU.

Geolocalizing aerial images depends on sophisticated GPS and IMU systems. These systems are expensive and heavy (several kg), especially the IMU component measuring the orientation and acceleration. A solution avoiding the GPS+IMU could enable lower costs and smaller weight, which is especially important for small UAVs only able to carry a very limited payload (a few kg or even less). Aerial images contain rich information which can also enable to solve this problem. The road network pattern tends to be so unique on a large scale that it can be used for recognizing the location and thus be used for georeferencing. Fig. 1 shows an illustration of this problem.

Objective 3:

Enhance road maps with street width by parsing aerial images.

Maps simplify the representation of the road to its centerline. This is enough for human navigation, but the real dimensions of the road remain important for many tasks. Aerial images cover a huge area of the world. If the road width is extracted automatically from these images, a large part of the world can be covered with enhanced maps. Fig. 2 shows this problem.

Objective 4:

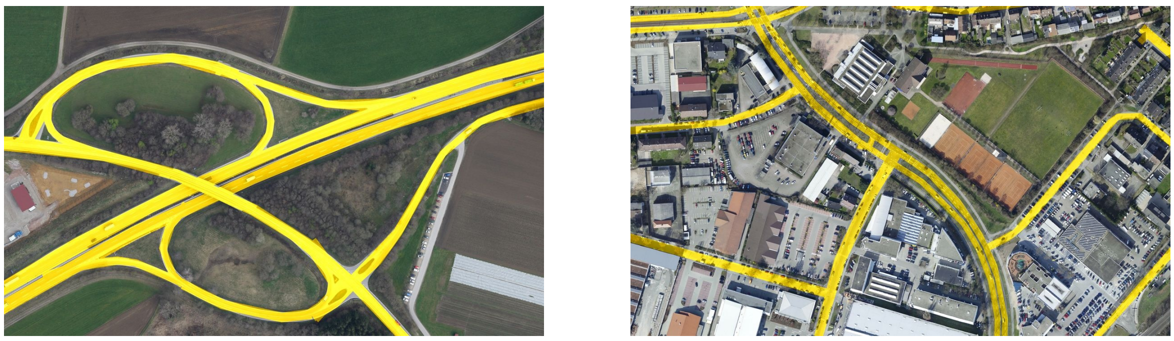
Fine-grained road segmentation by parsing ground and aerial images.

The road layout (number of lanes, presence of sidewalk, parking spots) is visible in high resolution aerial images but small details might only be visible in ground images. Extracting the road layout by using the aerial and ground perspective jointly can address the problems present if just one data source is used. Aerial images give a high coverage with good absolute geometry but they might miss some details, they need clear view over the road and the perspective makes it hard to detect thin vertical objects (e.g. barriers). Ground images are not effected by objects covering the sky (e.g. bridges, high building), and they show fine details not visible from the air (e.g. the curb of the road, a barrier) but getting a full coverage over a larger area needs exhaustive efforts and the view can be easily covered (e.g. by other vehicles). See an illustration of the task to solve in Fig. 2.

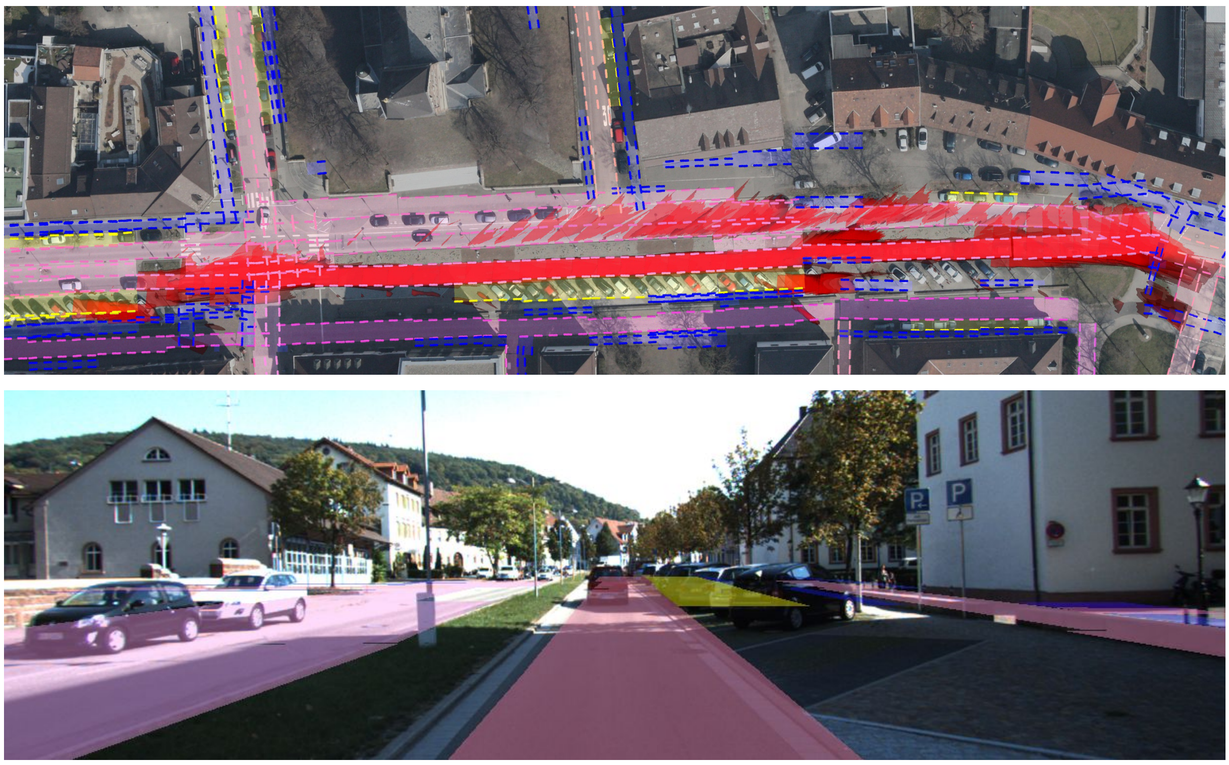
1.2 Reader's Guide

This is a *cummulative* dissertation, the four objectives are addressed by four peer-reviewed papers:

[3] `http://www.dlr.de/vabene/`



Objective 3: Enhance road maps with street width



Objective 4: Fine-grained road segmentation by parsing ground and aerial images

Fig. 2. Illustration of the objectives 3 and 4. The extracted road width is shown by yellow color. The two bottom image shows the road layout extracted jointly in the aerial and the ground (bottom) image. The coloring is pink: road, yellow: parking lane, blue: sidewalk. The detected road from the ground is projected into the aerial image and it is shown by red color.

◇ Kang Liu and Gellert Mattyus: *Fast Multiclass Vehicle Detection on Aerial Images*, IEEE Geoscience and Remote Sensing Letters (Liu and Mattyus, 2015).
◇ Gellert Mattyus and Friedrich Fraundorfer: *Aerial image sequence geolocalization with road traffic as invariant feature*, Image and Vision Computing (Máttyus and Fraundorfer, 2016).
◇ Gellert Mattyus, Shenlong Wang, Sanja Fidler and Raquel Urtasun: *Enhancing Road Maps by Parsing Aerial Images Around the World*, International Conference on Computer Vision 2015 (Mattyus et al., 2015)
◇ Gellert Mattyus, Shenlong Wang, Sanja Fidler and Raquel Urtasun: *HD Maps: Fine-grained Road Segmentation by Parsing Ground and Aerial Images*, Conference on Computer Vision and Pattern Recognition 2016. (Mattyus et al., 2016).

The papers are included in the appendix A, B, C, D.

The thesis is structured as following:

◇ *Chapter 2* gives a brief introduction of the basics of computer vision and machine learning.

In the second half of this chapter the task of image localization, object detection and semantic image segmentation is introduced and a short general state of the art is given. The more task specific related work is in the corresponding chapters.

- *Chapters 3-6* addresses the four objectives covered by the four papers.
- *Chapters 7* is the conclusion with an outlook on future work.

2 Basics and state of the art

This thesis is in the field of analyzing and interpreting aerial and ground images automatically. Therefore the fundamentals of automatic image analysis and interpretation is presented in this chapter by describing the basics of computer vision and machine learning related to the subject of this thesis. The second part of this chapter contains the state of the art and the problem formulation of three related fields: image based geolocalization, object detection in images and the semantic segmentation of images. This is a general overview, more task specific related works are at the beginning of each chapter.

2.1 Image analysis fundamentals

Optical images are a very rich information source about our environment and in general the world. Vision can be probably considered the most important sense for humans. The advent of digital images brought the demand to analyze these images automatically by a computer program. This was true for all application fields where digital images were available, e.g. medical imaging, manufacturing quality control, etc. and analyzing remote sensing images created by satellites or airborne sensors. As cheap digital cameras became available for the public and the amount of digital images was increasing rapidly, the automatic interpretation of the visual information turned also to analyzing everyday images, e.g. detecting human faces in family photos, understand the scene and and recognize the geolocation in a holiday photo. These images are taken by various sensors in uncontrolled environments resulting in a much higher variance for the content of the image, e.g. a car in an image can appear in multiple sizes, different lighting conditions and different views.

The problem of analyzing the content of an image in general turned out to be extremely challenging. Even the estimation of the difficulty of this problem was hard as humans excel in these tasks and the problem seemed to be simple or even trivial. A good illustration for the underestimation of the visual perception problem is the MIT *Summer Vision Project* proposal (Papert, 1966) from 1966 which planned to solve key vision problems like background-foreground segmentation and object description during one summer. After decades of research the automatic systems are getting only now close to solve these problems reliably by using huge amount of data and computing resources to train models with millions of parameters.

The excellent human vision performance easily leads to underestimation of the difficulty of the problem. On the other hand it can work as an oracle (a method defining an accurate ground truth) for the task enabling the creation of huge datasets by human annotators without any special training. Huge open datasets are crucial for making reproducible, quantitative observations about the performance of different methods what is a criteria for a scientific process. Famous datasets like *Pascal Visual Object Classes* (Everingham et al., 2010) for segmentation and object detection, *KITTI* (Geiger et al., 2012) for autonomous driving and *ImageNet* (Russakovsky et al., 2015) for image classification play an important role for advancements in the methods. The huge dataset of ImageNet (1000 classes, 1000 images samples per class) was particularly important to train huge, deep neural networks with millions of parameter outperforming other methods in many vision tasks.

Early image analysis approaches tried to find simple mathematical models solving the vision problem. These approaches resulted mostly in defining heuristics, constructing features expressing these heuristics and tuning parameters (mostly manually) until good results were achieved on relatively small datasets. The problem of these methods is that the heuristics are mostly very domain specific and the application to other types of data is not clear or needs laborious manual intervention.

Learning based methods can deliver much better results on high-level image analysis (e.g. image classification) and the adaption to new data is well defined and predominantly automatic. The visual appearance depends on many factors (e.g. lighting conditions, view point, background, etc.) and the state of these factors is usually unknown to the observer. This results in a very large variability in the appearance and makes the problem ill-posed where a priori knowledge is needed for the interpretation. Learning based methods can incorporate the a priori knowledge provided by the training data much better than methods based on heuristics. However, there are low level vision problems where the learned a priori knowledge plays a less important role and learning based methods are not widely applied. An example for this is 3D geometry described by the pinhole camera model and the epipolar geometry (Hartley and Zisserman, 2004). But even in problems which seem to be low level (e.g. edge detection), the learning based methods (Li et al., 2016; Dollár and Zitnick, 2015) can outperform the pure algorithmic methods (Canny, 1986) on datasets annotated by humans.

For a wide range of computer vision problems learning based methods are formulated as relating the input data to the output data via a function which can be tuned to approximate the underlying unknown and complicated function (this function can include many factors, e.g. the physical light properties, the appearance of the objects, the context around the object, etc.). This is the approach of machine learning, especially supervised machine learning. For an input data $\mathbf{x}$ with corresponding output data $\mathbf{y}$, a function transforming the input to the output $\mathbf{y} = f(\mathbf{x}, \mathbf{w})$ is created by setting the $\mathbf{w}$ parameters of the function. The phase when the trainable parameters $\mathbf{w}$ are set is called the *training*. Later the parameters $\mathbf{w}$ are kept fix and are used to generate an output $\mathbf{y} = f(\mathbf{x}|\mathbf{w}) = f(\mathbf{x})$. This is called the *prediction*.

The following subsections introduce machine learning and two important subfields, i.e. deep learning and graphical models.

2.1.1 Supervised machine learning

On Wikipedia machine learning is defined as: "Field of study that gives computers the ability to learn without being explicitly programmed". Machine learning uses many methods also existing in other disciplines like stochastic signal processing, estimation theory and it can be considered as a descendant of pattern recognition.

The research included in this thesis applies predominantly supervised learning where there is a set of input data available with the corresponding output. Analyzing and learning structures in the data if only the input and no reference output is available is called unsupervised learning. This is a very important and challenging task as there are huge datasets without output, but unsupervised learning is not a topic of this work. For a more general overview about machine learning the reader is referred to (Bishop, 2006).

In general the goal of supervised learning is to create a function:

$$\mathbf{y} = f(\mathbf{x}, \mathbf{w}) \tag{1}$$

which transforms the input $\mathbf{x}$ to the desired output $\mathbf{y}$ by setting the parameters $\mathbf{w}$. $\mathbf{y}$ and $\mathbf{x}$ can be either a scalar, a vector, a matrix or a higher dimensional data. In digital computers the higher dimensional data (e.g. matrix) is stored as vectors, therefore $\mathbf{x}$ and $\mathbf{y}$ are considered a vector or a scalar. The $\mathbf{w}$ vector represents the trainable parameters of the function $f()$.

Consider the problem of classifying (i.e. assigning a class for the content of the entire image) an image of size $rows \times cols$ with red, green, blue color channels. In this case the $\mathbf{x}$ input vector is the image pixel intensity tensor $rows \times cols \times 3$, while the output is a single scalar defining a class id.

The first step to create the function $f(\mathbf{x},\mathbf{w})$ is the design of the structure of the function. This crucial step defines the meaning of the trainable parameters $\mathbf{w}$, how the training can be solved, what kind of functions can be approximated, what is the computational demand, etc. The core of machine learning research is how more optimal function structures for learning can be created.

After the structure of the function $f(\mathbf{x},\mathbf{w})$ is fixed, the trainable parameters $\mathbf{w}$ have to be set. This is done by using data for which both the input and the ground truth output is known. This is called the training dataset and it is essential for supervised learning. In the ideal case the training data would be identical to the true distribution of the data, but in practice this is not achievable. The collection of the data needs considerable effort (e.g. manual annotation of images) and if the real world problem contains continuous variables infinite number of training samples would be needed. The training set will be only an approximation of the true data distribution. If it is *independent and identically distributed* (IID), then the true data distribution can be approximated by interpolation of the training set, and smoothness is a good prior. If the training set is not IID over the true data, then certain domains of the distribution have to be extrapolated, instead of interpolation, which gives a higher uncertainty.

The training set contains N training examples $\hat{\mathbf{y}}^n, \mathbf{x}^n, n = 1,\ldots,N$ where $\hat{\mathbf{y}}^n$ denotes the ground truth output for the input $\mathbf{x}^n$. The prediction of the trained function for the same input is $\mathbf{y}^n = f(\mathbf{x}^n,\mathbf{w})$.

A loss function (also called error or cost function)

$$\mathcal{L}(\hat{\mathbf{y}}^n, \mathbf{y}^n) = \begin{cases} 0 \text{ if } \hat{\mathbf{y}}^n = \mathbf{y}^n \\ > 0 \text{ otherwise} \end{cases} \tag{2}$$

has to be defined which measures how good the predicted output is. If the prediction and the ground truth output is equal the loss should be zero, otherwise it should be a positive number.

By minimizing the loss function w.r.t the function parameters $\mathbf{w}$ the function can be trained (learned).

$$\mathbf{w}^* = arg\min_{w} \sum_{n=1}^{N} \mathcal{L}(\hat{\mathbf{y}}^n, f(\mathbf{x}^n, \mathbf{w})) \tag{3}$$

By solving (3) the output of the training data can be well predicted, but the goal of a system applying machine learning is to predict on previously unseen data. The objective is to create a function from the training data, which predicts on unseen data correctly. This ability is referred as generalization and is one of the main question of any machine learning method. It represents the ability to learn relevant features (kind of insight) in the data. The ideal training method would need very few training data and could generalize to a very wide range of inputs.

In contrast, a method which works only on the training data is not useful. Consider the example of classifying an image. One could use the method of creating a hash based on the representation of the image (i.e. the pixel values encoded in a structure, e.g. row wise), calculate a Hamming distance to this hash and classify the image as the class of the nearest neighbor. This method could provide 100% accuracy on the training data, but it is easy to see that even the tiniest modification of the data (e.g. changing a few pixel values, translating or scaling the image,) would bring the method to fail. This would mean the generalization ability of the method is almost zero, it could not be used for real world tasks. This method would not learn the important features for the image classification problem, but instead use an irrelevant aspect of the image, its digital representation.

If the dimension of free trainable parameters $\mathbf{w}$ is N, then the function can be fitted on N points (samples) but it might not provide a good estimation outside these N samples. Either the number of training samples have to be increased or prior knowledge about the data needs to be incorporated, preventing overfitting on the training set.

The process to include prior knowledge to prevent overfitting is called regularization. It can take many forms, e.g. data augmentation, but a common way is to add an additional term to the target function to be minimized (3), a so called regularizer function. Usually this a $L1$ or $L2$ norm suppressing large weights in case of functions with weights, e.g. $\mathcal{R}(\mathbf{w}) = ||\mathbf{w}||^2$. This is also called weight decay, as it suppresses weights for which there is no evidence in the data. This would lead to a minimization problem in the form of (4) where λ is a parameter to set the importance of the regularizer term.

$$\mathbf{w}^* = \operatorname*{argmin}_{w} \sum_{n=1}^{N} \mathcal{L}(\hat{\mathbf{y}}^n, f(\mathbf{x}^n, \mathbf{w})) + \lambda \mathcal{R}(\mathbf{w}) \tag{4}$$

The majority of the supervised learning problems can be expressed as (4).

Supervised learning can be divided based on the output function. If the output is a continuous variable $\mathbf{y} \in \mathbb{R}^d$ where d is the number of dimensions of the output, it is called *regression*. A simple one-dimensional case is fitting a function by the least squares method.

If the output is a discrete variable, the problem is called a *classification* task. Consider the problem of recognizing a number in an image containing a single number. The high dimensional input data is transformed to $y \in [0, 1, \ldots, 9]$ giving which class the image belongs to. Classification problems are particularly important during image analysis, many tasks are formulated as classification, e.g. the detection of cars in an image is formulated as classifying the possible car positions as car or background, the semantic segmentation if formulated as classifying each pixel in the image.

In the next paragraphs a few important classifiers applied in the thesis are presented.

2.1.1.1 Support Vector Machines (SVMs)

SVM is a liner classifier constructing a hyperplane $\mathbf{w} \cdot \mathbf{x} - b = 0$ in the input data $\mathbf{x}$ separating two classes (in case of binary classification) (Cortes and Vapnik, 1995). If the data is linearly separable it maximizes the distance between the hyperplane and the closest data points of the classes (i.e. the margin). See Fig. 3 for an illustration. This margin functions as a regularization and achieves good generalization if the training data has similar distribution as the testing data.

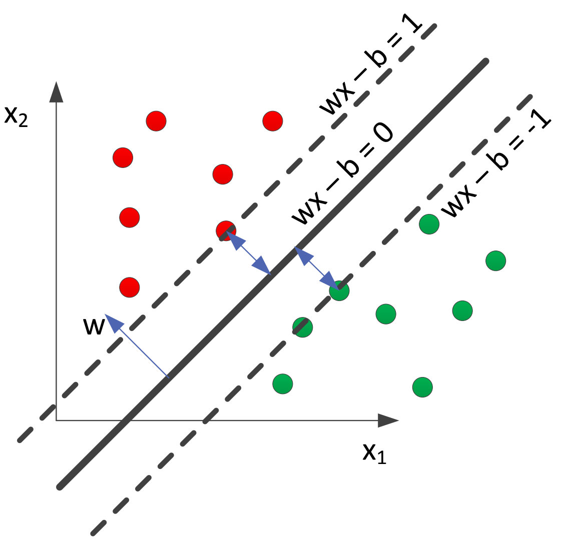
If the data is not linearly separable the misclassification error is considered in the form of the *hinge loss* $\mathcal{L}_h$

$$\mathcal{L}_h(\hat{y}, y) = \max(0, 1 - \hat{y} \cdot y) \tag{5}$$

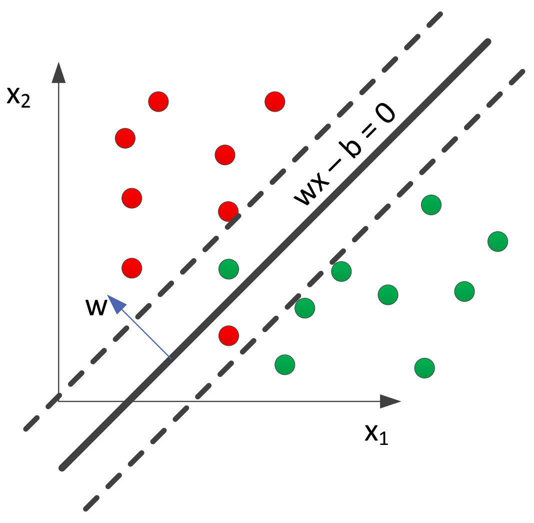
where y is the classifier score and $\hat{y} = \pm 1$ is the ground truth output.

The weights are trained by solving:

$$\mathbf{w}^* = arg \min_{\mathbf{w}} \frac{C}{N} \sum_{n=1}^{N} \mathcal{L}_h(\hat{y}^n, \mathbf{w} \cdot \mathbf{x}^n + b) + \frac{1}{2} ||\mathbf{w}||^2 \tag{6}$$



(a)



(b)

Fig. 3. An illustration of an SVM when the data is linearly separable (a). If the data is not linearly separable a trade-off is needed between the size of the margin and the mislabeling of the training data (b).

where C is a constant defining the weighting between the margin size and the classification error. This is a convex problem where the local and global minimum is identical. Finding the global optimum is guaranteed with any initialization.

By applying the kernel trick SVM can be extended to non-linear classification (Boser et al., 1992). The multiclass case can be handled by forming it as multiple binary classification problems (Duan and Keerthi, 2005).

2.1.1.2 Decision Trees

A decision tree predicts the output by making decisions (e.g. binary decisions) along a tree. The leaves of the tree gives the output which can either be a discrete or a continuous variable. The first case is a classification tree, the latter is a regression tree. Fig. 4 illustrates a fictional example for a decision tree predicting if a user of a website will click on an advertisement or not.

This gives an intuitive reasoning about a decision which can be easily understood by humans. The training chooses automatically from a pool of features. During testing only the features selected are calculated, not the entire pool. This enables the use of a huge feature pool for training without resulting in long feature computation time for prediction. In contrast, an SVM needs all the features used during training to make a prediction.

The optimal training of a decision tree (constructing the tree) is NP-complete (Hancock et al., 1996; Hyafil and Rivest, 1976). Therefore the training algorithms use heuristics, e.g. top-down greedy algorithms.

Decision trees tend to overfit, resulting in lower generalization capability (Bramer, 2007). This can be handled by using less decision nodes as the training set would imply, this is called pruning. Another way is to train an ensemble of decision trees on random subsets of the training data or the features and aggregate the predictions of these trees. This is called decision or random forests (Breiman, 2001).

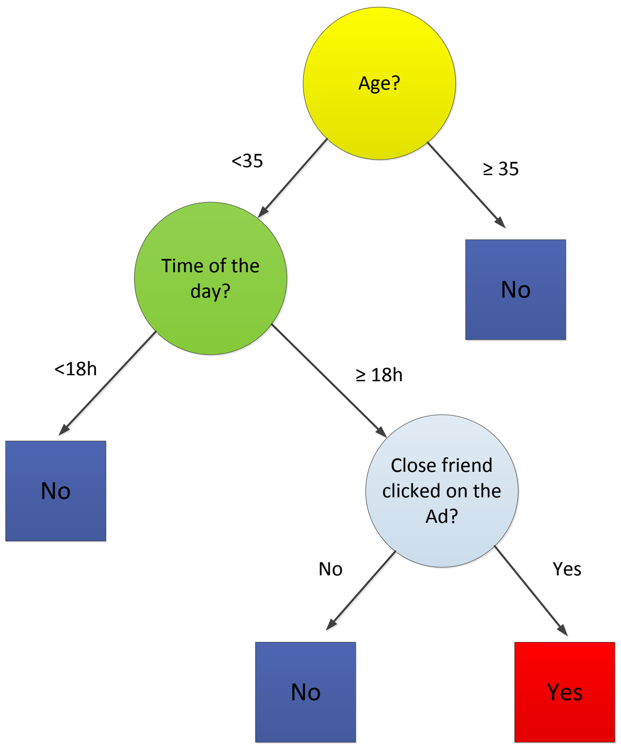


Fig. 4. An illustration of a binary classification tree predicting if the user of a website will click on an advertisement or not. The leaves (squares) are the class label output, the nodes (circles) are the binary decisions over an input feature.

2.1.1.3 Boosting

Boosting combines multiple weak (basic) classifiers $f_m(\mathbf{x})$ to form a strong classifier $F_M(\mathbf{x})$ performing much better than the individual classifiers (7). Weak classifiers are classifiers which only need to be slightly better than a decision by chance. The strong classifier is calculated from the weighted sum of these weak classifiers. A popular version is AdaBoost (Freund and Schapire, 1996). It is a sequential algorithm adding a new weak classifier $f_m(\mathbf{x})$ at each iteration. The new weak classifier is trained by putting more importance on samples misclassified by the previous classifiers. The final strong classifier is calculated as:

$$F_M(\mathbf{x}) = \operatorname{sign}\left(\sum_{m=1}^{M} \alpha_m \cdot f_m(\mathbf{x})\right) \tag{7}$$

As shown by (Friedman et al., 1998) this can be interpreted as a sequential minimization of an exponential loss function (8) with respect to the last weight α_m and the parameters of the last classifier $f_m(\mathbf{x})$.

$$\mathcal{L}_{\text{exp}} = \sum_{n=1}^{N} \exp(-\hat{y}^n \cdot F_M^*(\mathbf{x}^n)) \tag{8}$$

where

$$F_M^*(\mathbf{x}) = \frac{1}{2}\sum_{m=1}^{M} \alpha_m \cdot f_m(\mathbf{x}) \tag{9}$$

is the linear combination of the weak classifiers $f_m(\mathbf{x})$ and $\hat{y}^n = \pm 1$ is the ground truth output for $\mathbf{x}^n$.

The exponential loss function gives high values if $-\hat{y}^n \cdot F_M^*(\mathbf{x}^n)$ is positive which can lead to overfitting on samples producing these (e.g. mislabeled data). This makes AdaBoost sensitive to mislabeled data and can suffer from overfitting. The reader can find a more detailed description of boosting in the book (Bishop, 2006).

In computer vision decision trees were intensively applied as weak classifiers of boosting for object detection, e.g. face (Viola and Jones, 2001, 2004) or pedestrian detection (Dollár et al., 2014). Their main advantage is that they inherently select the relevant features, thus reducing the number of features to be calculated at prediction.

2.1.1.4 Neural Networks

Artificial Neural Networks were inspired by biology to train a complex function by a hierarchy of non-linear functions. The hierarchy is defined by layers where each layer takes the input from the previous layer(s), weights the input by trainable weights and applies a non-linear transformation. The layers defining the input and output are called input and output layer respectively, while the layers not directly observable (used as internal calculation stages) are called hidden layers.

Particularly successful are neural networks with many layers (hidden) being able to represent very complex non-linear functions. The sub-field of machine learning applying these many layer (deep) neural networks is referred to as *Deep Learning*. The basics of *Deep Learning* (neural networks) are introduced in the next subsection. For more information the reader is referred to the book (Ian et al., 2016).

2.1.2 Deep Learning

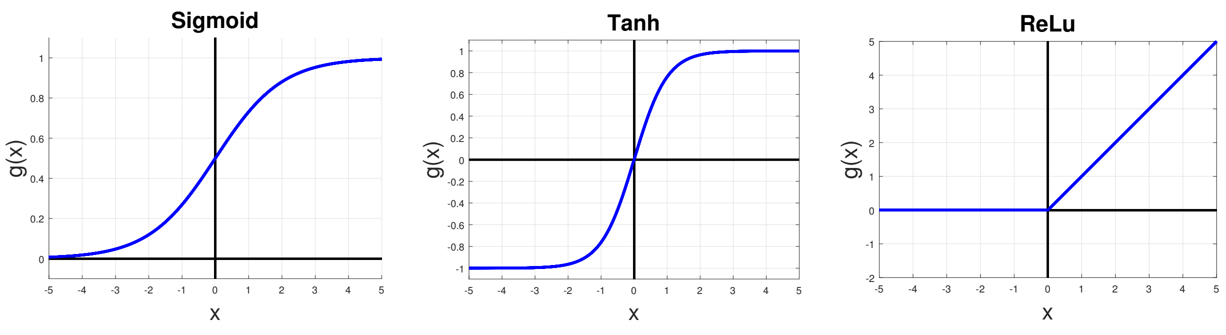
Deep Learning is one of the most successful areas of machine learning currently. Deep learning is the state of the art in image classification (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Szegedy et al., 2014; He et al., 2015), semantic segmentation (Chen et al., 2015; Schwing and Urtasun, 2015; Zheng et al., 2015), object detection (Ren et al., 2015a; Cai et al., 2015), action recognition (Gkioxari et al., 2015; Veeriah et al., 2015; Sun et al., 2015), image geolocalization (Workman et al., 2015; Weyand et al., 2016) and many other computer vision tasks and also other areas like speech recognition (Dahl, 2015; Hannun et al., 2014; Amodei et al., 2015) or game playing (Mnih et al., 2015) even against humans in the ancient board game *GO* (Silver et al., 2016).

A deep network with many hidden layers can give a very abstract and effective internal feature representation. Such a network can approximate highly non-linear functions efficiently and can transform non-linearly separable data distributions to a feature space where they are separable by simpler classifiers.

(Barron, 1993) has shown that already a network with one hidden layer and a proper non-linear function (e.g. sigmoid) can approximate a general smooth function to an arbitrary accuracy (i.e. integrated square error), but without a limit on the needed hidden units. This implies that a single hidden layer network with enough internal variables (hidden units) would be enough for any task. The problem is that this kind of approximation can become very inefficient in the number of internal units needed, while an approximation using larger depth can be much more efficient (needing less variables) and can create feature representations easier to interpret.

(Håstad, 1989) proved that for logical circuits there are functions being computable with polynomial number of circuit elements with depth k but requiring exponential number of elements when the depth is restricted to $k-1$. There is no such proof for general functions and those used in computer vision, but many experimental results indicate that deeper networks achieve better results, e.g. in image classification (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Szegedy et al., 2014; He et al., 2015).

The philosophy of *Deep Learning* is to train the prediction function $\mathbf{y} = f(\mathbf{x}, \mathbf{w})$ from end to end, from the raw input data to the final output. Hand-crafted features as used in "shallow learning" should be avoided as they prevent the automatic adoption to new data, e.g. hand crafted Histogram of Oriented Gradients (HOG) features (Dalal and Triggs, 2005) would need to be manually tuned for depth images. The features learned by the network become more and more high-level (abstract) in the deeper layers. Fig. 9 shows the visualization of the features in the different layers of a deep network.



(a) (b) (c)

Fig. 5. Common non-linear functions for neural networks. Sigmoid, hyperbolic tangent and rectified linear unit.

A neural network can be expressed as a composition of non-linear functions

$$\mathbf{h}^{t+1} = g(\mathbf{a}^t) = g(\mathbf{w}^t \cdot \mathbf{h}^t + \mathbf{b}^t) \tag{10}$$

where $g(x)$ is the non-linear function, $\mathbf{a}^t = \mathbf{w}^t \cdot \mathbf{h}^t + \mathbf{b}^t$ is the signal value before the non-linearity and $\mathbf{w}^t$, $\mathbf{b}^t$ are the trainable weights and biases for the layer t. See the Fig. 6 for the illustration of a network predicting $\mathbf{y} = f(\mathbf{x}, \mathbf{w}, \mathbf{b})$. Common non-linear functions are shown in Fig. 5.

◇ The *sigmoid:*

$$g_{\text{sigm}}(x) = \frac{1}{1 + e^{-x}} \tag{11}$$

◇ The *hyperbolic tangent:*

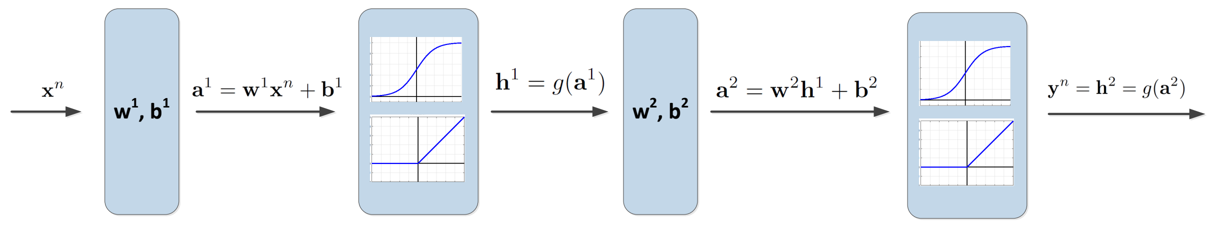
$$g_{\text{tanh}}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{12}$$

◇ The *rectified linear:*

$$g_{\text{relu}}(x) = max(0, x) \tag{13}$$

In deep networks it is common to use the Rectified Linear Unit (ReLu) as non-linearity. As they switch hidden units to zero they make the network sparse, while on the active units the function remains linear. This is advantageous for training the network with backpropagation since the gradients do not vanish or explode on these paths [4] and the computational implementation is also more efficient. ReLu functions are non-differentiable at zero and their gradient in the negative range is zero, but this does not cause a serious problem in the practice. This non-linear activation function is also biologically more plausible (Glorot et al., 2011).

[4] When composing many non-linear functions the gradient can become very low or high. This is referred as vanishing or exploding gradient problem during the training of neural networks.



$\mathbf{x}^n$
$\mathbf{w}^1, \mathbf{b}^1$
$\mathbf{a}^1 = \mathbf{w}^1\mathbf{x}^n + \mathbf{b}^1$
$\mathbf{h}^1 = g(\mathbf{a}^1)$
$\mathbf{w}^2, \mathbf{b}^2$
$\mathbf{a}^2 = \mathbf{w}^2\mathbf{h}^1 + \mathbf{b}^2$
$\mathbf{y}^n = \mathbf{h}^2 = g(\mathbf{a}^2)$

Fig. 6. Illustration of the forward pass (prediction) of a neural network with one hidden layer.

2.1.2.1 Training the network by Backpropagation

As (4) a neural network is trained by minimizing the loss w.r.t. the weight **w** and bias parameters **b**.

$$\mathbf{w}^*, \mathbf{b}^* = arg\min_{w} \sum_{n=1}^{N} \mathcal{L}(\hat{\mathbf{y}}^n, f(\mathbf{x}^n, \mathbf{w}, \mathbf{b})) + \lambda \mathcal{R}(\mathbf{w}, \mathbf{b}) \tag{14}$$

where λ is a constant. In contrast to SVMs (6) this problem is non-convex, the global minimum is not guaranteed to be found. But good solutions can be obtained using gradient descent methods. The gradients of the Loss function $\mathcal{L}$ w.r.t. the parameters $\mathbf{w}, \mathbf{b}$ can be calculated by backpropagating the gradients of the error signal (the loss function) to the trainable parameters. Since a neural network is a composition of multiple functions the chain rule (15) from calculus can be applied.

$$f(g(x))' = f'(g(x)) \cdot g'(x) \tag{15}$$

Fig. 7 shows an illustration for backpropagating the gradient of the loss function to the training parameters in a network with one hidden layer.

The gradient is calculated in mini-batches. The problem can be optimized by stochastic gradient descent with momentum. There are many possible local minima as many of the parameters are symmetric. But in practice these minima tend to be very similar in quality so a local minimum has probably a value close to the global (LeCun et al., 2015).

Large and deep neural networks have many parameters, e.g. the 19 layer VGG (Simonyan and Zisserman, 2014) has 144 million trainable weights. This makes these networks susceptible to overfitting. To prevent the overfitting regularization is used in many forms. A network typically applies weight decay in the form of $\mathcal{R}(\mathbf{w}) = ||\mathbf{w}||^2$. An additional regularization technique is *Dropout*. Dropout (Srivastava et al., 2014) prevents the co-adaptation of the weights to patterns only present in the training set by randomly deactivating units with their connections.

The augmentation of the training data can also be considered a regularization. The data is transformed by random transformations to which the task is invariant, e.g. rotate the image for road detection in aerial images.

Creating a certain network architecture can also prevent overfitting, a good examples widely used in computer vision are *CNNs*.

2.1.2.2 Convolutional Neural Networks (CNNs)

Sharing parameters in the network can efficiently reduce the number of parameters and the computation time if the nature of the data supports this.

In images certain features (e.g. edges) are spatially limited and the higher level features are agnostic to their exact spatial position. Consider the classification of a human face which is invariant to the exact position of the eye and the edges defining it. These features can

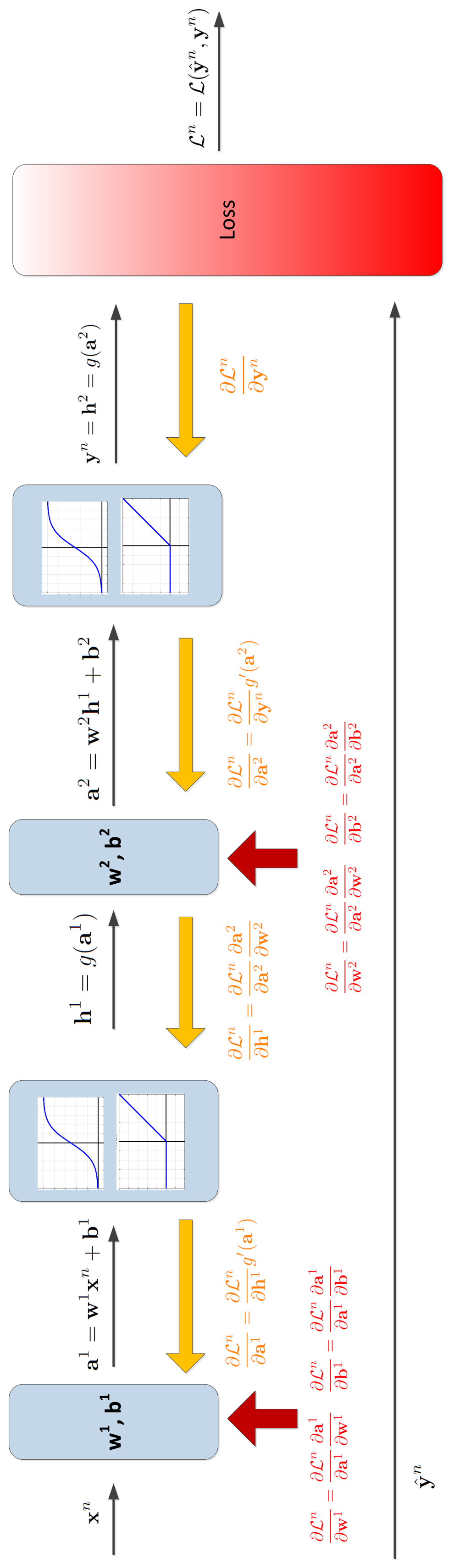

x^n
$\mathbf{w}^1, \mathbf{b}^1$
$a^1 = w^1 x^n + b^1$
$h^1 = g(a^1)$
$\mathbf{w}^2, \mathbf{b}^2$
$a^2 = w^2 h^1 + b^2$
$y^n = h^2 = g(a^2)$
Loss
$\mathcal{L}^n = \mathcal{L}(\hat{y}^n, y^n)$
$\hat{y}^n$
$\frac{\partial \mathcal{L}^n}{\partial y^n}$
$\frac{\partial \mathcal{L}^n}{\partial a^2} = \frac{\partial \mathcal{L}^n}{\partial y^n} g'(a^2)$
$\frac{\partial \mathcal{L}^n}{\partial h^1} = \frac{\partial \mathcal{L}^n}{\partial a^2} \frac{\partial a^2}{\partial w^2}$
$\frac{\partial \mathcal{L}^n}{\partial a^1} = \frac{\partial \mathcal{L}^n}{\partial h^1} g'(a^1)$
$\frac{\partial \mathcal{L}^n}{\partial w^2} = \frac{\partial \mathcal{L}^n}{\partial a^2} \frac{\partial a^2}{\partial w^2}$
$\frac{\partial \mathcal{L}^n}{\partial b^2} = \frac{\partial \mathcal{L}^n}{\partial a^2} \frac{\partial a^2}{\partial b^2}$
$\frac{\partial \mathcal{L}^n}{\partial w^1} = \frac{\partial \mathcal{L}^n}{\partial a^1} \frac{\partial a^1}{\partial w^1}$
$\frac{\partial \mathcal{L}^n}{\partial b^1} = \frac{\partial \mathcal{L}^n}{\partial a^1} \frac{\partial a^1}{\partial b^1}$

Fig. 7. Illustration of the backward pass of a neural network with one hidden layer. The backpropagation of the Loss function gradient is shown in orange, the gradients used for the parameter update are shown in red and the forward pass is in black. The regularization term $\mathcal{R}(\mathbf{w}, \mathbf{b})$ is not shown. View this figure rotated.

occur at multiple positions so the same weight parameters can be applied around the whole image. This invariance and weight sharing can be expressed by the convolution operation and a layer aggregating information in a spatial range.

CNNs use convolutional and max-pooling layers to implement the weight sharing and the invariance to translations. A convolutional layer consists of S filters of spatial size $(2\cdot D+1)\times(2\cdot D+1)$, spectral size C [5] with trainable weights $k_{m,n,c}$ where $m,n \in [-D,\ldots,D]$ define the spatial neighborhood of the kernel while $c \in [1,\ldots,C]$ defines the spectral one. The weighting $\mathbf{a}^{t+1} = \mathbf{w}^t \cdot \mathbf{h} + \mathbf{b}$ used in previous (fully connected) networks is replaced by

$$a^{t+1}_{i,j,s} = \sum_{m=-D}^{D}\sum_{n=-D}^{D}\sum_{c=1}^{C}\left(h^t_{i-m,j-n,c}\cdot k^{t,s}_{m,n,c}\right) + b^t_{i,j,s} \tag{16}$$

where $a^{t+1}_{i,j,s}$ is the value of the hidden unit at spatial location i,j, at spectral channel s and in the layer $t+1$. The layer of c spectral channels is transformed to s channels, e.g. in the VGG network (Simonyan and Zisserman, 2014) the input has 3 channels (i.e. RGB) while the first convolutional layer defines 64 channels.

A pooling layer aggregates information in the spatial range $[-P,\ldots,P]$. The most common is to use max pooling layers which propagate the maximal value further to the later layers (17).

$$p^t_{i,j} = \max_{m\in[i-P,i+P],n\in[j-P,j+P]} h^t_{m,n} \tag{17}$$

Fig. 8 illustrates a convolutional network with 2 layers of trainable parameters in one spatial and one spectral dimension. Fig. 9 shows features of a CNN in different layers.

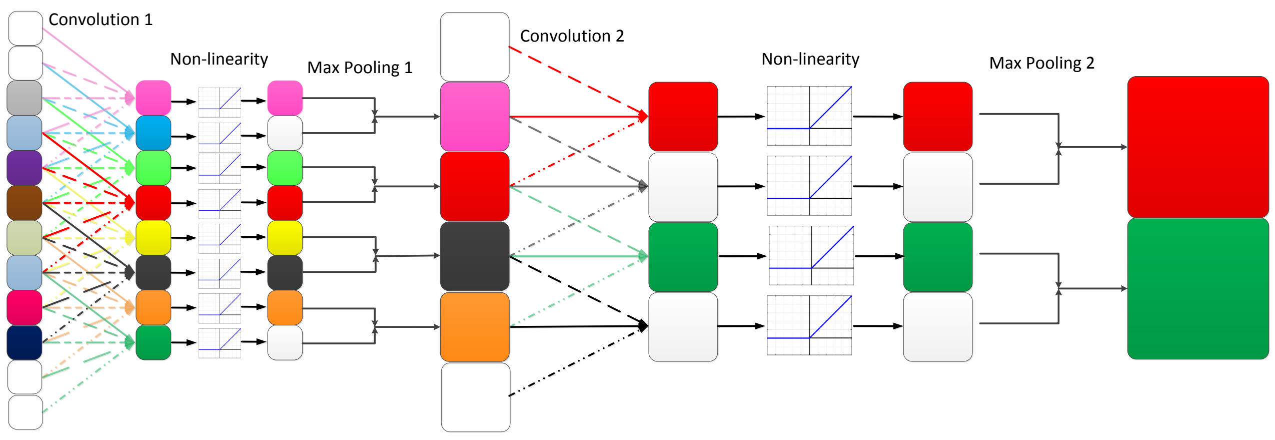


Fig. 8. Illustration of a CNN where the first convolution layer has a kernel size of 5 and the second 3. The max pooling layers have a receptive field of 2. The convolution is padded with zeros. The edges in one convolution layer with the same dash type have the same weights. As we move forward in the network the spatial resolution decreases due to the max pooling.

Applying convolutional and max pooling layers can be interpreted as transforming the input to a higher spectral dimensional space where the data becomes easier separable while the spatial resolution is decreased.

Many networks also use a spatial feature normalization layer. For these and further details (e.g. Recurrent Neural Networks and Autoencoders) the reader is referred to the book (Ian et al., 2016).

[5] The spectral size is the number of elements per pixel position, e.g. 3 for a RGB image.

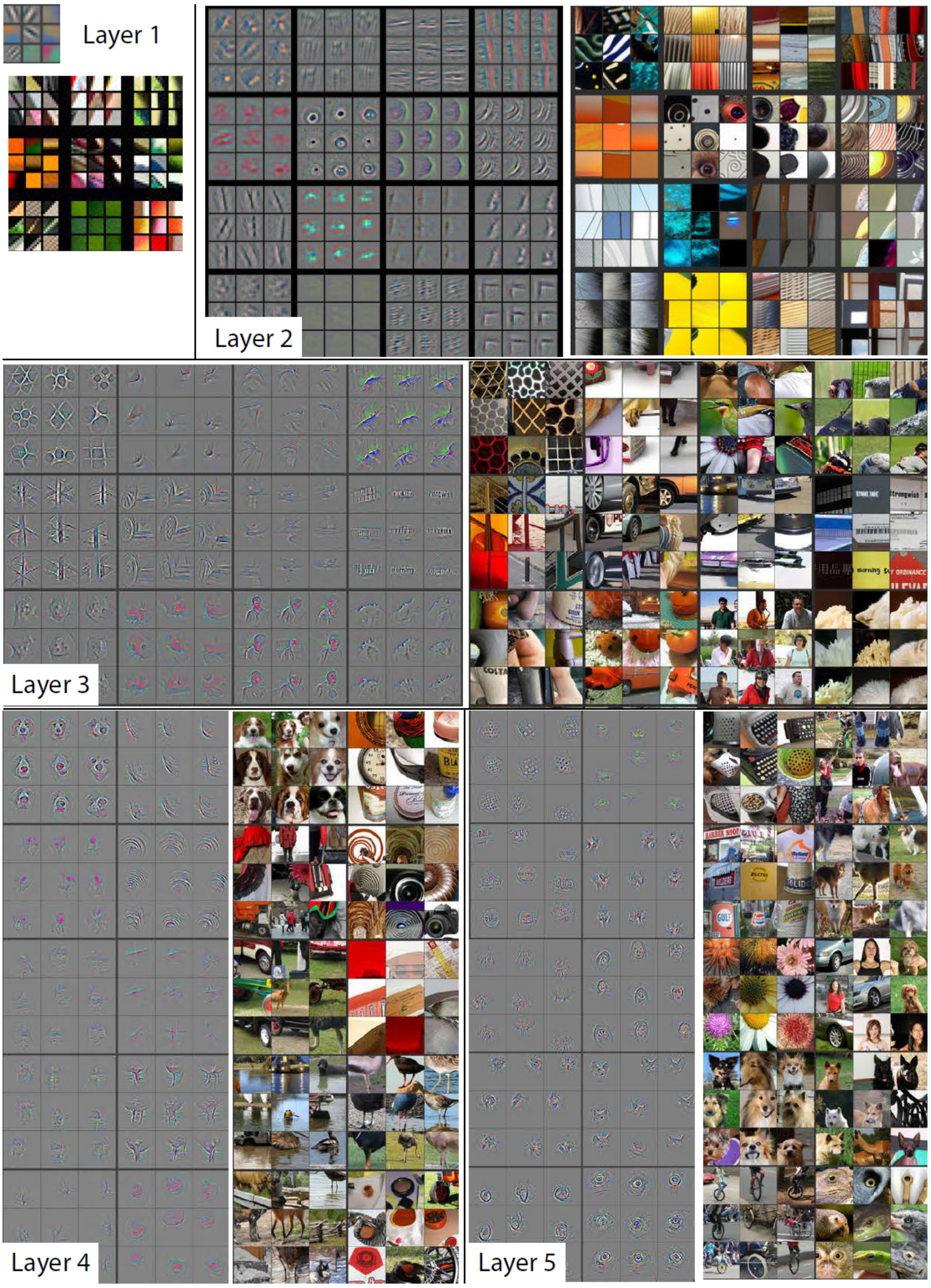


Fig. 9. From (Zeiler and Fergus, 2014). Visualizing different layers of a deep convolutional neural network. The left images show reconstructed patterns which cause high activations in a feature map. The right images show the corresponding image patches. The reconstructed patterns are projected down to pixel space using the deconvolutional network approach of (Zeiler and Fergus, 2014). The first layers extract simpler features (e.g. edges, corners, etc.), while the later layers describe complex shapes with greater variance.

2.1.3 Graphical Models

The methods presented so far consider only a single output variable or variables being independent. For many problems this is not the case, the output has a structure. For example in pixelwise semantic segmentation the labels of the neighboring pixels are correlated as they might belong to the same object.

A naive implementation would be to handle each output configuration as an own state in a single output variable. This would mean that y has y_c^d possible output states, where y_c is the number of states of a single output variable and d is the number of the output variables. This formulation can become easily intractable as the number of output states increases exponentially by the number of output variables.

A much more efficient approach is to consider a function which also depends on the output variables and expresses the relation between the outputs. Then the problem can be formulated by minimizing an energy function (alternatively one could also maximize a function) in the form of (18).

$$\mathbf{y} = f(\mathbf{x}) = \arg\min_{y} E(\mathbf{y}, \mathbf{x}) \tag{18}$$

The family of problems with structured output variables is also referred to as *Structured Prediction*. A special case of *Structured Prediction* is a *Graphical Model* which expresses the dependency of the variables in form of a graph. The nodes represent the variables and the edges define the conditional dependence between the variables.

A model described by a directed acyclic graph is called a *Bayesian Network*. For more details about Bayesian Networks the reader is referred to (Bishop, 2006).

If the graph has undirected edges it is named a Markov Random Field (MRF). MRFs (and Graphical Models in general) are very popular in computer vision since the conditional dependencies in the content of images can be modeled efficiently by undirected graphs. A good overview of graphical models and structured prediction in computer vision can be found in (Nowozin and Lampert, 2011).

2.1.3.1 Markov Random Fields (MRFs)

A MRF is a set of random variables where the conditional dependency of the variables is described by an undirected graph. The random variables have the Markov property: the conditional probability distribution of the variables only depend on the neighbors (nodes connected by an edge), not the neighbors of the neighbors. In other words, the Markov blanket of the node is the adjacent nodes. If the MRF is defined by the graph G with nodes (variables) $\mathbf{y} = [y_1, y_2, \ldots, y_T]$, then the joint probability distribution can be calculated as:

$$p(\mathbf{y}) = \frac{1}{Z} \prod_{C \in \mathcal{C}(G)} \Psi_C(\mathbf{y}_C) \tag{19}$$

where $\mathcal{C}(G)$ denote the cliques [6] in the graph G and $\Psi_C(\mathbf{y}_C)$ are the potential functions, also called factors. Z is a normalizing constant, called partition function, ensuring that a proper probability distribution is obtained. It is calculated as

$$Z = \sum_{\mathbf{y} \in \Upsilon} \prod_{C \in \mathcal{C}(G)} \Psi_C(\mathbf{y}_C), \tag{20}$$

[6] A clique in a graph is a subset of the nodes where all nodes are connected (the subgraph is a complete graph).

the sum over all possible output configurations $\Upsilon = \gamma_1 \times \gamma_2 \cdots \times \gamma_T$, where γ_i is the set of possible states of the variable y_i.

The potentials (factors) $\Psi_C(\mathbf{y}_C)$ can be arbitrary non-negative functions defining the interaction of the variables $y_i \in \mathbf{y}_C$ without having a specific probabilistic interpretation.

Since the potentials can be chosen arbitrary, the graph G alone cannot define the joint probability distribution exactly, multiple factorizations are possible. An exact factorization can be described by a *factor graph*. The factor graph depicts the variables as circles, the factors as squares and if the variable and the factor are connected by an edge, then the potential is a function of that variable. Fig. 10 (a) shows an MRF given by a graph. It could be factorized to pairwise potentials or to a single third order potential.

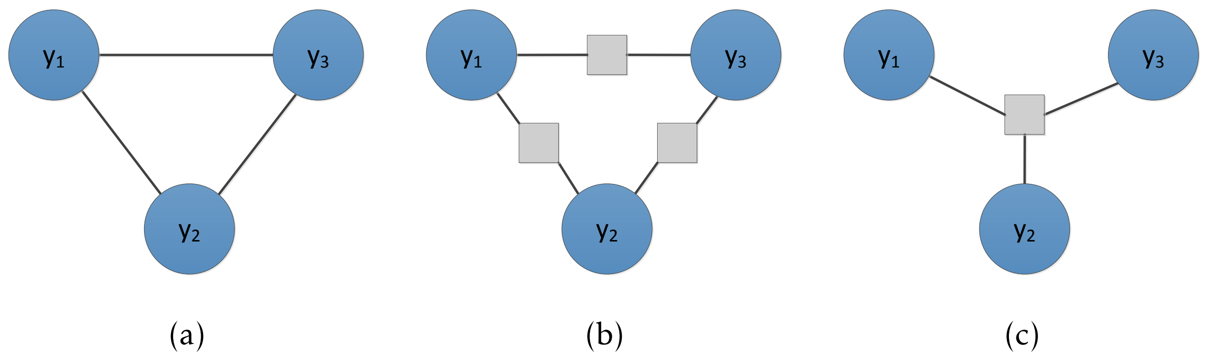


Fig. 10. A MRF with 3 variables defined by a graph (a) and two possible factorizations (b-c). The first (b) defines pairwise potentials $\Psi(y_i, y_j), i \neq j$. The second (c) defines one third order potential $\Psi(y_1, y_2, y_3)$.

In many problems certain variables can be observed directly, without any uncertainty, e.g. the pixel values of an image. These variables do not have to be considered as random variables and they can be conditioned on. The MRFs containing conditioned variables are also called Conditional Random Fields (CRFs). These variables are often referred to as input variables and are denoted by $\mathbf{x}$. A potential $\Psi(y_i, x_i)$ of a conditioned variable can be simply written as potential not a function of x_i:

$$\Psi(y_i, x_i) = \Psi(y_i | x_i) = \Psi_{un}(y_i) \tag{21}$$

In computer vision the conditioned variables are often expressed as unary potentials, functions of a single variable (21). A typical unary potential in computer vision is an output of a classifier. Consider the problem of road detection in aerial images using a MRF. The unary potentials express the evidence in the image in form of a classifier providing a probability for a pixel being road/non-road. The pairwise (or higher order) potentials typically reflect on the spatial structure, e.g. the road has to be smooth. Fig. 11 illustrates a CRF.

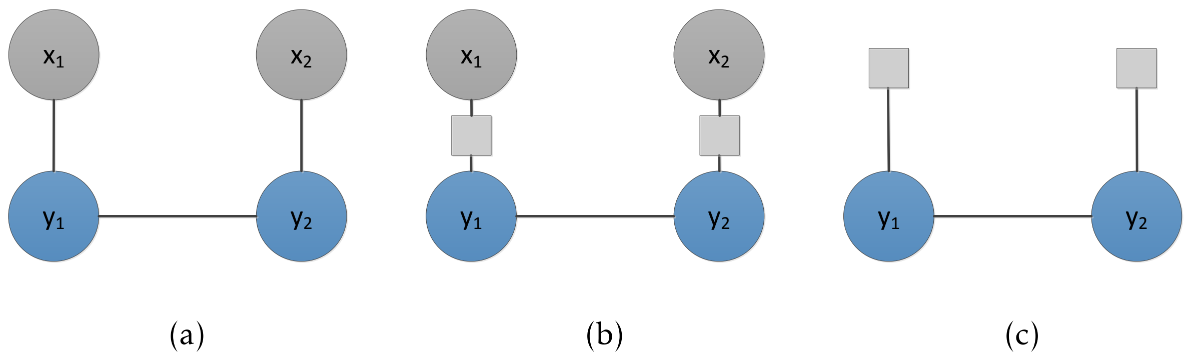


Fig. 11. The random variables y_1, y_2 are conditioned on x_1, x_2 in a CRF (a). The potentials $\Psi(y_1, x_1)$ and $\Psi(y_2, x_2)$ in (b) can be simplified to unaries $\Psi(y_1), \Psi(y_2)$ (c).

If the potential functions are restricted to be exponential

$$\Psi_i(\mathbf{y}) = \exp(-\phi_i(\mathbf{y})) \qquad \text{and} \qquad \phi_i(\mathbf{y}) = -log(\Psi_i(\mathbf{y})), \tag{22}$$

then the probability for a given output state $\mathbf{Y} \in \Upsilon$ can be formulated as

$$p(\mathbf{y} = \mathbf{Y}) = \frac{1}{Z} \exp\left(-\sum_{i \in \Xi} \phi_i(\mathbf{y})\right) \tag{23}$$

where Z is:

$$Z = \sum_{\mathbf{y} \in \Upsilon} \exp\left(-\sum_{i \in \Xi} \phi_i(\mathbf{y})\right) \tag{24}$$

and Ξ denotes the set of all potentials (factors) in our model. This reformulation allows to handle the problem of finding the maximum probability as finding the minimum in the energy domain:

$$\operatorname*{argmax}_{\mathbf{y}}(p(\mathbf{y})) = \operatorname*{argmax}_{\mathbf{y}}\left(\frac{1}{Z}\exp\left(-\sum_{i \in \Xi} \phi_i(\mathbf{y})\right)\right) = \tag{25}$$

$$\operatorname*{argmin}_{\mathbf{y}}(E(\mathbf{y})) = \operatorname*{argmin}_{\mathbf{y}}\left(\sum_{i \in \Xi} \phi_i(\mathbf{y})\right) \tag{26}$$

2.1.3.2 Inference

When a problem is modeled as a MRF two important questions are mostly raised:

◇ *Probabilistic inference*: What is the marginal probability distribution of the individual variables (27) and factors and what is the value of the partition function Z?

$$\mu_i(y_i) = p(y_i = Y_i|\mathbf{x}) \quad \forall i \in [1 \dots T], \forall Y_i \in \gamma_i \tag{27}$$

This reasons about the individual variables, but it cannot provide the global output with the highest probability.

◇ *Maximum a prosteriori (MAP) inference*: Which variable configuration has the highest probability?

(28) This is the mode of the joint probability distribution (19). Since the exact probability is not relevant just its location, the partition function does not have to be computed. The MAP inference is analogous to energy minimization, thus it can also be applied without defining a probability distribution over the variables, just an energy function.

$$\mathbf{y}^* = \operatorname*{argmin}_{\mathbf{y}} \sum_{C \in \mathcal{C}(G)} \phi_C(\mathbf{y}_C) \tag{28}$$

Both inference problems are in general NP hard problems (Kolmogorov and Zabin, 2004). But for certain graph structures and potential functions there are exact and efficient inference algorithms.

Exact and efficient inference if the graph structure is restricted:

◇ *Inference in a chain*: In a chain the inference problems can be solved exactly and efficiently using dynamic programming. Since this is used extensively in my work, a detailed description is given in section 2.1.3.2.1.

- ◇ *Inference in a tree*: If the graph has no loops, exact inference is possible via the *Sum Product* algorithm (Waltz, 1982; MacKay, 2002). This is a dynamic programming algorithm also called *belief propagation*. First a root node is chosen in the factor graph and messages are sent to this from all other nodes. Then the messages are sent backward from the root. When all the messages arrived the partition function and the marginal distributions can be computed. The variant to calculate MAP inference is the *max-sum algorithm*.
- ◇ *Junction tree algorithm*: Even if the graph contains loops the junction tree algorithm can convert it to a tree by *triangulating* the graph and merging the variables of cliques (Bishop, 2006). However, the complexity is exponential in the size of the largest number of variables in a clique. This can make the algorithm inefficient.

Exact and fast inference if the potentials are restricted:

- ◇ *Graph Cut*: If the nodes of the MRF are binary variables, the network has only pairwise and/or unary potentials and the pairwise potential is submodular, then the energy minimization (MAP inference) can be solved via *graph cuts* (Kolmogorov and Zabin, 2004).

Approximate inference techniques can be applied for general graphs and potentials but the global optimum is not guaranteed to be found and the efficiency depends on the specific problem. There are many techniques for performing approximate solutions, here only a few are listed. (Nowozin and Lampert, 2011) provide a more comprehensive description of inference methods. The *OpenGM*[7] library provides implementation for many inference algorithms.

- ◇ *Loopy belief propagation*: The *Sum Product* (or *max-sum*) algorithm gives exact result for trees, but it can also be applied for graphs with loops. In contrast to trees the messages can circle around in the graph many times. Therefore a schedule is defined to control the flow of the messages. For certain graphs this algorithm converges, for others not. (Frey and MacKay, 1997; Bishop, 2006)
- ◇ *Mean-field*: The original probability distribution $p(\mathbf{y})$ is approximated by a distribution $q(\mathbf{y})$ for which the inference is tractable. $q(\mathbf{y})$ is searched by minimizing the *Kullback-Leibler divergence* between the approximate and the original probability distribution. If $q(\mathbf{y})$ is close to $p(\mathbf{y})$, then the marginals are also approximated with a small error. (Jordan et al., 1999; Nowozin and Lampert, 2011)
- ◇ *BCD*: In certain graphs we can choose blocks of variables for which we can solve the inference exactly and fast (e.g. a chain or a tree). BCD works by iteratively selecting a block of variables, solving the inference for this, update these variables if the energy can be decreased and repeat this until convergence. A more detailed description is provided in the paragraph 2.1.3.2.2. This finds a local minimum. The algorithm conditions on the rest of the variables, thus the higher order potentials fold to lower, e.g. a pairwise to a unary.

2.1.3.2.1 MAP Inference in a chain

If the graph is a chain, exact and efficient inference is possible using dynamic programming. This description considers an MRF with T variables where each variable has S states. The energy function (29) of the MRF is factorized to unary $\phi_{un}(y^{(t)})$ and pairwise potentials $\phi_{pw}(y^{(t)}, y^{(t+1)})$ where $t = 1, 2, \ldots, T$ is the index of the variable. For this explanation the lower index $y_i, i = 1, 2, \ldots, S$ denotes a specific value of a variable. If there is no lower index the symbol defines a vector of all the states.

$$E(\mathbf{y}) = \sum_{t=1}^{T} \phi_{un}(y^{(t)}) + \sum_{t=1}^{T-1} \phi_{pw}(y^{(t)}, y^{(t+1)}) \tag{29}$$

[7] http://hciweb2.iwr.uni-heidelberg.de/opengm/

The naive implementation to find the minimum energy would need S^T steps by searching over all possible output configurations.

By applying dynamic programming Algorithm 1 can solve the MAP inference problem $\mathbf{y}^* = \text{argmin}_y(E(\mathbf{y}))$ in $O(TS^2)$ complexity. The key is starting from one end of the chain, so a minimum energy $\delta_i^{(t)}$ can be defined for a given state i depending only on the minimum values of the previous variable, the pairwise potential and the current unary value (30). At the end of the chain $\delta^{(T)}$ contains the contribution of all variables to the energy. A simple minimum search can give the minimum for the last variable and the state sequence resulting in this variable can be followed back to the start of the chain. This is equivalent to finding the shortest path in a graph where each state of a variable is a node (*trellis diagram*). Fig. 12 illustrates such a graph.

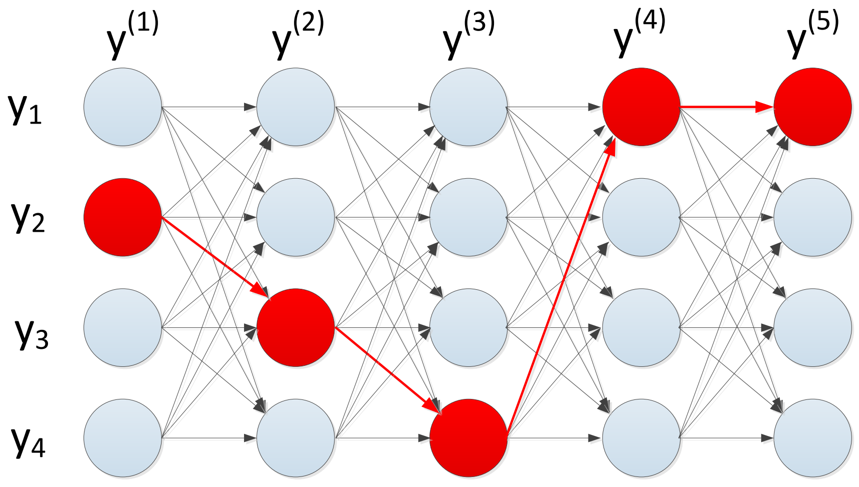


Fig. 12. Trellis diagram illustration of the MAP inference in a chain with 5 variables each having 4 states. The $\mathbf{y}^* = \text{argmin}_y(E(\mathbf{y}))$ state sequence is the shortest path if the distance is the sum of the unary potentials visited and the pairwise potentials (edges) connecting the variables. The shortest path (i.e. lowest energy) is shown in red.

$$\delta_i^{(t)} = \phi_{un}(y_i^{(t)}) + \min_{j\in[1,S]}\left(\delta_j^{(t-1)} + \phi_{pw}(y_j^{(t-1)}|y_i^{(t)})\right) \tag{30}$$

Algorithm 1 MAP Inference in a chain by dynamic programming

```
Initialize $\delta^1 = \phi_{un}(\mathbf{y}^{(1)})$
for $t = 2$ to $T$ do
  for $i = 1$ to $S$ do
    $\delta_i^{(t)} = \phi_{un}(y_i^{(t)}) + \min_{j\in[1,S]}\left(\delta_j^{(t-1)} + \phi_{pw}(y_j^{(t-1)}|y_i^{(t)})\right)$
  end for
end for
$y^{*,(T)} = \text{argmin}_i(\delta_i^{(T)})$
for $t = T-1$ to $1$ do
  $y^{*,(t)} = \text{argmin}_i\left(\delta_i^{(t)} + \phi_{pw}(y_i^{(t)}|y^{*,(t+1)})\right)$
end for
return $\mathbf{y}^*$
```

The minimum search in (30) has $O(S)$ complexity and since this has to be computed for each state in each variable the complexity of the whole algorithm is $O(TS^2)$. When applied for finding the MAP state sequence in a Hidden Markov Model (HMM) this algorithm is referred to as the *Viterbi* algorithm (Viterbi, 1967).

2.1.3.2.2 BCD Inference

Block coordinate descent (BCD) inference is applied in this thesis to solve the inference in graphs containing many loops, thus imposing an NP-hard problem. BCD is a version of the *coordinate descent* optimization algorithm, but instead of searching the local minimum along one coordinate, the search is done over a block of variables. In a graphical model this means that for an iteration certain variables are kept fix and so they do not have to be considered random variables. This can convert a graph structure defining a NP-hard inference problem to a graph where the MAP inference for a given iteration can be solved better. Favorable is to simplify the graph to a tree or chain because in this case the MAP inference problem can be solved exactly in polynomial time. The variables are only updated if the resulting new solution has lower total energy than before.

An illustration for solving a problem defined by a grid is shown in Fig. 13. The energy for this graph is defined by Equation (31). During a horizontal update the Equation (32) is minimized. The pairwise potentials fold to unaries as shown on Fig. 13 (b-c). Algorithm 2 shows the pseudocode for the inference algorithm.

$$E(\mathbf{y}) = \sum_{i=1}^{N}\sum_{j=1}^{N}\phi_{un}(y_i^j) + \sum_{j=1}^{N}\sum_{i=1}^{N-1}\phi_{pw}(y_i^j, y_{i+1}^j) + \sum_{i=1}^{N}\sum_{j=1}^{N-1}\phi_{pw}(y_i^j, y_i^{j+1}) \tag{31}$$

One step:

$$E^b(\mathbf{y}_i) = \sum_{j=1}^{N}\phi_{un}(y_i^j) + \sum_{j=1}^{N-1}\phi_{pw}(y_i^j, y_i^{j+1}) + \sum_{j=1}^{N}\phi_{pw}(y_i^j|y_{i-1}^j) + \sum_{j=1}^{N}\phi_{pw}(y_i^j|y_{i+1}^j), \tag{32}$$

where $\mathbf{y}_i$ denotes one row of variables.

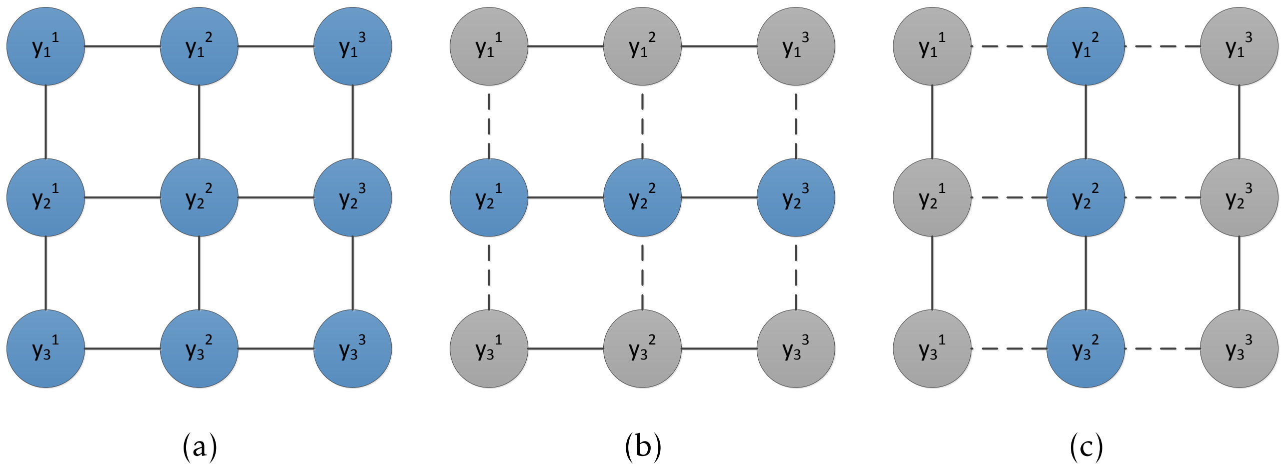


Fig. 13. The graph is shown in (a). An iteration for updating a row of variables is shown in (b), while (c) depicts an inference iteration in a column. The gray circles show variables which are kept fix. The dashed lines show the pairwise potentials folded to unaries.

2.1.3.3 Structured Support Vector Machines (SSVMs)

Similarly to classifiers and deep learning, graphical models can also have trainable parameters. In my work SSVMs are applied to train the parameters of the applied graphical models.

Algorithm 2 Block Coordinate Descent (BCD)

```
 1: Initialize ($\mathbf{y}$) by minimizing Equation (31) for the blocks of variables independently.
 2: repeat
 3:     for all blocks of variables $\mathbf{y}^b$ do
 4:         Minimize Equation (31) w.r.t $\mathbf{y}^b$ holding the rest fixed, e.g. as in Equation (32). $\mathbf{y}^{b*} = \min_{\mathbf{y}^b} E^b(\mathbf{y}^b)$
 5:         Calculate the total energy (31) with the values $\mathbf{y}^{b*}$.
 6:         if energy can be reduced then
 7:             Update $\mathbf{y}$ by $\mathbf{y}^{b*}$
 8:         end if
 9:     end for
10: until no energy reduction or max number of iterations is reached
```

In the probability domain the potentials have to be non-negative $\Psi(\mathbf{y}) \geq 0$, but $\Psi(\mathbf{y}) = \exp(-\phi(\mathbf{y}))$ allows the potential to be an arbitrary function in the energy domain. This gives the freedom to introduce linear weights $\mathbf{w} = [w_1, w_2, \ldots, w_F]$, $F = |\Xi|$ for each feature defining the impact of the potentials. Using these weights the energy function becomes:

$$E(\mathbf{y}, \mathbf{x}, \mathbf{w}) = \sum_{i \in \Xi} w_i \phi_i(\mathbf{y}|\mathbf{x}) = \sum \mathbf{w}^t \phi(\mathbf{y}|\mathbf{x}) \tag{33}$$

By adjusting the weights the MAP inference output (34) can be influenced.

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmin}}\, E(\mathbf{y}, \mathbf{x}, \mathbf{w}) = \underset{\mathbf{y}}{\operatorname{argmin}} \left(\sum \mathbf{w}^t \phi(\mathbf{y}|\mathbf{x}) \right) = f(\mathbf{x}, \mathbf{w}) \tag{34}$$

Ideally the $\mathbf{w}$ weights should be set in a way that ”good” result are obtained from the MAP inference. If there are N training samples $\mathbf{x}^n, \hat{\mathbf{y}}^n$ the ”goodness” of the output can be measured by the loss function (as defined in Equation (2)).

$$\mathcal{L}(\hat{\mathbf{y}}^n, \mathbf{y}) = \mathcal{L}(\hat{\mathbf{y}}^n, f(\mathbf{x}^n, \mathbf{w})) \tag{35}$$

where $f(\mathbf{x}^n, \mathbf{w})$ is a structured prediction function:

$$f(\mathbf{x}^n, \mathbf{w}) = \underset{\mathbf{y}}{\operatorname{argmin}}\, (E(\mathbf{y}, \mathbf{x}^n, \mathbf{w})) \tag{36}$$

The standard learning equation of (4) could be applied but since $\mathcal{L}(\hat{\mathbf{y}}^n, f(\mathbf{x}^n, \mathbf{w})$ is piecewise linear, gradient based techniques can not handle this (Nowozin and Lampert, 2011). (Zhang, 2004) showed that the minimization of the *convex upper bound* can also provide good results.

SSVMs train the $\mathbf{w}$ weights of a structured problem by minimizing $\ell(\mathbf{x}^n, \hat{\mathbf{y}}^n, \mathbf{w})$, which is the the convex upper bound of the loss $\mathcal{L}(\hat{\mathbf{y}}^n, \mathbf{y})$.

$$\ell(\mathbf{x}^n, \hat{\mathbf{y}}^n, \mathbf{w}) = -\min_{y} (E(\mathbf{y}, \mathbf{x}^n, \mathbf{w}) - \mathcal{L}(\hat{\mathbf{y}}^n, \mathbf{y}) - E(\hat{\mathbf{y}}^n, \mathbf{x}^n, \mathbf{w})) \tag{37}$$

Note that this definition is different from the one in (Nowozin and Lampert, 2011) and (Tsochantaridis et al., 2005) because here the output is defined as a result of an energy minimization, not score maximization. The proof why this formulation is also an upper bound and convex w.r.t. to $\mathbf{w}$ is provided later. The training can be defined as:

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{N}\sum_{n=1}^{N} \ell(\mathbf{x}^n, \hat{\mathbf{y}}^n, \mathbf{w}) \tag{38}$$

where C is a constant defining the importance of the loss compared to the regularization.

The upper bound in (37) generalizes the *Hinge loss* to multiple outputs and the problem formulation of (38) can be interpreted as a *maximum margin* training procedure. The name Structured Support Vector Machine implies this (Tsochantaridis et al., 2005).

To proof that $\ell(\mathbf{x}^n, \hat{\mathbf{y}}^n, \mathbf{w})$ is an upper bound for the loss function the following has to be proved:

$$\mathcal{L}(\hat{\mathbf{y}}^n, \mathbf{y}) \leq \mathcal{L}(\hat{\mathbf{y}}^n, \mathbf{y}) - E(f(\mathbf{x}^n, \mathbf{w}), \mathbf{x}^n, \mathbf{w}) + E(\hat{\mathbf{y}}^n, \mathbf{x}^n, \mathbf{w}) \leq \tag{39}$$

$$\leq -\min_{y}\left(E(\mathbf{y}, \mathbf{x}^n, \mathbf{w}) - \mathcal{L}(\hat{\mathbf{y}}^n, \mathbf{y}) - E(\hat{\mathbf{y}}^n, \mathbf{x}^n, \mathbf{w})\right) = \ell(\mathbf{x}^n, \hat{\mathbf{y}}^n, \mathbf{w}) \tag{40}$$

(39) is true if $E(\hat{\mathbf{y}}^n, \mathbf{x}^n, \mathbf{w}) \geq E(f(\mathbf{x}^n, \mathbf{w}), \mathbf{x}^n, \mathbf{w})$ which is always the case since $f(\mathbf{x}^n, \mathbf{w})$ is the minimum location by definition.

For any two functions $p(x), q(x)$ it is true that $p(x) - q(x) \geq \min_x(p(x) - q(x))$, since the minimization includes $p(x) - q(x)$. By applying this:

$$E(f(\mathbf{x}^n, \mathbf{w}), \mathbf{x}^n, \mathbf{w}) - \mathcal{L}(\hat{\mathbf{y}}^n, \mathbf{y}) \geq \min_{y}\left(E(f(\mathbf{x}^n, \mathbf{w}), \mathbf{x}^n, \mathbf{w}) - \mathcal{L}(\hat{\mathbf{y}}^n, \mathbf{y})\right) \tag{41}$$

$$\mathcal{L}(\hat{\mathbf{y}}^n, \mathbf{y}) - E(f(\mathbf{x}^n, \mathbf{w}), \mathbf{x}^n, \mathbf{w}) \leq -\min_{y}\left(E(f(\mathbf{x}^n, \mathbf{w}), \mathbf{x}^n, \mathbf{w}) - \mathcal{L}(\hat{\mathbf{y}}^n, \mathbf{y})\right) \tag{42}$$

$E(\hat{\mathbf{y}}^n, \mathbf{x}^n, \mathbf{w})$ is a constant w.r.t. $\mathbf{y}$, thus

$$\mathcal{L}(\hat{\mathbf{y}}^n, \mathbf{y}) - E(f(\mathbf{x}^n, \mathbf{w}), \mathbf{x}^n, \mathbf{w}) + E(\hat{\mathbf{y}}^n, \mathbf{x}^n, \mathbf{w}) \leq -\min_{y}\left(E(\mathbf{y}, \mathbf{x}^n, \mathbf{w}) - \mathcal{L}(\hat{\mathbf{y}}^n, \mathbf{y}) - E(\hat{\mathbf{y}}^n, \mathbf{x}^n, \mathbf{w})\right) \tag{43}$$

which proves $\ell(\mathbf{x}^n, \hat{\mathbf{y}}^n, \mathbf{w})$ to be an upper bound.

To proof that $\ell(\mathbf{x}^n, \hat{\mathbf{y}}^n, \mathbf{w})$ is convex, it can be rewritten by applying that for any function $p(x)$ $\min(p(x)) = -\max(-p(x))$:

$$\ell(\mathbf{x}^n, \hat{\mathbf{y}}^n, \mathbf{w}) = \max_{\mathbf{y}}\left(-E(\mathbf{y}, \mathbf{x}^n, \mathbf{w}) + \mathcal{L}(\hat{\mathbf{y}}^n, \mathbf{y}) + E(\hat{\mathbf{y}}^n, \mathbf{x}^n, \mathbf{w})\right) \tag{44}$$

This is a maximum over many affine functions (which are convex) over $\mathbf{w}$, thus $\ell(\mathbf{x}^n, \hat{\mathbf{y}}^n, \mathbf{w})$ is also convex over $\mathbf{w}$.

(38) is a convex quadratic problem w.r.t. $\mathbf{w}$. An efficient method to solve this is the cutting plane algorithm of (Tsochantaridis et al., 2005) using slack variables. During the training the $\min_y(E(\mathbf{y}, \mathbf{x}^n, \mathbf{w}) - \mathcal{L}(\hat{\mathbf{y}}^n, \mathbf{y}))$ minimization problem has to be solved. This is also called the *loss augmented inference*. If the loss is expressed as a similar potential as the other potentials in the MRF the same inference algorithm can be applied. If there is no access to the ground truth of the variables directly, they can be approximated by doing inference using the ground truth labeling as features. This output is called the *Oracle*.

2.2 Image geolocalization

The geolocation of an image can relate the visual input with other type of information, which makes it very informative. The location of the image provides also the location of the camera which can be employed as self-localization. Everyday, personal images (e.g. a photograph of our family smiling) can be interpreted without any geolocation, however the location can be a useful plus. This is hardly the case for aerial images containing macro scale information (e.g. the position and extent of buildings, vegetation, roads etc.). This makes it particularly important to have geographic information for aerial images. Aerial images can be best used in Geographic Information Systems (GISs) where they are overlaid with maps in the same projection.

2.2.1 Map projections

Maps and orthorectified aerial images are projections of the surface of the Earth. A projection needs the definition of the shape of the Earth and the definition of the projection from this shape to a visualizable format, e.g. a paper or a screen.

Throughout the course of history many map projections were created and still today there are different systems in use. The reason for not using a single standard system worldwide is the complex shape of the Earth (it is not a mathematical sphere). The mathematical description of this shape can be either simple (defined by few parameters), adjusted for a given region or adjusted for the whole globe, but these three objectives cannot be fulfilled simultaneously.

On local scale we define 3D points in the Euclidean coordinate system by the x, y, z coordinates, but on large geographical scales this is not practical. The objects of our interest mainly exist on the surface of our planet or very close to it. This would make the Euclidean coordinate representation very sparse and hard to interpret, visualize or calculate. Before the invention of digital computers this was a serious issue and it is still relevant today. The *geographic coordinate system* gives a more practical description for a point on a geographical scale. It defines a 3D point as a point on the Earth's surface expressed by two angles the latitude and longitude and an elevation above this point.

2.2.1.1 Geographic Datum

The mathematical definition of the coordinate system of the surface of the Earth is called *Geographic Datum*.

The true surface of our planet is very complex and it is changing by time, just consider volcano eruptions, landslides or floods. The simplest model, known from high school, is the sphere, however it is not accurate. Isaac Newton proved that the surface of a rotating fluid mass is a spheroid, also called *ellipsoid*, due to the gravity and the centrifugal force. An ellipsoid can be described by two parameters the minor axis and the flattening $1/f$ which makes it a practical formulation, therefore it is widely used.

The Earth is not fluid, therefore its true shape is more complex than an ellipsoid. The *Geoid* defines the surface of the Earth as a contour surface of the gravity potential energy (gravity + centrifugal force) where the normal of the surface is opposite to the gravity vector. This is identical to the sea level without the effect of tides and winds. This surface definition needs numerous reference measurements over the whole planet. Calculating with coordinates respect to the Geoid needs access to the reference measurements and is only applicable where this is feasible.

Earlier the datum were defined to be precise on a local scale, e.g. a country. With the emergence of Global Navigation Satellite Systems (GNSSs) (e.g. the GPS, Galileo, Glonass, BeiDou) it became important to define a Datum which targets the whole planet. The most widespread Datum today is the World Geodetic System 1984 (WGS84) also used by GPS. This positions the center of the ellipsoid to the mass center of the Earth as the satellites orbit around this point. The WGS84 defines the earth as an ellipsoid with a major (equatorial) radius a = $6378137m$ at the equator and flattening $f = 1/298.257223563$. Position measuring devices (i.e. GPS, GPS+IMU) provide mostly the WGS84 geographic coordinates.

2.2.1.2 Projections

To visualize the surface of the Earth we project it to a developable surface[8] which can be visualized later as a plane (i.e. a paper or a screen) without distortions. Every projection preserves some features (e.g. area, direction, distance) but distort others on the scale of the whole globe.

The most widely used projection system for aerial images is the Universal Transverse Mercator (UTM), a collection of projections. The Earth is projected to cylinders transverse to the equator distributed in 6 degrees zones around the world. This projection is close to a metric Euclidean space within a given zone. It can be used with the camera models used in computer vision and photogrammetry.

For more information about map projections the reader is referred to Snyder and Voxland (1989).

2.2.2 Aerial image orthorectification

To create accurate, orthorectified aerial images which can be overlaid with maps special systems are employed designed for this purpose. They consist of a calibrated camera and a GPS fused with an IMU. This combines the short-term precise position and orientation measurements of the IMU with the GPS position measurement which does not have drift and keeps its accuracy on the long-term. This provides both the internal and the external camera parameters, thus a pixel location in the image can be assigned to a line in the absolute world frame.

For recovering the full 3D world position of a pixel it is projected on a Digital Surface Model (DSM). The DSM can either be already given (e.g. the one provided by the Shuttle Radar Topography Mission (SRTM)) or it can be calculated from the image sequence by dense stereo matching. For accurate results the camera parameters and the 3D points are optimized by bundle adjustment (Hartley and Zisserman, 2004). The absolute spatial accuracy is often enhanced by using Ground Control Points (GCPs). These are reference points with accurate world position which can be matched to points in the image. If there are no GPS+IMU measurements assigned to the images, the orthorectification is still possible by recovering the camera parameters from the images and defining the world coordinates by GCPs. However this needs longer manual work. For more information about the orthorectification of aerial images the reader is referred to (Kraus et al., 2007).

As the camera is relatively distant from the ground ($> 100m$) the absolute position of a point is very sensitive to the orientation of the camera. E.g. if the camera is $1000m$ above ground an angle error of $0.5°$ can cause $8.7m$ error in the horizontal position. To measure the orientation with high accuracy and without drift sophisticated, expensive and heavy IMU

[8] A surface which can be flattened without distortions

units[9] are needed. This limits the applicability, particularly for UAVs which are intended to be a low cost alternative to manned aerial vehicles.

2.2.3 Image based localization

Additionally to using extra devices to measure the location of images, the information in the image can be exploited for localization too. A typical application is to optimize the camera parameters and feature points by bundle adjustment during 3D reconstruction. This is necessary in most of the cases because even with very precise IMUs their might be a few pixel misalignment caused by noise in the synchronization. In certain cases it is also possible to localize an image over a given search area exclusively on the visual information and a preexisting knowledge (e.g. a collection of geotagged images, a map over an area). There are scenes which are inherently ambiguous (e.g. the trees of a forest, a hotel room, etc.), while other scenes contain landmarks being unique over the whole world (e.g. the Eiffel tower in Paris). Image based localization does not work in general, but it can still provide a good solution in many cases. The main idea of image based localization methods is to match the image with unknown location to data with known location. Based on the reference data, the methods can be categorized as *image to image* matching, *cross-view image* matching and *image to map* matching. The first covers the case when images with similar view point and resolution are matched. In the second case images of different perspective (e.g. ground to aerial) are matched, while the last category contains the employment of higher level info (e.g. a road map) as reference data instead of images.

2.2.3.1 Image to image matching

Image matching is applied very often when multiple images are processed, e.g. in 3D reconstruction or visual odometry. The standard pipeline is to extract feature points in the images, match these points and find a transformation which explains the correspondence between the points (Hartley and Zisserman, 2004). Since all the feature points in two images are matched, this step can have high computation demand. When applying this matching to a very high number of reference images (what is typical in localization tasks) it can take impractically long time. To address this problem global image descriptors can be applied instead of feature point descriptors e.g. (Hays and Efros, 2008; Weyand et al., 2016) or approximated feature point matching can be performed (Wu et al., 2008).

(Hays and Efros, 2008) is the first attempt for geolocalizing images worldwide. They create a database of geotagged images and match the query image to this database by calculating the nearest-neighbor in a feature space of various image cues (e.g. color, lines, texton histograms). This approach can achieve a localization 30 times better than by chance. A recent approach to the same problem is (Weyand et al., 2016). They consider the problem as classification instead of image matching and apply deep learning to efficiently integrate multiple visible cues. The classes are defined as geographical cells and the labeling for an image is the cell to which it belongs. In their experiments the automatic method could outperform even well-traveled humans in the world-wide image localization task. Fig. 14 shows an example of this methods.

(Wu et al., 2008) localize satellite images by matching them to geolocalized satellite images. Efficient matching is performed by approximate nearest neighbor search of Scale-Invariant Feature Transform (SIFT) features (Lowe, 2004) called visual words based image search. The SIFT feature vector is quantized to a single index (i.e. a visual word) by hierarchical k-means clustering. This reduces the nearest neighbor search to a look up and voting in the visual

[9] The IMU-m+SMU of IGI weights 2.8 kg and costs 80k Euro. `http://www.igi.eu/aerocontrol.html`

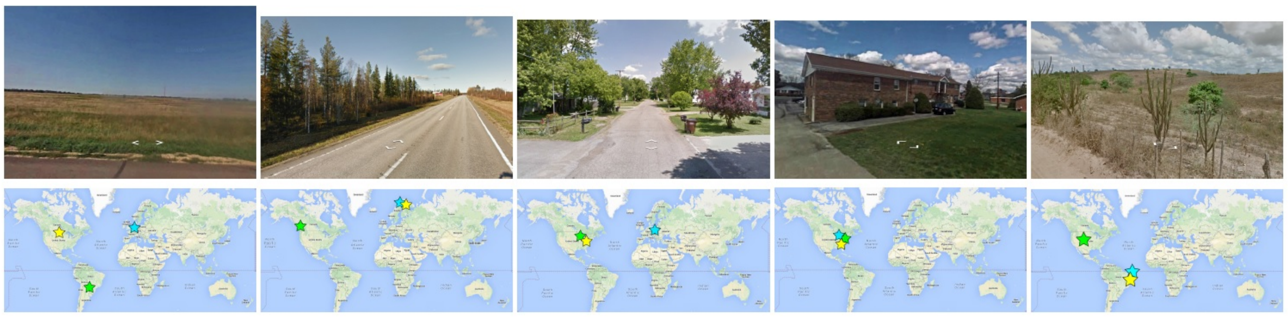


Fig. 14. From (Weyand et al., 2016). Worldwide image based localization. At the top the query images are shown. The bottom map shows the ground truth position (yellow), the human location guess (green) and in blue the automatic guess of (Weyand et al., 2016).

word indices and can enable much faster query in large image databases. As a final step the possible locations are verified by a 2D similarity transformation (scale, rotation, location).

2.2.3.2 Cross-view image matching

The previous methods targeted to match an image to an image of the same perspective (ground to ground, aerial to aerial). It is a more difficult task to match images acquired from different perspectives, but this approach can have benefits. This is typically the matching of a ground image to aerial images. The main advantage is that aerial images have a full coverage over a large area, while acquiring the same coverage with ground images would be much more laborious, if possible. Due to the different perspectives and resolutions, classical feature point based matching is not possible here. Instead, higher level abstract features are constructed from images with known correspondence. The high learning capability of deep learning can deliver state of the art also in this case.

(Lin et al., 2015) train a deep convolutional neural network to match ground images to oblique aerial views. The network takes two images as input and transforms them to lower dimensional feature vectors. The network is trained so that the Euclidean distance of these two vectors should be close if the two images are over the same location, and large if not. This allows to apply nearest neighbor search in the feature space of the deep network as localization. (Workman et al., 2015) use a similar approach but instead of using oblique view images they use aerial ortho images. They fine-tune existing deep networks for the task of cross-view image matching. They have created a huge dataset of ground and aerial images over the whole United States of America.

2.2.3.3 Image to map matching

In the case of image to map matching the features extracted from the images have to be matched to features in the map. This is advantageous since the maps applied are already available and storing them is typically easier than storing a huge database of images. Another advantage is that the map features are robust to visual appearance changes and do not depend on the sensor applied for image acquisition.

(Brubaker et al., 2013) localize a car from the visual odometry of the cameras mounted on the roof of the car and a road map over the region. The map is represented as a graph and the path of the vehicle in the graph is described by a probabilistic model. The position and heading of the car is predicted by a filtering algorithm exploiting the structure of the graph using a Mixtures of Gaussian model. This method can localize the car up to 3 m after driving a few seconds in an area containing more than 2000 km of roads. Fig. 15 shows results from the paper.

(Wang et al., 2015b) use the floor plan of a mall and a monocular image taken on the corridor of the mall for self-localization. The problem is posed as one of inference in a MRF jointly

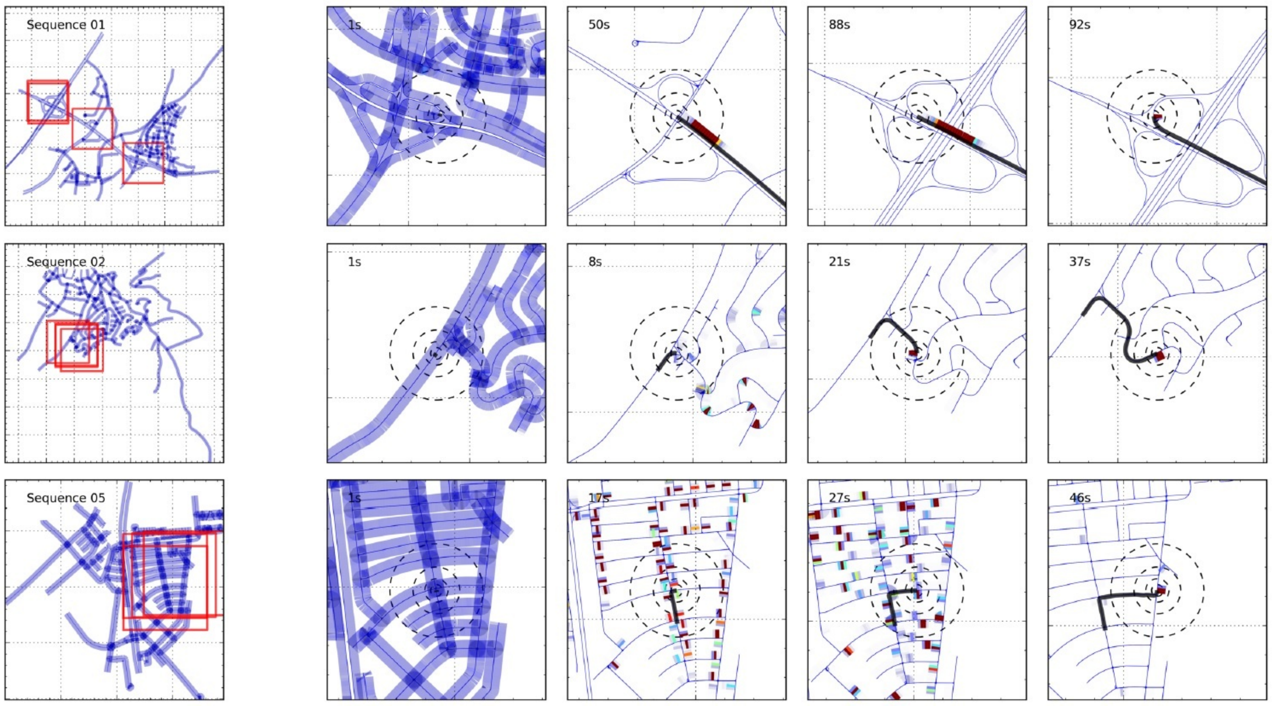


Fig. 15. From (Brubaker et al., 2013). Examples for localizing a car by visual odometry and a map. "The left most column shows the full map region for each sequence, followed by zoomed in sections of the map showing the posterior distribution over time. The black line is the GPS trajectory and the concentric circles indicate the current GPS position. Grid lines are every 500m. "

reasoning about the shop names, shop facade segmentation and the position and orientation of the camera. Since this method only utilizes map information (i.e. shop floor plan) it is robust to appearance variations, e.g. shop design, illumination.

In (Kozempel and Reulke, 2009) aerial images are orthorectified (projected on the Earth's surface) without an IMU. The orientation is initialized by an accurate GPS measurement and optimized by matching the detected streets in the image to the road network.

A more detailed description about the state of the art related to localizing aerial images can be found in chapter 4.1.

2.3 Detecting objects in images

In computer vision the term *object detection* is used for the task of marking individual instances of a specific object in an image, e.g. drawing a square around faces in an image. Due to implementation reasons, the objects are in the most cases marked by axis aligned rectangles (bounding boxes). The localization and counting of the vehicles (as described in chapter 3) needed for estimating the traffic in aerial images is also an object detection task. This section describes the basics of general object detection.

The most successful approach is to formulate the detection problem as one of image classification. Bounding boxes are proposed and each bounding box is classified as either object (of a specific class) or background. The number of possible bounding boxes is very high as both the position (top-left corner), the aspect ratio and the scale can change. For a fast and reliable object detector there are two main challenges. The first is to generate promising candidate bounding boxes without missing any. The second is to create a rapid and high performing classifier.

The first system to detect human faces reliably in real-time is the work of (Viola and Jones, 2001, 2004). They restrict the bounding boxes to squares and propose bounding boxes in a grid search manner densely sampling for both scale and position. This is referred to as *sliding window* and it is still applied by the state of the art methods. This dense search results in a very high number of bounding box candidates, therefore the success of this method is based on a very fast and robust binary image classifier. The binary image classifier uses Haar-like features which can be calculated very fast by integral images. The boosting algorithm used for classification selects the relevant features, so the rest does not have to be calculated at test time. The classification is organized in a cascade which can reject background already by only evaluating a few features. Since background usually dominates the images, this can result in a huge speedup.

The first cascade structure was rigid and therefore not optimal. (Bourdev and Brandt, 2005) proposed to use the weak classifiers of the boosting algorithm as stages. This enables more fine grained rejection stages and allows to reuse the already calculated weak classifiers.

Before the conquest of deep learning, hand-crafted features were used to capture the relevant information in the image while being invariant to the irrelevant. The most widely used hand-crafted feature for object detection is the Histogram of Oriented Gradients (HOG) (Dalal and Triggs, 2005). Applied together with a SVM classifier it was the state of the art for a long time in object detection. It was particularly successful by applying it in a Deformable Part Models (DPMs) (Felzenszwalb et al., 2010). (Dollár et al., 2014) extended the concept of Haar features to features over general channels, e.g. approximations of the HOG features. These can be calculated very fast, also on different scales by feature pyramids.

In the last years the automatic internal features learned by deep neural networks outperformed hand-crafted features also in the detection task, not just the classification. Since deep neural networks are very computation demanding, a fast detector needs good object bounding box candidates. (Girshick et al., 2014) address this problem by using a region proposal method, while (Ren et al., 2015b) apply a small neural network sharing its features with the more complex classifier network. The idea of using a cascade for efficient computation can also be extended to the new deep learning frameworks. (Cai et al., 2015) create a complexity aware cascade which uses simple features in the early stages and complex deep learning based classifiers in the later ones.

An alternative to the sliding window is the Effective Subwindow Search of (Lampert et al., 2009). This can find the candidates in a very efficient way by using a branch and bound algorithm over the range of possible bounding boxes.

Fig. 16 shows the output of a state of the art object detector.

2.4 Semantic image segmentation

Object detection marks the individual objects by a predefined shape (e.g. bounding box). However, many objects are much more complex and the reduction to a rigid shape can lead to abandoning important information. In many image analysis tasks it is necessary to extract information for each pixel. Classifying each pixel of an image is called *semantic segmentation*. Augmenting road maps with width and lane layout from images (objective of this thesis) can be considered a semantic segmentation task, since pixelwise accurate object information is extracted.

There are two important aspects of semantic segmentation. First, dense (pixelwise) prediction is needed in an efficient way. Second, the output variables are correlated (e.g. the pixels belong to the same object) and an effective method should incorporate this.

The most common way to express the conditional dependency of the output variables is to apply MRFs (CRFs). The shapes of the objects can be arbitrary, e.g. containing thin struc-

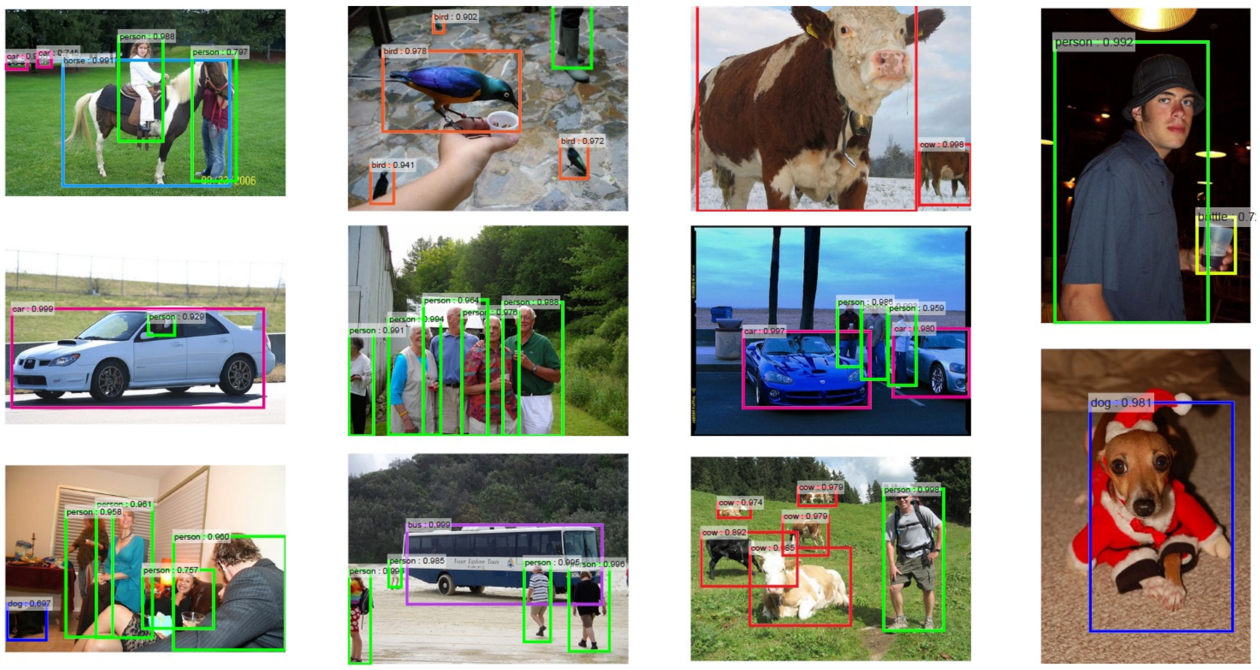


Fig. 16. From (Ren et al., 2015a). Object detection outputs of the method. The bounding boxes of different object classes are shown in different colors, with the class name and a confidence score.

tures. The grid structured pairwise potentials of simple MRFs tend to oversmooth and remove these. Thin structures can be preserved by *Fully Connected CRFs* defining pairwise potentials between each variable (pixel)(Krähenbühl and Koltun, 2011). Unfortunately, a Fully Connected CRF of N pixels (variables) has $N(N-1)/2$ pairwise dependencies and standard inference algorithms would take impractically long. (Krähenbühl and Koltun, 2011) address this problem by formulating the mean-field inference in the Fully Connected CRF as convolutions in the feature space. This can be solved efficiently, magnitudes faster than standard methods. This highly efficient inference enables the application of Fully Connected CRFs for the semantic segmentation of images.

The best image classifiers are deep Convolutional Neural Networks (CNNs), e.g. (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014), containing fully connected layers at the end of the network. These networks predict a single output variable for an input with predefined size, since a fully connected layer needs predefined input and output dimensions. Applying the classifier networks in a sliding window manner for each pixel would need too many computations, since the overlapping regions would be processed multiple times. Fortunately, these networks can be transformed to *fully convolutional neural networks* (CNNs containing only convolutional layers) without retraining the learned weights. The key is that a fully connected layer with N_i input and N_o output variables is identical to a convolutional layer with 1×1 kernel of N_i input and N_o output channels, but the convolutional layer is a digital filter computing an output vector for arbitrary input sizes. A fully convolutional neural network functions as a filter and overlapped regions are calculated only once. If the network contains layers performing downsampling, e.g. a max pooling layer with striding, then the output is also downsampled. As instance, the fully convolutional version of the VGG network (Simonyan and Zisserman, 2014) yields a class probability map with the size 1/32 of the input image because each of the 7 max pooling layers perform a downsampling by 2 due to the striding.

(Chen et al., 2015) apply the fully convolutional version of the VGG to produce a coarse pixel class probablity map, these values are then interpolated to the original resolution and feed into a fully connected CRF (Krähenbühl and Koltun, 2011) to infer the pixelwise labeling. To keep both high output resolution (small downsampling) and large receptive field size, dilated convolutions are applied in the last layers. The trainable parameters of the neural network and the CRF are trained separately. Fig. 17 shows outputs of this method.

(Schwing and Urtasun, 2015) and (Zheng et al., 2015) apply the same CNN and CRF as (Chen et al., 2015), but they formulate the iterations of the inference in the CRF as a Recurrent Neural Network and propagate the gradients back through this. This allows to learn the deep and the structured trainable parameters jointly. (Yu and Koltun, 2015) propose to remove pooling layers and zero padding and apply a context network to enlarge the receptive field of the deep network. This modifications increase the performance significantly, suggesting that segmentation networks should preserve more translation variant features than classification networks and larger receptive field is beneficial.

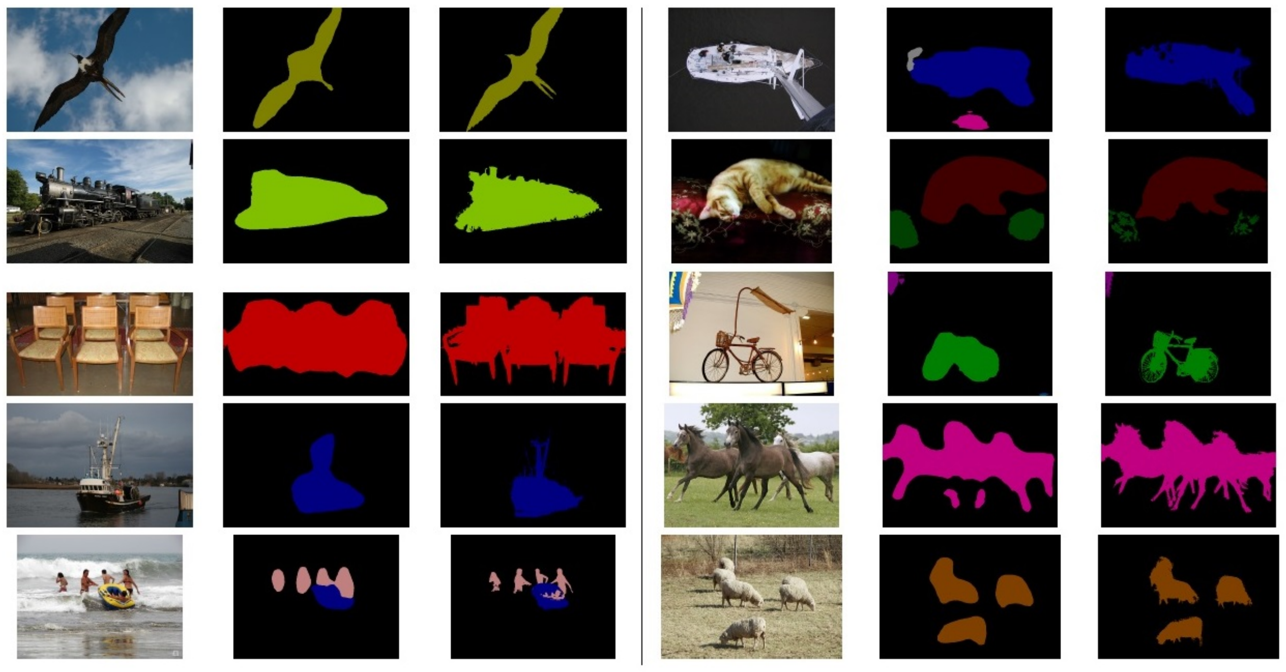


Fig. 17. From (Chen et al., 2015). In each column the left image is the input, the middle image shows the labels produced by the fully convolutional neural network and the right image shows the output of the fully connected CRF. Different colors represent different classes. Note that the coarse output of the deep network if refined by the CRF.

3 Fast multiclass vehicle detection on aerial images

This chapter includes the paper Kang Liu and Gellert Mattyus: *Fast multiclass vehicle detection on aerial images* (Liu and Mattyus, 2015)[10] . The original version of the paper can be found in the appendix A.

The road maps typically contain only the static properties of the road. However, the utilization and traffic flow estimation is very important for infrastructure planning, live navigation and the management of the traffic. Various methods are applied for extracting the road/parking utilization and the traffic flow. The number of the cars passing at a road section can be measured by manual counting, induction loops, surveillance cameras or other sensors/systems deployed at the road. The traffic flow can also be measured by tracking a fleet of vehicles (e.g. installing GPS and transmitters on taxis) and gathering this information. The locations of smartphones can also be used to estimate their motion and infer the traffic on the roads from this.

Aerial images can be an alternative to extract dynamic road information from the ground. Collecting traffic and parking data from an airborne platform gives fast coverage over a larger area and eliminates the need for on the spot measurements. Getting the same coverage by terrestrial sensors would need the deployment of more sensors, more manual work, thus higher costs.

A good example for an airborne road traffic measuring system is the one in the project *Vabene* Leitloff et al. (2014) of the German Aerospace Center (DLR). In this real-time system aerial images are captured over roads and the vehicles are detected and tracked across multiple consecutive frames. This gives a fast and comprehensive information of the traffic situation by providing the number of vehicles and their position and speed. Fig. 18 provides the overview of our vehicle detection work flow and illustration of the output. The detection is a challenging problem due to the small size of the vehicles (a car might be only 30×12 pixels) and the complex background of man-made objects which appear visually similar to the cars. Providing both the position and the orientation of the detected objects supports the tracking by giving constraints on the motion of the vehicles. This is particularly important in dense traffic scenes where the object assignment is more challenging. The utilization of roads and parking lots depends also on the type of the vehicle (e.g. a truck impacts the traffic flow different as a personal car). A system having access to this richer information can manage the infrastructure better. In a real-time system as in Leitloff et al. (2014) the processing time (and computing power) is limited. Therefore the processing method should be as fast as possible.

Our vehicle detection method provides both robust performance, fast speed and vehicle orientation and type information fully automatically based only on the input image. We detect the bounding box of the vehicles by a very fast binary sliding window detector using Integral Channel Features and an AdaBoost classifier in Soft Cascade structure. The bounding boxes are further classified to different orientations and vehicle type based on HOG features Dalal and Triggs (2005). We test and evaluate our method on a challenging dataset over the city Munich, Germany and another dataset collected by a UAV. These datasets contain original, non-orthorectified frame images which makes the problem more challenging since the exact Ground Sampling Distance (GSD) is unknown (we have only an approximate prior).

The main contributions of the presented method are: (i) The presented method uses features which can be calculated rapidly in a Soft Cascade structure. This makes the detection very

[10]This paper received the *IEEE Geoscience and Remote Sensing Society 2016 Letters Prize Paper Award.*

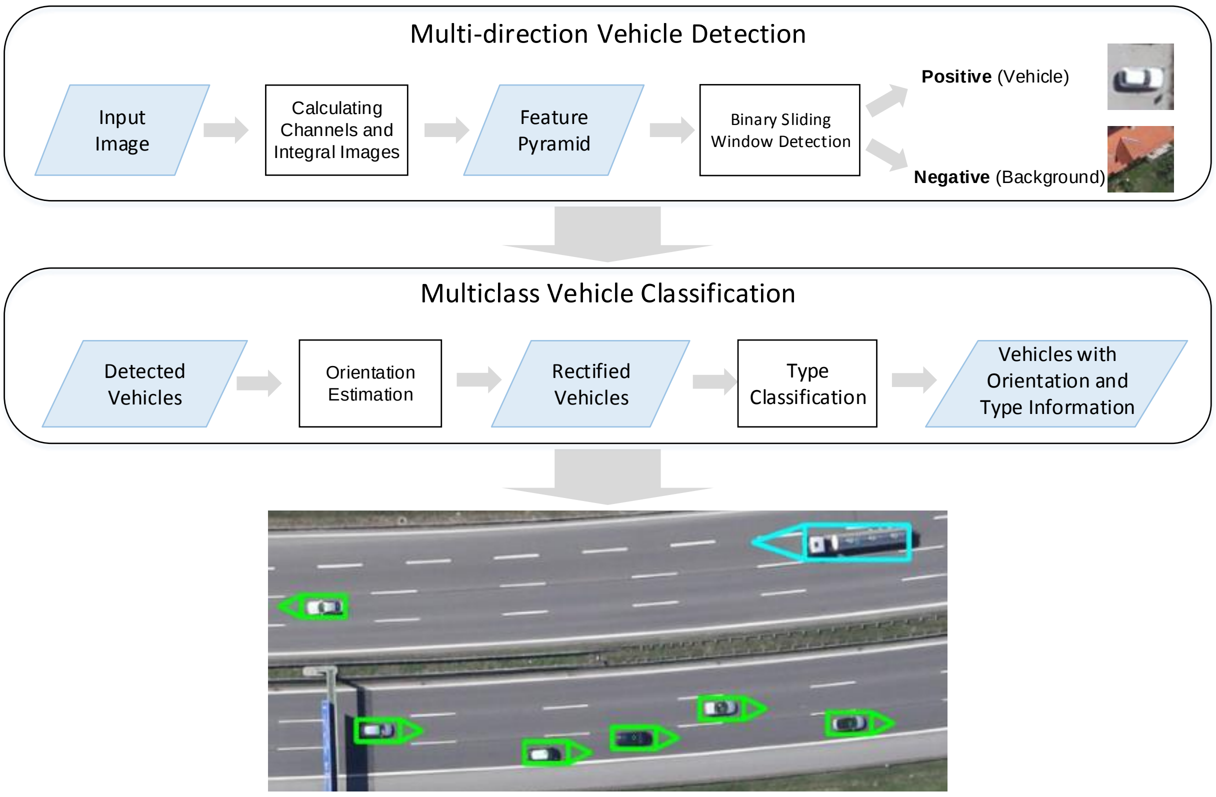

Multi-direction Vehicle Detection
Input Image
Calculating Channels and Integral Images
Feature Pyramid
Binary Sliding Window Detection
Positive (Vehicle)
Negative (Background)
Multiclass Vehicle Classification
Detected Vehicles
Orientation Estimation
Rectified Vehicles
Type Classification
Vehicles with Orientation and Type Information

Fig. 18. Proposed vehicle detection framework. The input image is first evaluated by the multi-direction vehicle detector. A sliding window goes along x- and y-axes. Features are extracted from the detection window and sent to trained binary classifier. The binary classifier classify whether current detection window contains a positive object or not. Detected vehicles are then processed for estimating their orientations and categories.

fast, it takes only a few seconds on a 21 MPixel image on a laptop single threaded. (ii) Our method also works on a single original frame image without any georeferencing, exact GSD, street database or 3D image information. (iii) Beside the location we also estimate the orientation and type of the vehicles.

3.1 Related Work

The problem of car detection in remote sensing images (aerial or satellite) were addressed by both the computer vision and the remote sensing community.

An important work in the computer vision community is (Kluckner et al., 2007). They apply an online version of the boosting algorithm on Haar-like features, local binary patterns and orientation histograms. They train the detector for cars in one direction and during testing they rotate the image in 15 degrees step. This detector is trained for a known object size 35×70 pixels and tested on images with the same scale. For minimizing the needed manual labeling effort false-positive samples are extracted by examining the height information of the images. The assumption is that true cars in a city are located mainly on the same height level (i.e. the street level). Our work differs by applying a soft-cascade and integrated channel features for faster speed, performing multiscale detection and predicting both the orientation and the type of the vehicle.

(Liu et al., 2014) define rotation invariant HOG descriptors by Fourier analysis in polar and spherical coordinates. The rotation invariant HOG descriptors are then used as features in a sliding window with a linear SVM or a random forest classifier. The cars in remote sensing images are rotation invariant, thus using rotation invariant features is well suited for the car detection problem. However the computation of these features can be computationally expensive[11] and by using simpler but more features similar results can be achieved on a comparable or even faster speed.

(Leitloff et al., 2010) detect the vehicles on satellite images which have a resolution of 60 cm per pixel which makes this task very challenging. They perform three steps. First they limit the Region of Interest (ROI) based on GIS data concentrating only on street areas. Second, Haar-like features are used in an AdaBoost classifier for generating detection hypotheses. As last, the candidates are grouped into vehicle queues and outliers are removed. This approach leverages priors given by the roads, which is necessary due to the low resolution. In aerial images the higher resolution enables the detection of vehicles around the whole images, and as our work shows, this can be achieved in reasonable computation time even on high resolution images.

(Tuermer et al., 2013) utilize the road map and stereo matching to limit the search area to roads and exclude buildings. HOG features with an AdaBoost classifier are applied to detect the cars on the selected region. This method is limited to georeferenced image pairs and areas covered by the road database.

(Leitloff et al., 2014) use a two stage approach for the detection of cars: first an AdaBoost classifier with Haar-like features and then an SVM on various geometric and radiometric features. They use the road database as a prior to detect only along the roads in a certain direction. The method achieves good results running fast on a Central Processing Unit (CPU), however it is limited to orthorectified images and areas covered by the road database.

(Moranduzzo and Melgani, 2014a,b) process very high resolution (2 cm GSD) UAV images for car detection. In (Moranduzzo and Melgani, 2014a) a feature point detector and SVM classification of SIFT descriptors is applied. To eliminate multiple counting of the same car the nearby detections are merged. These is an alternatives to sliding-window based approaches, however the results indicate that they perform poorer. The method in (Moranduzzo and Melgani, 2014b) uses HOG features with a SVM classifier in a sliding-window manner and later an orientation estimation. It is performing better than (Moranduzzo and Melgani, 2014a), but as our results, show it can be outperformed by our method.

In (Chen et al., 2014) the cars are detected by a deep neural network running on the Graphics Processing Unit (GPU) in a sliding window approach on a known constant scale. Neural networks deliver good results in vision tasks, also shown by this paper. However, the high computation demand, typically needing a GPU, puts limits on the applicability. Our method can deliver good results by running only on a single CPU thread.

3.2 Multi-direction Vehicle Detection

We handle the vehicle detection problem in two stages. The first stage is a very fast binary sliding window object detector which delivers axis aligned bounding boxes of the vehicles without type or orientation information. The second stage is a multiclass classifier applied on the bounding boxes estimating the orientation and the type of the vehicles. The processing steps are shown in Fig. 18.

3.2.1 Binary sliding window detector

For fast detection both the feature calculation and the classification has to be efficient.

[11] The implementation of the authors takes 18 seconds on a 792×636 image

3.2.1.1 Fast image features

Viola and Jones Viola and Jones (2001) introduced the integral image concept with Haar-like features for fast and robust face detection. By using the integral image I_Σ the pixel intensity I sum of the Haar-like features is calculated by a few operations independent of the area of the feature. The value $I_\Sigma(x,y)$ at (x,y) location in an integral image is the sum of the pixels above and to the left of (x,y):

$$I_\Sigma(x,y) = \sum_{i=0}^{i\le x}\sum_{j=0}^{j\le y} I(i,j) \tag{45}$$

The integral f_I within an axis aligned rectangle defined by its upper left corner x_0, y_0, width w and height h is calculated as $f_I = I_\Sigma(x_0+w, y_0+h) + I_\Sigma(x_0, y_0) - I_\Sigma(x_0+w, y_0) - I_\Sigma(x_0, y_0+h)$.

This idea is generalized by the Integral Channel Features (ICFs) in the work of Dollar et al. (Dollár et al., 2009). Instead of working on pixel intensity values as in (Viola and Jones, 2001), an ICF can be constructed on top of an arbitrary feature channel (i.e. the transformation of the original image). Features are defined as linear combinations of sums over local rectangular regions in the channels. By using the concept of integral images, an integral channel can be pre-computed for each feature channel so that the computation of the sum over the rectangle is very fast. The most commonly used channels are the color intensities, the gradient magnitude and the gradient histogram. The gradient histogram is a weighted histogram where the bin is determined by the gradient orientation. It is given by $Q_\Theta(x,y) = G(x,y)\mathbf{1}[\Theta(x,y)=\theta]$, where $G(x,y)$ is the gradient magnitude and $\Theta(x,y)$ is the quantized gradient orientation at x,y image location. The gradient histogram can approximate the powerful and widely used HOG features (Dalal and Triggs, 2005). If the rectangles are defined as squares, the sum can be aggregated to a single pixel in a downsampled image. In this case the integral is calculated even faster as a single pixel look up. This method is also called Aggregated Channel Features (ACFs) (Dollár et al., 2014). For rapid speed we apply this method with fast feature pyramid calculation as described in (Dollár et al., 2014).

3.2.1.2 AdaBoost classifier in Soft Cascade structure

The number of ICFs is very large (larger as the number of pixels in the image window) since it is the linear combination of local rectangular regions in the image window. We select only relevant features by the Discrete AdaBoost algorithm (Friedman et al., 1998) for N weak classifiers $h_t(\mathbf{x})$. $h_t(\mathbf{x})$ is a simple classifier, e.g. a threshold or a shallow decision tree of a few features from the input feature vector $\mathbf{x}$. AdaBoost is an iterative algorithm, in each step it reweights the samples in the training set according to the classification result from the previous weak classifier. The final strong classifier H is composed of the weighted α_t weak classifiers $h_t(\mathbf{x})$.

$$H = \operatorname{sgn}\sum_{t=1}^{N}\alpha_t h_t(\mathbf{x}) \tag{46}$$

At numerous sliding window positions (e.g. homogeneous regions) not all the weak classifiers have to be evaluated to classify the image as non-vehicle. To leverage this property for speed improvement we form a Soft Cascade (Bourdev and Brandt, 2005) from the weak classifiers. During the training a threshold r_t is set for all the weighted weak classifiers $c_t = \alpha_t h_t(\mathbf{x})$. If the cumulative sum $H_t(\mathbf{x}) = \sum_{i=1,\ldots,t} c_i(\mathbf{x})$ of the first t output functions is $H_t(\mathbf{x}) \ge r_t$, then input sample is passed to the subsequent evaluation process; otherwise it is classified as negative and rejected immediately.

3.2.1.3 Multi-direction detection

The orientation of the vehicles in aerial images can be arbitrary. This increases the intra-class variation of the appearance in the axis aligned sliding windows. A straightforward but computationally expensive solution, used in (Kluckner et al., 2007), is to train the detector for one specific direction and rotate the input image and do detection for each rotation. This would need the computation of the integral images separately for each direction and would result in slow processing speed. To overcome this we propose two methods: One is to train a single classifier which is able to detect differently oriented vehicles; The other is to aggregate several simple classifiers, where each is only sensitive to specific directions.

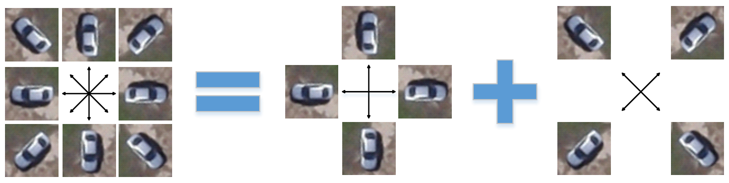
3.2.1.4 Single classifier method

A single binary classifier is trained with samples covering all the directions. The training process has to deal with the high intra-class variety and find the common part of all the positive samples. When the detector is applied on the input image, vehicles in any directions can be classified as positive samples.

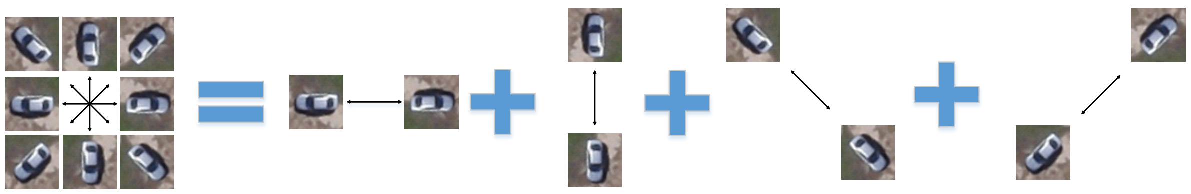
3.2.1.5 Aggregated classifier method

Alternatively the intra-class variety is reduced by splitting the training to different orientations. Multiple binary classifiers are trained, each for specific vehicle orientations. These classifiers are employed in sequence during the detection phase, and the results from each classifier are aggregated using non-maximal suppression. The integral image does not need to be calculated multiple times, only the classification. Fig. 19 illustrates how the training samples are grouped for training the aggregated classifier.

The performances of these two methods are examined in Section 3.4.



(a) Samples are split into two groups with rotation step of 90° in each group.



(b) Samples are split into four groups with rotation step of 180° in each group.

Fig. 19. The training samples for the aggregated classifier method are split into multiple groups.

3.3 Multiclass Vehicle Classification

The detector provides the axis aligned bounding boxes of the vehicles. In this next step we refine the extracted information by classifying the orientation and the type of the vehicle. We propose a two-step approach containing an orientation estimator and a type classifier.

A sample is sent to the orientation estimator first, then rotated to horizontal direction according to the orientation estimation, and finally processed by the type classifier to identify which type category this vehicle belongs to.

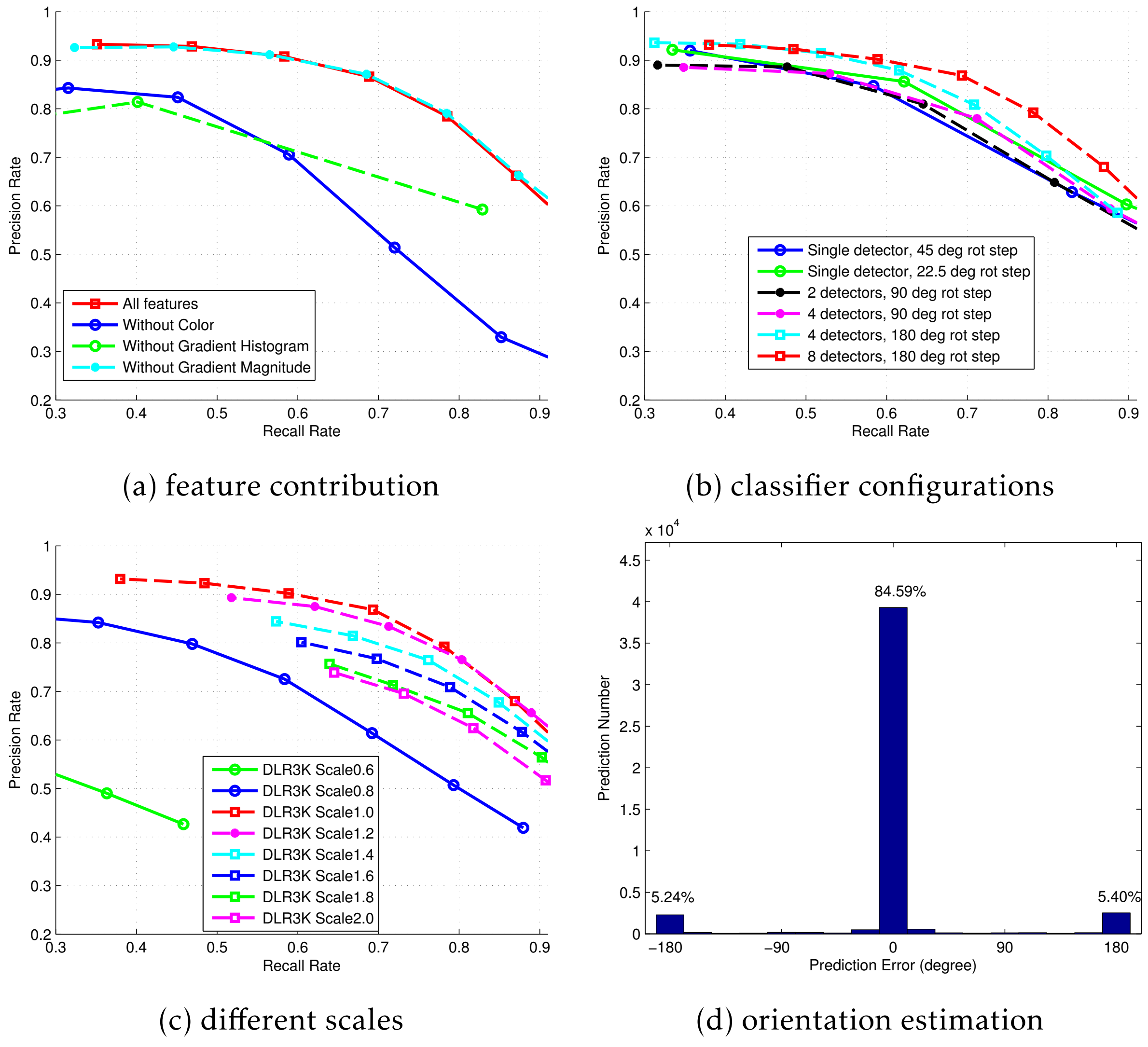


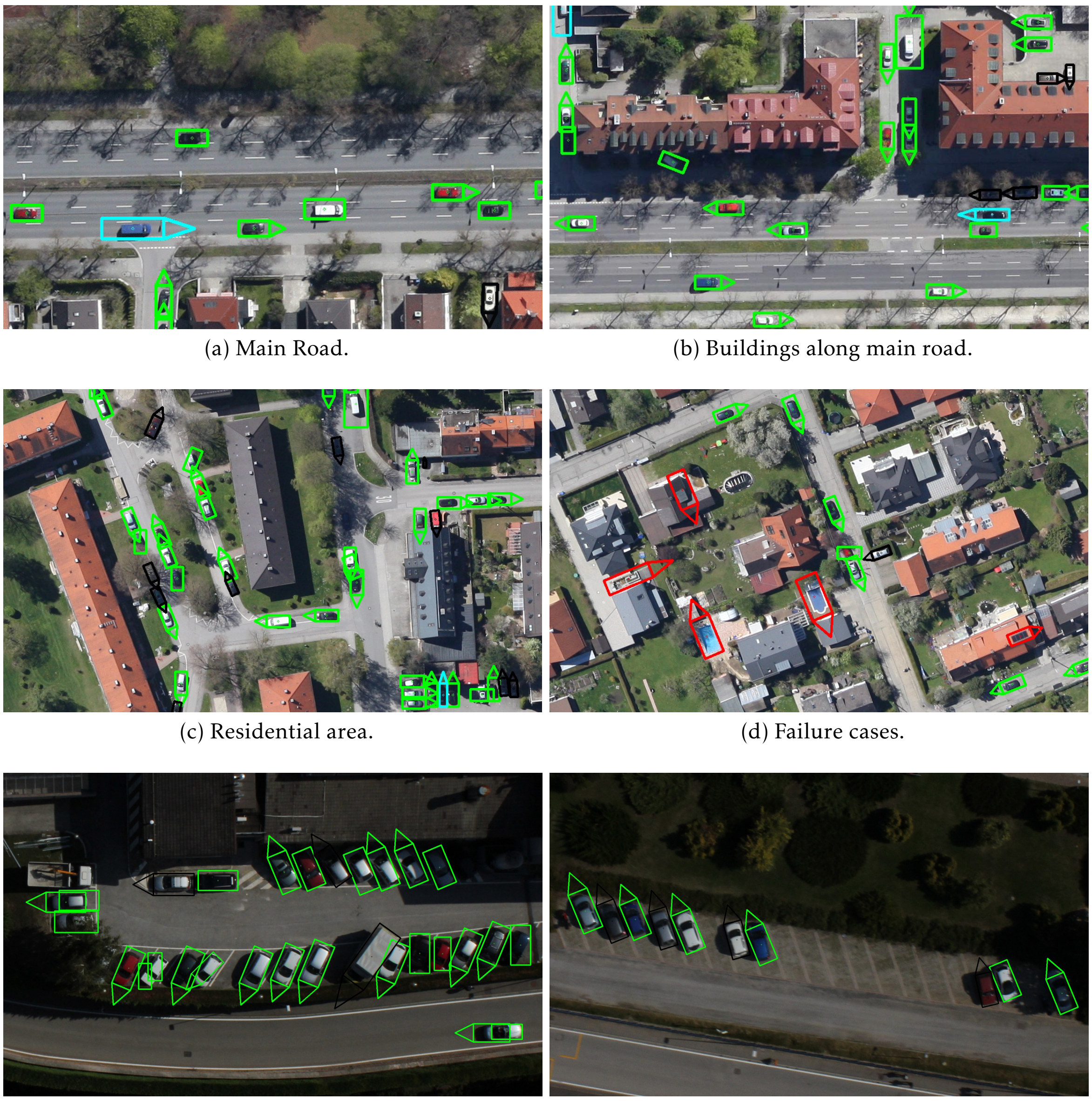
Fig. 20. (a) Evaluation of the Integral Channel Features. Gradient histogram channels play the most important role while gradient magnitude channel has least affects on the final result. (b) Detection result of aggregated detectors. (c) Performance after rescaling the image with different factors. (d) Orientation estimation error histogram using artificial neural network with 16 output classes.

3.3.1 Orientation estimation

We consider the orientation estimation as a multi-class classification problem. The directions are clustered, each cluster is considered as a class. The ICFs can be calculated fast, but they have a very high number, thus they are not suitable for multiclass classifiers working on a fixed length feature vectors. Therefore we apply the powerful Histogram of Oriented Gradients (HOG) features (Dalal and Triggs, 2005) which has a fixed feature vector length. We use a neural network with one hidden layer as a multi-class classifier (Lecun et al., 1998).

3.3.2 Type classification

The type classifier needs to classify the input image into corresponding categories. We have defined two type classes: car and truck but the presented method could be extended to more classes. The object bounding box is rotated to horizontal direction based on the orientation



(a) Main Road.

(b) Buildings along main road.

(c) Residential area.

(d) Failure cases.

(e) Detection on dataset in Moranduzzo and Melgani (2014a), Moranduzzo and Melgani (2014b)

(f) Detection on dataset in Moranduzzo and Melgani (2014a), Moranduzzo and Melgani (2014b)

Fig. 21. Detection results from the DLR test images. Green and cyan bounding boxes are the correct detected samples, representing cars and trucks, respectively. Black bounding boxes are the missed ones and red are the false positives. The results show that our method works well in most scenarios (a)(b)(c), however the complicated rooftops or outdoor swimming pools may lead to false positive detections (d). We also evaluated our method on the dataset presented in Moranduzzo and Melgani (2014a), Moranduzzo and Melgani (2014b), the detection results are shown in (e)(f).

estimation. Unrelated context is cropped out and HOG features are again extracted and classified by the type classifier.

3.4 Experiments

We test the multi-direction detection and multiclass classification parts in our detection method, respectively, and give quantitative results for the different processing stages. The binary detector is trained with 2048 weak classifiers in each test. We use depth-two decision trees as weak classifiers.

3.4.1 Results on Munich images

The quantitative evaluation is performed on 20 aerial images captured by the *DLR 3K* camera system (Leitloff et al., 2014) over the area of Munich, Germany. We use the original nadir images with the resolution of 5616×3744 pixels. They are taken at a height of 1000 meters above the ground, the approximate ground sampling distance is 13 cm. The first 10 images are used for training and the other 10 for testing. Positive training samples come from 3418 cars and 54 trucks annotated in the training images, while the negatives are randomly picked from the background, i.e. areas without vehicles. To overcome the low number of truck samples we randomly transformed them additionally 30 times. Fig. 21 shows detection results on the test images. We set the detection window to 48×48 pixels. For the ground truth the vehicles in the images are annotated manually as oriented bounding boxes.

3.4.1.1 Multi-direction vehicle detection

Integral Channel Features contain rich information and can be computed rapidly. They are selected as the features for training and detection. Experiments are performed to evaluate the importance of each feature channel type and the performance of different classifier configurations.

3.4.1.1.1 Feature channel

We use three types of feature channels: Luv color, gradient magnitude and gradient histogram. We have evaluated the contribution of each feature channel, the Precision-Recall (PR) curves are plotted on Fig. 20 (a). These curves indicate that gradient histogram channels play the most important role in representing the vehicles while the gradient magnitude channel affects the final result the least. For the later tests we use all the feature channels.

3.4.1.1.2 Multi-direction detection methods

We proposed two methods, single and aggregative classifiers, to detect vehicles in different directions (Section 3.2.1.3). The performances are depicted in Fig. 20(b). The PR curve shows that the optimal solution is the 'Classifier aggregation method' with each classifier trained using samples in opposite directions (8 detectors with sample rotation step of $180°$). This means 8 detectors and thus longer computation time. 2.7 s is needed for a single detector while the detection with 8 classifiers takes 4.1 s. This is sublinear since the integral images doesn't have to be calculated again. We use the 8-classifier configuration for the later tests.

3.4.1.1.3 Detection on images with different scales

To show the ability of our method to detect the cars on images with different scales we resized the image for the test but not the training. These results are shown on the Fig. 20(c). The detector performs best on the same scale as it was trained, if the resolution is increased the performance remains comparable. But if we decrease the resolution we lose information which leads to a lower performance.

3.4.1.2 Multi-class vehicle classification

After the axis-aligned bounding box detection we classify the orientation and type of the vehicles. We convert all the bounding boxes to 48×48 pixel gray images and calculate HOG features for this image. We get the best performance with 4×4 cell size, 1×1 block size, 1×1 block stride HOG feature configuration and use this for the later tests.

Method	Ground Truth	True Positive	False Positve	Recall Rate	Precision Rate
Munich dataset					
Viola-Jones	5892	3237	1467	54.9%	68.8%
Ours	5892	**4085**	**619**	**69.3%**	**86.8%**
UAV dataset					
Moranduzzo and Melgani (2014a)	119	88	143	73.95%	38.1%
Moranduzzo and Melgani (2014b)	119	87	111	73.1%	43.4%
Ours	119	**94**	**6**	**79.0%**	**94.0%**

Table 1. Performance comparison between different methods. The UAV dataset is from Moranduzzo and Melgani (2014a,b)

3.4.1.2.1 Orientation estimation

Orientation classification is performed according to Section 3.3.1 with 16 classes (22.5° rotation difference between adjacent sample groups, respectively). The orientation estimation error histogram is depicted in Fig. 20(d). The most common error is when the samples are classified in the opposite direction. This is because the front part of a vehicle might be similar to the rear part from the top view in aerial images.

3.4.1.2.2 Type classification

The detected bounding box is rotated to the horizontal direction according to the orientation estimation. We trim the input image by cropping the upper and lower parts, from 48×48 to 48×28. In our dataset the number of trucks is much less than the number of cars. We generate new ones from the existing samples using random transformation. The performances with different cropping configurations are compared in table 2 and the supplementary material. The optimal type classification can reach 98.2% in accuracy with a one-hidden-layer neural network.

Cropped Size	48×48			**48×28**		
Confusion Matrix	A/P [a]	Car	Truck	A/P	Car	Truck
	Car	2843	60	Car	**2838**	**65**
	Truck[b]	123	685	Truck	**0**	**808**
Accuracy	95.1%			**98.2%**		

Table 2. Confusion matrices of type classification using different cropping configurations. *a* : Actual class / Predicted class. *b* : The number of truck type is increased by random transformation of the existing samples.

3.4.1.3 Baseline comparison

As baseline we use the OpenCV[12] implementation of the Viola-Jones detector (Viola and Jones, 2001). We have trained it on one vehicle direction while at detection we rotate the image similar as in (Kluckner et al., 2007) and apply the detector for each rotated image. Table 1 contains the numerical comparison of this method on the Munich dataset.

[12]`http://opencv.org/`

3.4.1.4 Computation time

Since the processing time is also important for the detector we compare our method with other methods where the computation time is provided in the paper. Table 3 contains the computation times. Our experiments are performed on a laptop with Intel® Core™ i5 processor and 8 GB memory and our program is running single threaded written in Matlab and C++. The comparisons show that the speed our method is considerably faster. This makes our method more suitable for real-time systems where the computation time is a serious issue. The method of (Chen et al., 2014) achieves comparable detection performance but on a different dataset, therefore we show only the processing time of the method.

Method	**Image Resolution**	**Detection Time Per Image [s]**	**Detection Time Per MPixel [s]**
Proposed	5616×3744	**4.4**	**0.2**
Viola-Jones	5616×3744	1160	55.2
Moranduzzo and Melgani (2014b)	5184×3456	14400	803.8
Chen et al. (2014) *	1368×972	8	6.0

Table 3. Comparison of computation times. *Running on the GPU.

3.4.2 Baseline comparison on UAV images

We also evaluated our method on the dataset presented in (Moranduzzo and Melgani, 2014a), (Moranduzzo and Melgani, 2014b) and compared to the results provided without screening. The results can be found in the Table 1. The precision rate of our method outperforms the other methods significantly. Due to the higher resolution we set the detection window to 96×96 pixels for this dataset and have only car vehicle type (no truck).

4 Large scale aerial image sequence geolocalization with road traffic as invariant feature

This chapter describes the paper Gellert Mattyus and Friedrich Fraundofer: *Aerial image sequence geolocalization with road traffic as invariant feature*. The paper is currently under revision after a minor revision at the *Image and Vision Computing* journal. The submitted paper can be found in the Appendix B.

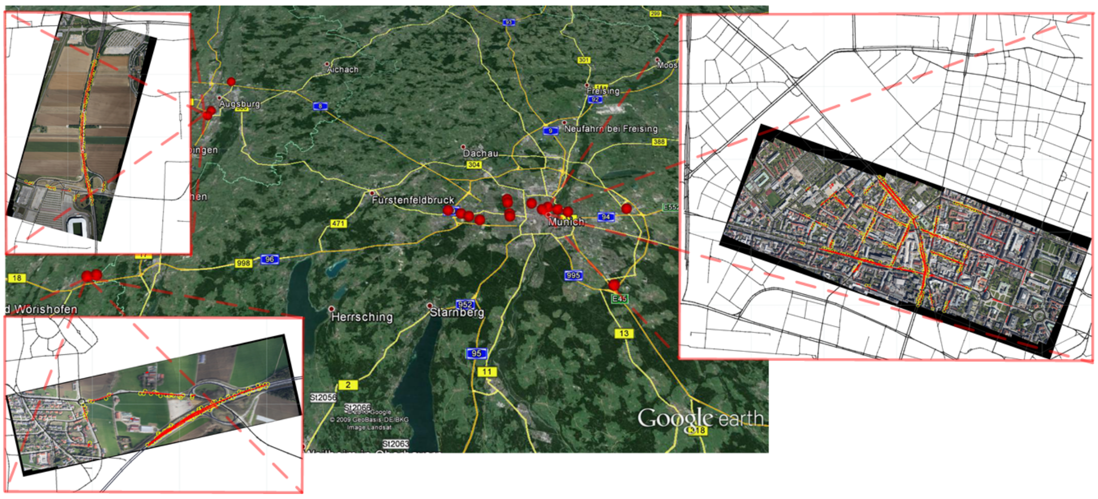


Fig. 22. Illustration. We search the geolocation of aerial image sequences. The red dots on the map show the ground truth positions. The photos can be located by matching the road traffic (in red) in the image to the road network (in black). The matching tracks are marked with yellow circles in the images. The search area can be as large the shown map. The accuracy of the location is around 25 m.

Our goal is the augmentation of aerial photo sequences with the geolocation information on a large scale by utilizing only the image information and the road network. We propose to match the information acquired from the current aerial image sequence to an object database invariant to the lighting and weather conditions, the road network map. Since the maintenance of an accurate, up-to-date road map is needed for many other applications (e.g. navigation, administration, etc.), it is more easily accessible than a large image database and requires significantly less storage [13] .

We detect the road traffic in the image scene by tracking cars over the frames. By assuming the cars drive on the roads the vehicle trajectories can be interpreted as subsets of the roads, and be matched to the road map.

We propose a method for the fast retrieval of a shortlist containing the possible correct location. This retrieval is based on Geometric Hashing. We call it *Polyline Based Geometric Hashing (PLBGHashing)*, and it can search rapidly over a larger search area. The more complex verification matching needs to be done only on the retrieved shortlist. The car tracks can be considered as a road detector with low completeness but high correctness. We combine this with a simple pixel color based road detector (with high completeness but low correctness) for the verification.

We analyze the proposed PLBGHashing on synthetic data generated from the road network of two large cities. This confirms that the pattern of the road network is discriminative on a larger scale. We can search an area of 22500 km^2 containing 32000 km of streets within minutes on a single CPU. An evaluation is done on 20 image sequences captured over urban, suburban, rural with motorway, and industrial scenes by a consumer-grade camera mounted on an airplane, the Figure 22 shows the location of the scenes. In this test the PLBGHashing

[13]The OpenStreetMap is an open, crowd-sourced map with good coverage and fast updates.

provided the shortlist, while the verification ranked the correct geolocation as the best match in most of the cases.

Our main contributions are: (1) We use car tracks to detect parts of the road network. This does not need a known GSD [14] as standard road detectors. (2) We utilize road networks as appearance invariant features to localize an aerial image sequence over a large area. This also avoids the need for an image database over the search area. (3) We present a geometric hashing method to match partial line segments to line structures. This does not need a complete road detector which detects relations between roads as a graph, e.g. intersections.

4.1 Related Work

The standard for creating georeferenced aerial images is applying an accurate GPS and IMU and/or measuring Ground Control Points (GCPs) manually as described in section 2.2.2. Geolocalization is also possible by using the image information combined with already georeferenced images or maps, this is described in section 2.2.3. An important work is (Brubaker et al., 2013) showing that a long track on the road network is characteristic enough to obtain the geolocation of the car by only using visual odometry and a road map over the area. Our method shares the idea of exploiting the pattern of the roads in the form of car tracks, but instead of a single long vehicle trajectory acquired from the ground we use many short ones extracted from the air.

The idea of improving the location information of the image is also applied for aerial and satellite images. Matching the image to a precisely geolocalized image is applied by (Müller et al., 2012). They improve the absolute geometric accuracy of satellite images by automatically extracting Ground Control Points from existing orthorectified reference images. In (Wu et al., 2008) satellite images are localized with feature-based indexing, but not on a real large scale (maximal 16 km×12 km). In (Lin and Medioni, 2007) UAV images are matched to georeferenced ortho images (map). But the search area is limited by manually labeling several correspondences between the first frame of the UAV sequences and their corresponding satellite images.

The registration of aerial images to road maps is investigated in various works. However, these methods does not address large scale problems and/or rely on features defined by road intersections which are not necessarily present in the scene or extractable from the image.

In (Kozempel and Reulke, 2009) aerial images are orthorectified (projected on the Earth's surface) without an IMU. The orientation is initialized by an accurate GPS measurement and optimized by matching the detected streets in the image to the road network. In comparison to this method, our approach does not require an accurate position information, but only a search area, which might be as large as an entire metropolitan area. This formulates a considerably different problem, where the solution space is far greater and the problem is not guaranteed to be convex. The presence of many possible local minima requires the search over many possible solutions in an efficient way.

In (Wang et al., 2007), road networks are represented as graphs, with road intersections as graph vertices. It is assumed that the intersections are detected correctly and the georegistration problem can be solved as a graph matching problem. In comparison to this work, in our approach the road intersections does not need to be detected in the image. We assume that there is no appropriate intersection detector or there might be no intersections in the image (just non intersecting roads).

[14]Ground Sample Distance, the distance between pixel centers measured on the ground.

In (Wilson and Hancock, 1993) the graph structure of the road network (junctions and roads between them) in aerial image is extracted using a relaxational line-finder. Then the graph junctions are matched to junctions in the map using probabilistic relaxation. This method assumes that road junctions are present and they can be detected. In contrast, our method works without junctions, it needs only segments of the road network and we consider a much larger search area.

(Li et al., 1992) extract and match line segments instead of junctions. They define rotation and translation invariant features between the segments and use these pairwise features to calculate the cost of an assignment between aerial image and map. This combinatorial optimization problem is solved via continuous relaxation labeling. Solving this optimization problem is nontrivial, especially for a large number of variables. To limit the number of possible matches, the orientation of the image is used to consider only matches with similar direction. However, this orientation information is not accessible in the general case which we address.

Matching line segments between images is investigated in (Gros et al., 1998). They use local, geometric invariant features to match images related by similarity or affine transformations. For similarity transformations they compute geometric invariants from pairs of line segments having an endpoint in common. The invariants are the angle between the two segments and the ratio of the length of the segments. These invariants are matched between the images, similarity transformations are computed from the matches and they are aggregated in a Hough-transform manner, i.e. clusters are searched in the parameter space. In contrast to this method, in our problem the segment length ratio invariant cannot be applied. A track segment can lay anywhere within the map segment. Additionally, our tracks are mostly straight (or with a small curvature), the angle between the consecutive line segments is usually around zero and thus the angle invariant is not discriminative enough.

4.2 Method

We extract the road traffic from the images and match the car tracks to the road network. Detecting the roads directly would be more straightforward, but this may pose problems due to the high intraclass variety of roads. The methods described in (Wegner et al., 2013), (Mnih and Hinton, 2012) show good results, but they work on images with a constant GSD and the effect of an unknown GSD is not investigated. In our case there is no information about the GSD. Using only the car tracks also highlights that already a fraction of the road network is enough for the geolocalization.

The intraclass variety in visual appearance of cars is much smaller than that of roads. Thus existing robust object detection methods (e.g. Viola-Jones object detection framework (Viola and Jones, 2004)) can be applied with satisfying results (e.g. a detector based on boosting is presented in (Kluckner et al., 2007)). The features of the car detector are pixel intensity differences, which can be robust against different weather and lighting conditions. The motion information is also a strong cue, enabling object detection in cases where a non-moving object could not be distinguished from the background. We utilize the motion information by using the tracks of only the moving cars to reduce the possible false positive car detections. The length of the tracked cars in the image also gives information about the GSD. The proposed geolocalization method could also work with a suitable road detector (e.g. based on deep learning), or by labeling the roads manually and limiting the GSD search space.

4.2.1 Track extraction

The track extraction works on a mosaic image compiled from the single images. We use the tool *VisualSFM* (Wu, 2013) to calculate the camera parameters and 3D point coordinates in the scene. The earth surface is assumed a plane, and a homography is calculated between the images from the camera parameters and the equation of the plane.

We detect the cars independently on the image frames. We apply the Viola-Jones object detection framework (Viola and Jones, 2004) with the Gentle AdaBoost (Friedman et al., 1998). The detector is rotation variant, thus we rotate the image in steps to cover all directions and group the independent detections together (it would be more time efficient to use a multi-view detector).

The detections are transformed to the mosaic image. During the tracking we model the motion of the cars with a simple linear model and use Kalman-filtering. In each frame the tracked objects are assigned to detections matching the predicted object positions. We discard the short tracks to avoid false positives. This is a simple method, delivering satisfying results.

4.2.2 Matching the tracks to the road network

We formulate the aerial image geolocalization as a model recognition and pose estimation task of the road network pattern. The search area is tiled to overlapping square areas (in object recognition terms a tile is a model). An image scene query returns a model and a transformation between the image and the model. Since the absolute world position of the tile (model) is known, the transformation from the image coordinates to the geocoordinates can be recovered. The recognition has to handle geometric transformations for the pose estimation, a high number of models (> 10000) for larger search areas, and partially occlusions since the extracted tracks are just a subset of the road network. The geometric hashing method is suitable for handling these challenges.

4.2.2.1 Geometric hashing

Geometric hashing is an efficient, low polynomial complexity technique for matching geometric features against a databases of such features. Matching is possible even when the recognizable database objects have undergone transformations or when only partial information is present. For a more detailed explanation the reader is referred to (Wolfson and Rigoutsos, 1997).

4.2.2.1.1 Map and track representation

The roads in the map are represented as 2D polylines. A polyline contains vertices (points), and the road consist of the line segments between consecutive vertices. The tracks are also polylines, whereas the vertices are the positions in the registered image during different frames. Although the representation of the tracks and roads is identical, there is no strict correspondence between the vertex positions in the road network and the tracks. A track segment can lie anywhere within the road segment.

4.2.2.1.2 Polyline Based Geometric Hashing

The PLBGHashing method compiles the polylines to the hash table. Since we assume the earth's surface to be a plane and the tracks projected on this plane, the perspective transformation between the tracks and the roads can be reduced to a scaling, rotation and translation. As a basis we use a segment in a form of one point and a direction instead of two or

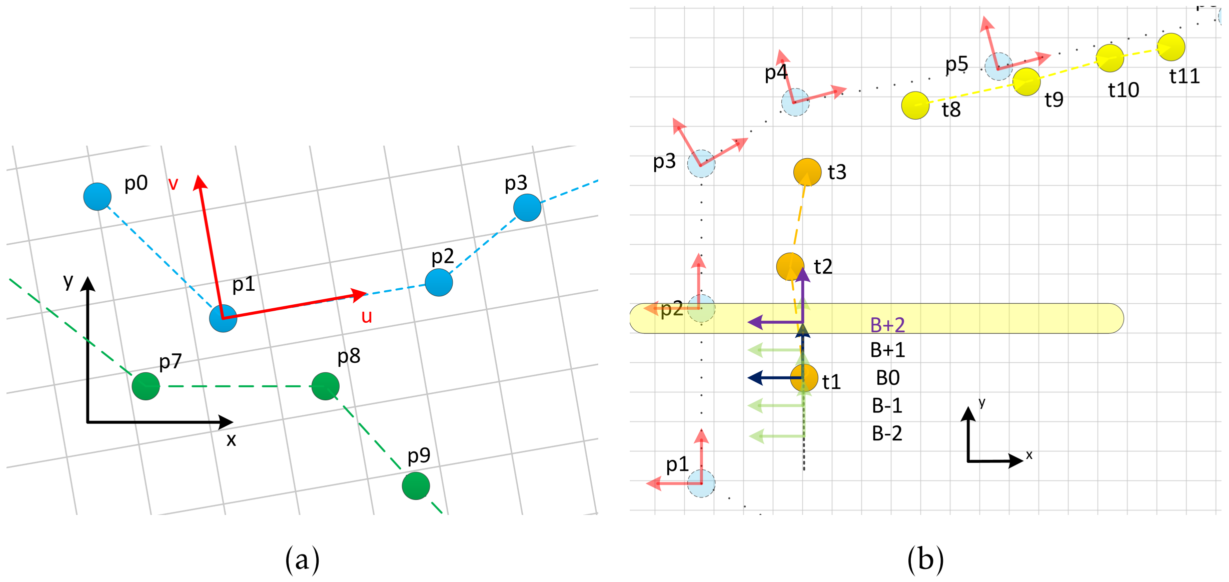


Fig. 23. (a): The blue and green polylines are different roads. The x-y is the original coordinate system of the road database. The line segment p1-p2 defines the basis (u,v) with the center point and the direction. The grid points intersected by line segments are the indices for the hash table. (b): The p points are the road vertices (from the database), the t points are the track vertices extracted in the mosaic image (with different color for different cars). The red arrows are the road bases in the hash table. The basis B0 from the track end points t1-t3 is shifted along the track. At B+2 position the track basis lies in the the same grid row, as the road basis.

more points. The relation of a **p** point's original coordinate (x, y) and the coordinate (u, v) in the basis defined by a line segment S with end points $\mathbf{p}_{s_1}, \mathbf{p}_{s_2}$ is described by the equation:

$$\mathbf{p} - \mathbf{p}_{s_1} = u\mathbf{p}_x^s + v\mathbf{p}_y^s \tag{47}$$

where $\mathbf{p}_x^s = \frac{\mathbf{p}_{s_1} - \mathbf{p}_{s_2}}{\|\mathbf{p}_{s_1} - \mathbf{p}_{s_2}\|}$ and $\mathbf{p}_y^s$ is $\mathbf{p}_x^s$ rotated by 90°. On this basis the 2D hash table is invariant to rotation and translation but not to the scaling. To find the correct GSD we query with multiple scales. By leveraging the low variation of the car lengths $(4 - 6m)$, the number of different scales are kept low. We calculate the maximum in the histogram of the car lengths (the bounding rectangle delivered by the detector) in the tracks. Utilizing this length we needed only 3 scales to search in our experiments to get good geolocalization.

Each hash bin stores a set of entries. An entry contains a model number and a basis. It is important to store a set with unique elements, otherwise some bins could be biased by areas containing roads densely.

4.2.2.1.3 Database construction (off-line)

The search area is tiled to squares with a fixed side length d_s and overlap. The nodes are interpolated to make the line segments shorter as a defined maximal value d_{max}. Then the hash table is generated by the following steps:

For each polyline (road) P_j in each tile M_i do:

(1) For each line segment S_k in the polyline P_j:
  (a) Create an entry E_k with the model number i and the basis of S_k.
  (b) Solve the (Eq. 47) with S_k to calculate (u_m, v_m) of all the vertex points in M_i.
  (c) Quantize the points (u_m, v_m) to a grid (u_m^q, v_m^q).
  (d) Calculate the coordinates (u_l^q, v_l^q) of the rasterized line in the grid between the neighboring $(u_{m-1}^q, v_{m-1}^q), (u_m^q, v_m^q)$ vertices. The (u_l^q, v_l^q) line points are indices for

the 2D hash table. For each line point (u_l^q, v_l^q) add the entry E_k to the set at the (u_l^q, v_l^q) hash bin.

The Figure 23a illustrates the hash index generation. The complexity of the number of bases is $O(n)$, where n is the number of node points after interpolation in the tile. The complexity of hash table entries per basis is $O(l/d_g)$ where l is the sum of road length and d_g is the grid size for geometric hashing. Since d_g is fixed in an implementation the complexity of hash entries is $O(nl)$.

4.2.2.1.4 Shortlist retrieval (on-line)

To find the correct geolocation of a scene, a basis (1-point and direction) in the tracks has to be aligned to a similar basis in the correct model with the correct scale. The direction of a road segment can be properly recovered from the end points of a track. We choose long straight tracks, because they define the direction more accurately if the vertex positions are noisy. The correct scale is searched with multiple queries. The list of scales is derived from the car length. Since the line segment length is limited by d_{max}, a street vertex exists in the range $[-d_{max}/2, d_{max}/2]$ along a track vertex, the Figure 23b illustrates this. It is enough to search this range in shifts by the grid size d_g, because then the hash indices are the same at creation and query. We resize the scene with all the search scales. For each resized scene we choose track segments T_i (the line segment between the end points). For each T_i in both directions we do:

(1) Shift T_i between $[-d_{max}/2, d_{max}/2]$ in d_g steps.
(2) Each shift gives a line segment S_k. For each S_k:
    (a) Solve the (Eq. 47) with S_k to calculate (u_m, v_m) of the vertex points of all the tracks in the scene.
    (b) For each polyline (track) P_i in the scene:
        (i) Calculate the rasterized points (u_m^q, v_m^q) of the vertices
        (ii) Calculate the coordinates (u_l^q, v_l^q) of the discrete (rasterized) line segment in the grid between the $(u_{m-1}^q, v_{m-1}^q), (u_m^q, v_m^q)$ points.
        (iii) Each point (u_l^q, v_l^q) of the line segment is an index for the hash table. For every entry at the hash bin (u_l^q, v_l^q) cast a vote for a recognition (model, basis and scale).

After the query with multiple straight tracks, the votes are aggregated. One recognition gets only one vote from a bin, otherwise a basis might be biased due to many tracks over the same index. A shortlist of matches is created by sorting the recognitions based on the number of received votes. The transformation from image to a world coordinate can be recovered from the basis in the image and in the model, the scale and the world position of the model (tile).

4.2.2.2 Verification of the shortlist

The retrieved shortlist also contains false recognitions. We re-sort the shortlist by a stricter criteria, the verification match score S. The quality of the match is measured by the mean distance d_{m_t} of the tracks R_t to the roads in the model, like the Chamfer Matching (Barrow et al., 1977). The distance image I_d is calculated to the roads (Borgefors, 1986), and the I_d pixels along the tracks are extracted. Since the tracks may contain outliers, we calculate a trimmed mean, where the 10% of the tracks with the highest distances are discarded. The d_{m_t} does not penalize roads not covered by tracks. To compensate for this, we extract a second road detector with high completeness but low correctness. We assume that the color of the roads is similar all over the scene. The CIELUV color values of the image pixels are grouped to 16 clusters by k-means. Clusters are added to the road class until 85% of the

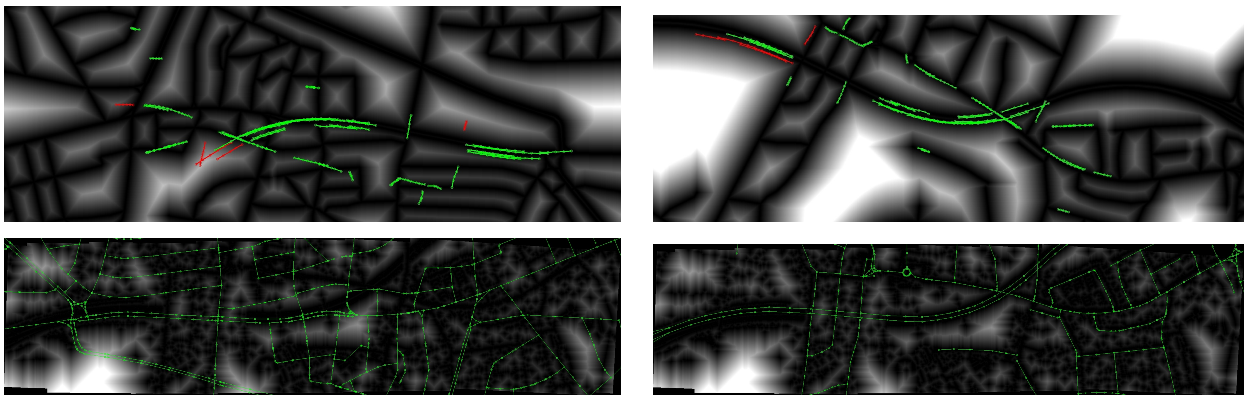

(Wrong localization) (Correct localization)

Fig. 24. The distance images used for calculating the verification score at *Suburban 1* image sequence. On top the tracks are shown with green (inlier) and red (outlier) in the distance image to the road network. In the bottom the road network (green) is shown in the distance image to classified road pixels in the image.

tracks lie over pixels belonging to the road class. A second d_{m_p} mean distance is calculated as the distance of the roads in the model to the roads detected by the pixel based detector (R_p). The Figure 24 shows the distance images, a brighter pixel is a larger distance.

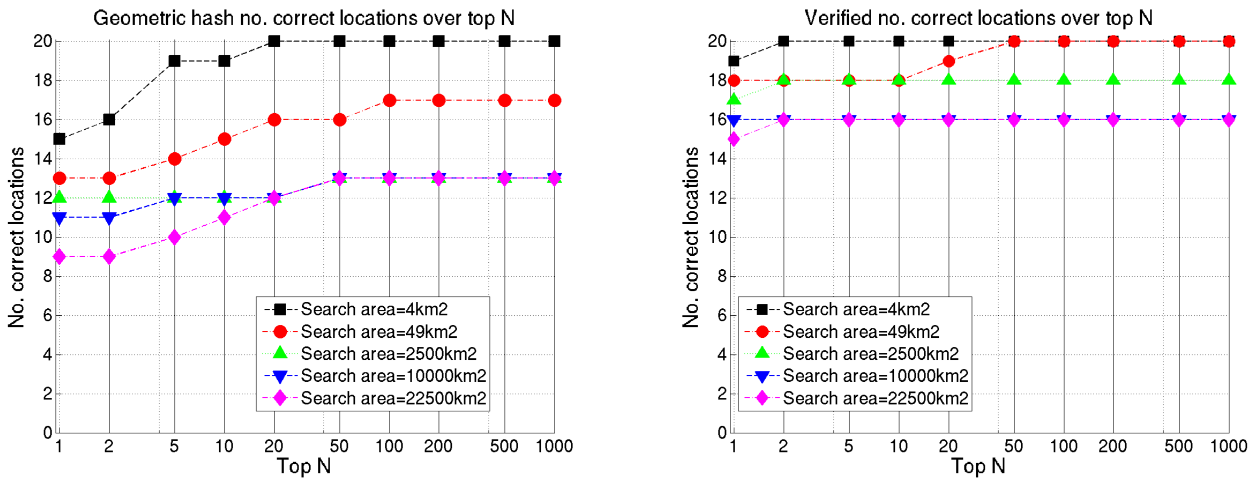
We assign likelihood values L_t (R_t match), and L_p (R_p match) to the two mean distances d_{m_t} and d_{m_p}. The likelihood function $f(x)$ of d_{m_i} for a correct match are modeled by the following function, a normal distribution function $\mathcal{N}(\mu,\sigma)$ if $x \geq \mu$, uniform with the max of the normal distribution if $0 \leq x < \mu$ and 0, otherwise. The normal distribution parameters are set experimentally, for R_t; μ_t = 6m, σ_t = 15m, for R_p; μ_p = 7.5m, σ_p = 15m. We set μ_t smaller than μ_p because for d_{m_t} outliers are considered. The likelihood for a mean distance value is $L_d = f(d_m)$. The final score S that the image sequence I matches the roads R is the joint likelihood $S = L_t L_p = f(d_t)f(d_p)$, d_t and d_p are handled as independent. At a correct match both likelihoods L_t and L_p have a high value, thus the final score is also high. For a false match at least one of the likelihoods has to decrease, thus the final score also decreases.

4.3 Experiments

We have tested the method on real aerial photos over Germany. 20 image sequences were acquired by a 21 MPixel Canon Digital Single-Lens Reflex (DSLR) camera with a fixed focal length mounted in nadir direction on an airplane. The internal camera parameters were known. A GPS device registered the position of each image for the ground truth. Each image sequence contains $17-20$ consecutive images taken with approximately 1 Hz frequency. The altitude over ground was between 1000 and 1500 m, which gives a GSD of $13-19$ cm/pixel. The image sequences contain roads with car traffic. We grouped the scenes as urban (U), suburban (S), industrial (I) and motorway (M). This classification is not strict; we arbitrarily used it for this paper. The scenes were relatively flat, thus the earth surface could be modeled by a plane. Studying mountainous areas is planned for future work. The scenes *Suburban* 1 and 4, 5 and 2, 6 and 3 and *Urban* 3 and 4 are approximately over the same area but with two month difference. The original images were rectified with a 2 parameter radial distortion model. All the processing was done on these rectified images. The tiles of the road database were 2000 m $\times$2000 m with 50% overlap. The grid size d_g for the geometric hashing was 15 m. The 5 longest straight tracks in the scene were used for the shortlist retrieval. The size of the scenes was around 2000 m $\times$ 500 m.

4.3.1 Quantitative evaluation

The PLBGHashing delivers a shortlist of locations, while the verification gives a ranking of these locations. To decide if a geolocation was correct, the geolocalization was compared to the GPS values. If the GPS positions were correct, the mosaic image was inspected visually by overlaying it in Google Earth. We extracted a TOPN value, which indicates if there is a correct location in the first N list elements. The number of possible locations and orientations is very high even for a small search area. For a scene of 1000 m $\times$ 1000 m in a search area of 2 km $\times$ 2 km with 15 m translation steps, 5° angle steps, and 3 scales $(\frac{1000}{15})^2 \times \frac{360}{5} \times 3 = 960000$. A search area of 22500 km^2 consists of 22201 overlapped tiles, thus the total number of possibilities is in the 10^{10} scale.



(Shortlist from *PLBGHashing*) (After verification)

Fig. 25. Completeness over top N.

In Figure 25 the aggregated number of correct localizations is plotted for different N values and search areas. There is a separate plot for the shortlist retrieval and the verification. The increase of Top1 by the verification was significant. We increased the search areas around the GPS position of the camera in steps 4, 9, 25, 49, 100, 225, 400, 625, 1225, 2500, 5625, 10000, 15625, 22500 km^2. The Figure 26 shows for each scene the maximal size of the search area in which it was correctly located in the topN after the verification.

19 scenes out of 20 could be located correctly as top1. The scenes *Industrial 1, 2* contain tracks acquired in large parking lots (the tracking algorithm might extract wrong tracks on dense parked cars) outside the road network. If the number of outliers becomes greater than what the verification accepts, then the verification ranks the locations wrong. Thus even if the correct location was in the shortlist it cannot be found. Therefore bad localization. The location of *I1* was wrong already at the smallest search area, while the *I2* could be localized only on 4 km^2. *I2* contains more tracks over streets as *I1*, which has tracks only about one motorway and industrial area. The other scenes show that if we have enough tracks on the roads, then the localization only weakly depends on the search area. On the Figure 27 we show the geolocalizations by projecting the mosaic images into the street network and as an overlay in Google Earth (the visualization as an image overlay is geometrically not fully accurate).

Some areas might be inherently ambiguous (e.g. chessboard like streets). An indicator for this ambiguity is if there are many candidates with the same number of votes in the shortlist and in the list after validation.

We defined the grid size for the geometric hashing as 15 m. This makes the method robust to small displacements in the tracks and it can handle multi-lane roads. If we assume 3 m

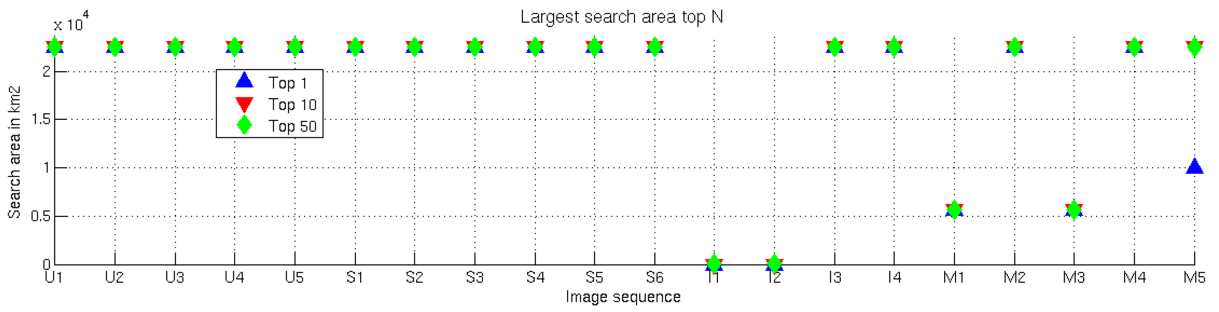


Fig. 26. The largest search area for each image sequence where the correct verified geolocation was in the TOPN in km^2

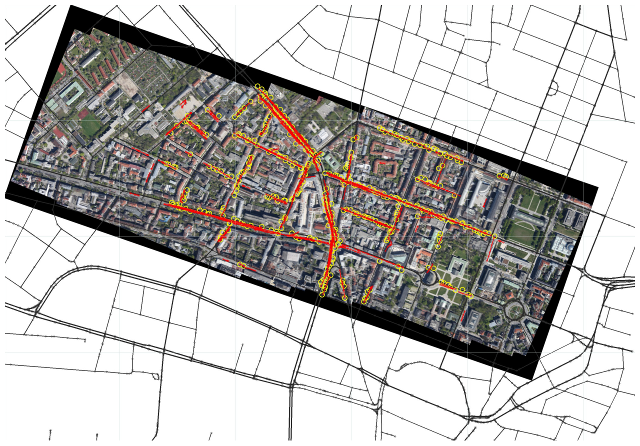
wide lanes, then 5 lanes are quantized to the same grid (i.e. the localization is invariant to this displacement). Roads with more than 5 lanes are sparse as large roads (e.g. motorways) in OSM are stored as two separate roads for each direction.

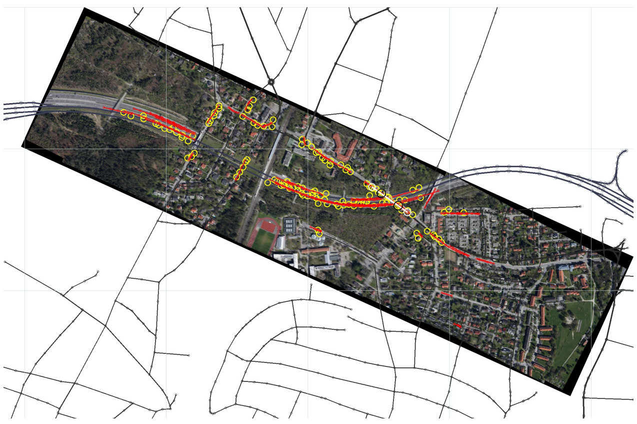
Our method is agnostic to splits or merges of the tracks as the assignment between line segments and tracks is not relevant. A track consisting of 10 line segments has the same effect as if it is split to 10 individual tracks.

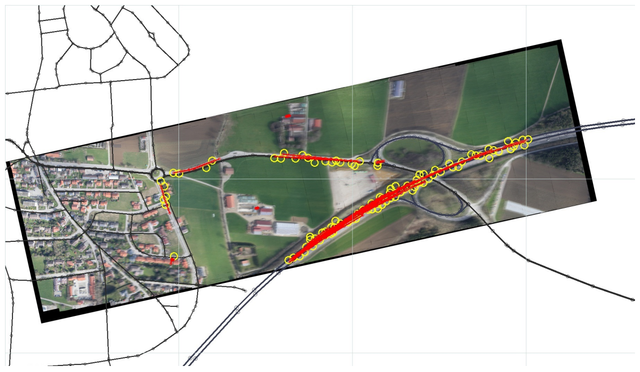
The search area 150 km × 150 km around the *Urban* scenes contain nearly 32000 km of roads, compiling to 2262M entries in the hash which requires 30500 MB memory. For the *Urban 1* scenes containing many tracks the retrieval took 9 minutes on the largest search area, for a scene with less tracks the retrieval is faster, for the *Suburban 1* it took 2 minutes. The average accuracy of the geolocalization is around 25 m, this also depends on the accuracy of the road data (A more accurate location could be acquired by optimizing the resulting location, but this was not our goal in this paper). Our implementation was written in C++, multi-threaded. The experiments run on an Intel Xeon E5-1650v2 processor with 6 cores. The verification took around 36 ms per shortlist element. Since the geometric hashing is very suitable for parallel implementation, the retrieval time on large search areas can be significantly reduced by utilizing more cores or machines.

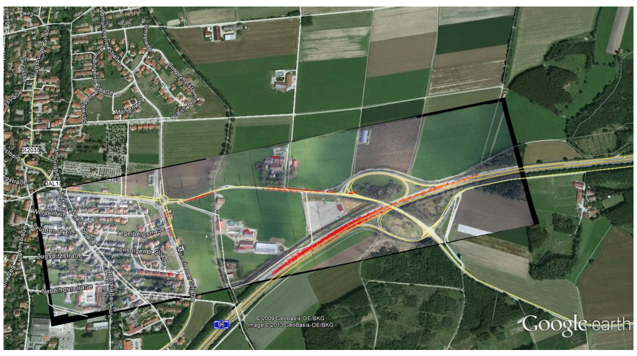
4.3.2 Comparison to simple chamfer matching

We consider chamfer matching (Barrow et al., 1977) with brute force search as a baseline. Without the PLBGHashing for shortlist generation, by only a brute force search and verification of each position the number of required verifications would be very high. Since the processing time of the verification is relatively long (36 ms per location), it would take $960000 \times 0.036s = 9.6$ hours on 2 km × 2 km search area $10^{10} \times 0.036s = 3.6 \times 10^{8}s = 11$ years for the largest search area (on a single thread). This is too long for practical usage.

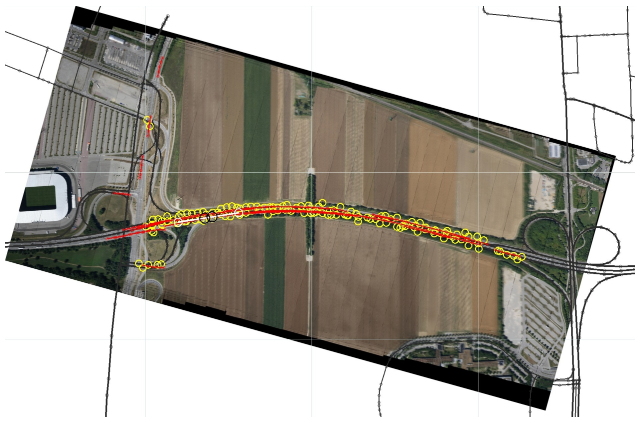

(a) *Urban 1.*


(b) *Suburban 1.*


(c) *Motorway 2.*



(d) *Motorway 2* as an overlay in Google Earth.


(e) *Industrial 3.* (90° rotated)

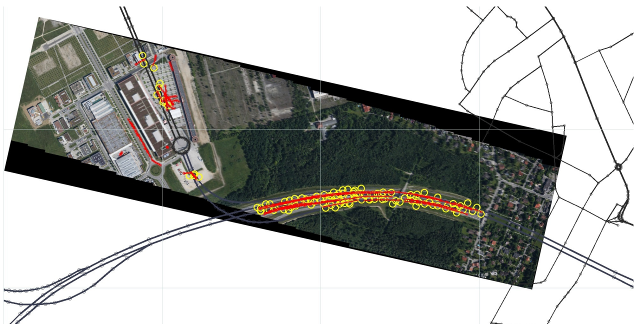

(f) *Industrial 1* false location.

Fig. 27. Geolocalization results. The mosaic images are projected into the street network. The roads are in black, the tracks are highlighted with red lines, the locations of votes from the *PLBGHashing* are marked with yellow circles. (When an image scene is only partially overlapped with a model during the shortlist retrieval, there are no vote circles at every matching track-road locations (e.g. on *Urban 1.*)) On (d) the geolocalized mosaic image with the tracks is overlaid in Google Earth. All these scenes were correctly located on a 22500 km^2 search area except (f). The (f) shows the scene *Industrial 1* located false on a 4 km^2 search area. The causes for this are; the scene contains 2 roads which makes it inherently ambiguous, multiple tracks are extracted on a parking lot, outside of the road network. Since there are more tracks in the parking lot than on the road, the verification matching handles the correct tracks as outliers. The color based road detection also does not work properly, since a parking lot has similar color as the road. The error is approximately 100m.

5 Enhancing road maps with street width by parsing aerial images

This chapter describes the paper Gellert Mattyus, Shenlong Wang, Sanja Fidler and Raquel Urtasun: *Enhancing Road Maps by Parsing Aerial Images Around the World*, International Conference on Computer Vision 2015 (Mattyus et al., 2015). The original paper can be found in the Appendix C.

The current basic road maps represent the topology of the road by their centerlines (additionally maybe the number of lanes and turning lanes at intersections) and they do not describe the physical dimensions of the road, e.g. the width. Most self-driving cars (e.g., Google car, participants of the Defense Advanced Research Projects Agency (DARPA) urban challenge) rely on detailed maps of the environment to facilitate navigation and perception. Maps can support localization (Liu et al., 2015; Brubaker et al., 2013), layout estimation (Liu et al., 2015) and holistic scene understanding (Wang et al., 2015a). The detailed maps are typically obtained via costly manual intervention and/or by driving surveying vehicles along the roads (e.g. the Google street view car). This needs huge effort to cover all the roads limiting the large scale applicability of current approaches.

We propose to exploit aerial images in order to augment maps with road geometry (i.e. road width). This is not an easy task as despite decades of research, large-scale automatic road segmentation from aerial images remains an open problem. Most approaches either do not deliver a topologically correct road network and/or rely on classifiers that have to be re-trained for each location in order to properly capture appearance variations. As a consequence they require tedious manual annotation for each region of the globe to be segmented, current approaches focus on a small set of locations.

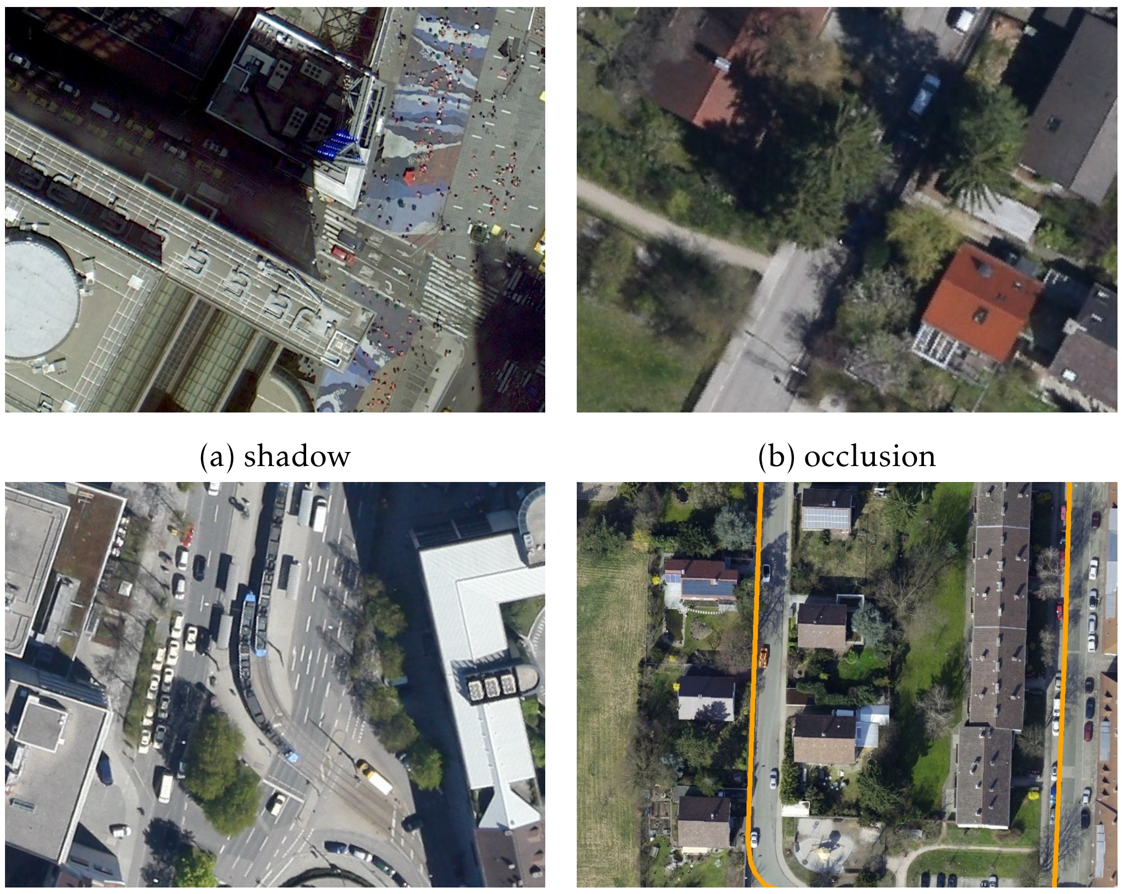
We extract pixelwise information from the images, a task typically formulated as semantic segmentation (i.e. assigning a class label for each pixel, see also section 2.4). In contrast, we propose to use existing maps (we apply the freely available OpenStreetMap (OSM)) to formulate the problem as one of inference in a Markov Random Field (MRF) which is directly parameterized in terms of the centerline of each OSM road segment as well as its width. This parameterization enables very efficient inference and returns the same topology as OSM. In particular, we can segment the OSM roads of the whole world in only 1 day when using a small cluster of 10 computers. Furthermore, this approach can be trained using only 1.5 km^2 of aerial imagery over Germany and is able to generalize to the entire world and produce state-of-the-art results without any further manual interaction. As the method reasons about the location of the centerline, it can handle and correct OSM mistakes as well as geo-localization/projection errors. This is not an easy task as illustrated in Fig. 28 due to shadows, occlusions and misalignments. The energy encodes the appearance of roads, edge information, car detection, contextual features, relations between nearby roads as well as smoothness between the line segments. All energy terms can be computed very efficiently via local, non-axis aligned integral images. Learning can also be done very efficiently using Structured Support Vector Machines (SSVMs) (Tsochantaridis et al., 2005) taking 1 minute on a desktop computer.

The coverage of OSM is very high in most areas, and thus by employing the proposed parameterization no roads are missed in the applied datasets. The lower road categories have to be excluded as they include forest tracks and pedestrian areas, which are not sufficiently visible in the aerial images. In other regions of the globe the coverage is not as dense and our approach might miss some roads. We refer the reader to the OSM project[15] for a more de-

[15]`http://wiki.openstreetmap.org/wiki/Stats`

tailed explanation of the coverage and its growth, and [16] for a comparison with other maps. Detecting new roads that are missing in OSM is our plan for future work.

The effectiveness of this approach is demonstrated by extracting road information from aerial images from different camera sensors taken around the whole world (e.g., Toronto, Sydney, New York, Manila, Nairobi). Importantly, only 1.5 km^2 imagery over Germany captured by one camera sensor is employed for training, illustrating the ability of our approach to generalize (domain adaptation). The aerial image datasets we are aware of are not labeled with the geometric information we want to extract. They either consider the road as a single centerline or label the other surfaces instead, e.g., the International Society for Photogrammetry and Remote Sensing (ISPRS) [17] dataset contains the ”impervious surfaces” class but no roads. Therefore we collect two new datasets namely Bavaria and aerial KITTI, which we manually annotate and show that our approach significantly outperforms all competitors. We then demonstrate the usefulness of our road priors for the task of semantic segmentation on KITTI ground images, and show that we can provide better cartographic priors than (Wang et al., 2015a).



(a) shadow (b) occlusion

(c) vehicles (d) misaligned centerline

Fig. 28. Road segmentation is challenging due to shadows, occluding trees and vehicles which make the appearance heterogeneous as well as OSM/projection misalignment errors.

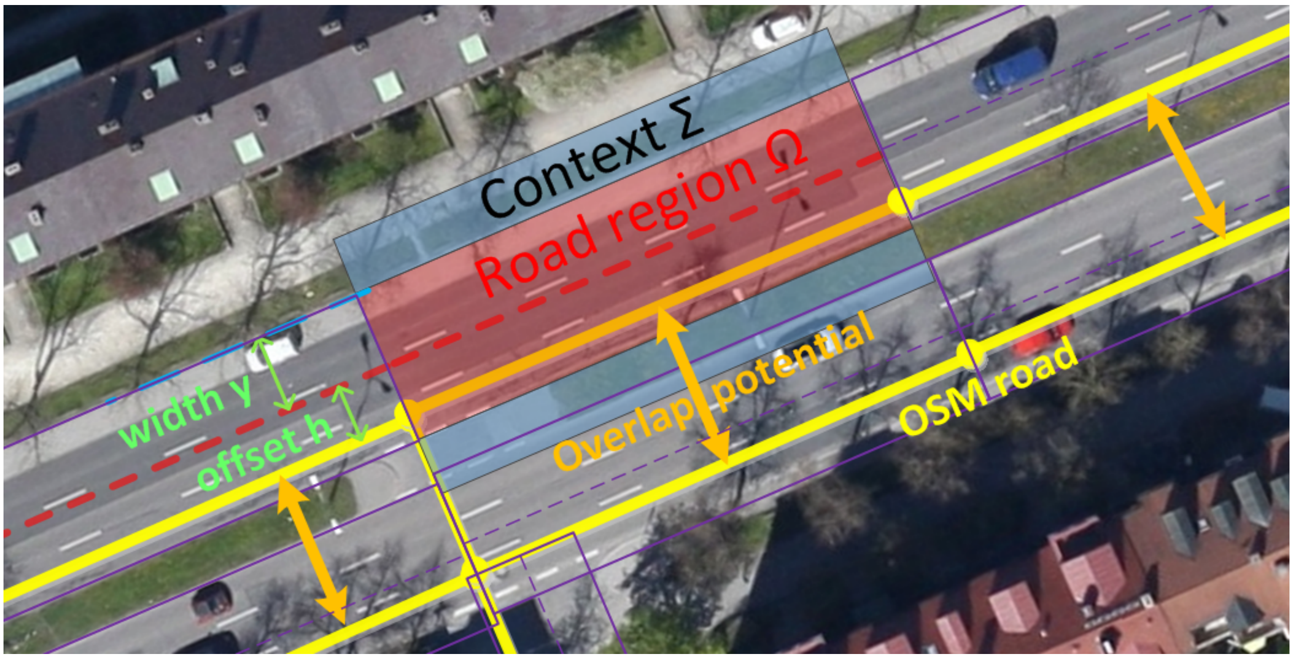


Fig. 29. Illustration of the road centerline with the width parameterized by the center offset h and symmetrical width y. The direction and length of the rectangle Ω_i is defined by the p_i, p_{i-1} points given by the street database. The context is depicted as Σ.

5.1 Related Work

Road segmentation in aerial images has drawn a lot attention for decades in the computer vision and remote sensing communities. However, it still remains an open-problem due to the difficulties in handling appearance variations and producing topologically correct segmentations. Early approaches search for objects that fulfill a pre-defined criteria. (Barzohar and Cooper, 1996) defines a geometric-stochastic model and estimates the roads by tiling the input image. (Stoica et al., 2004) use a Point Process to simulate and detect a network of connected line segments. We refer the reader to (Mayer et al., 2006) for a detailed literature review and comparison. These approaches, however, share a common drawback: they require manual parameter tuning. Learning based methods have been proposed to be more robust to appearance variations. Mnih and Hinton (Mnih and Hinton, 2010) proposed a two stage approach, where first a neural net is used to label patches independently. Road topology is then corrected using a post-processing step. This was extended in (Mnih and Hinton, 2012) to deal with noisy training labels by employing a robust loss function. However, this method suffers from block effects due to the patch-based prediction. (Wegner et al., 2013) model the road classification as a CRF, where the high-order cliques are sampled over straight segments or junctions to maintain a road-like network structure. In (Montoya-Zegarra et al., 2014) height-field contextual information captured from dense stereo matching is used to improve segmentation. This approach is computationally very expensive and results were shown in a single location. (Chai et al., 2013) sample graph junction-points using image consistency and shape priors, resulting in long computation times (4 min/image). (Turetken et al., 2013) formulate the delineation of linear loopy structures as an Integer program. However, only simple suburban scenes were tackled. Detecting roads without any prior map information is relevant for places without any maps available. Such places are nowadays scarce around the world, but the augmentation of existing maps is important around the whole globe. Leveraging the map priors effectively and efficiently to enhance the maps is not a trivial task.

[16] `http://tools.geofabrik.de/mc`

[17] `http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html`

Method	*Bavaria*						*Aerial KITTI*					
	IoU		F1		$\overline{\Delta h}$[m]	$\overline{\Delta y}$[m]	IoU		F1		$\overline{\Delta h}$[m]	$\overline{\Delta y}$[m]
	GT	Oracle	GT	Oracle			GT	Oracle	GT	Oracle		
Road Unary *	49.7	48.3	66.4	65.1	–	–	32.8	31.2	49.4	47.6	–	–
OSMxSeg	61.6	60.6	76.2	75.5	–	–	50.3	48.8	67.0	65.6	–	–
FSeg **	63.0	65.3	77.3	79.0	2.11	1.15	55.4	58.6	71.3	73.9	2.32	1.25
OSMFixed	64.7	66.9	78.6	80.2	1.75	1.45	51.0	53.8	67.6	70.0	2.38	1.21
Ours	**73.5**	**77.2**	**84.8**	**87.2**	**1.30**	**0.97**	**71.8**	**77.5**	**83.6**	**87.4**	**0.91**	**0.79**
Oracle	86.5	100	92.7	100	0	0	84.2	100	91.4	100	0	0

Table 4. Performance of our method vs baselines. The IoU and F1 values are in %, while $\overline{\Delta h}, \overline{\Delta y}$ are the mean absolute error of the offset and width measured in meters. * the road unary of (Winn et al., 2005) is applied. ** is the method of (Yuan and Cheriyadat, 2013).

Map information has been used also in computer vision and robotics communities. Aerial image and land cover attribute maps are exploited in (Lin et al., 2013) for single image geo-localization. Kalogerakis et al.(Kalogerakis et al., 2009) built a human travel prior from maps to geolocalize time-stamped photographs. Brubaker et al.(Brubaker et al., 2013) use road networks for self-localization. In (Matzen and Snavely, 2013) various maps of New York city were used to detect and localize cars from ground images. (Liu et al., 2015) use floor plans to localize and reconstruct in 3D single images in apartments. (Wang et al., 2015a) use OSM to generate a geographic prior for outdoor holistic scene understanding improving performance in 3D object detection, pose estimation, semantic segmentation and depth reconstruction.

The most similar works to ours (improving the road layout) are (Seo et al., 2012a) and (Yuan and Cheriyadat, 2013). (Seo et al., 2012a) formulate the road segmentation in aerial images as a weakly supervised classification problem, in which superpixels that overlap with road vector data are adopted as positive samples. Then all the superpixels are classified as road/non-road by applying a Markov Random Field with the Potts model. This method does not consider the potential inaccuracies of the road vector data which can lead to applying mislabeled data as the weekly labeled reference. A MRF with the Potts model can not incorporate the road topology , hence the solution does not preserve topology.

(Yuan and Cheriyadat, 2013) consider road segmentation as a width estimation problem. By analyzing the spatial distribution of superpixel boundaries along the direction of the road, the road width is retrieved for each line segment independently as the width corresponding to the maximum in the edge distribution. Our model can be reduced to this, if only the edge feature (50) is applied. However, a single feature without any a priori knowledge (e.g. the smoothness term in our model) is not robust to shadows and occlusions.

5.2 Augmenting Road Maps from Aerial Images

In this section we show how to augment world maps by parsing aerial images. In particular, we frame the problem as the one of inference in a Markov Random Field where the noisy cartographic map is employed to directly parameterize the problem. This parameterization is very robust and enables efficient inference.

5.2.1 Energy Formulation

In OSM, each road centerline is defined as a polyline chain (i.e., piece-wise linear curve) but no information about the road width is typically available. Unfortunately OSM roads are

Method	*Bavaria*						*Aerial KITTI*					
	IoU		F1		$\overline{\Delta h}$[m]	$\overline{\Delta y}$[m]	IoU		F1		$\overline{\Delta h}$[m]	$\overline{\Delta y}$[m]
	GT	Oracle	GT	Oracle			GT	Oracle	GT	Oracle		
Road+E+Car	72.2	75.7	83.8	86.2	1.57	1.10	70.7	76.3	82.8	86.5	1.05	0.84
Road+E+Car+‖	72.8	76.4	84.2	86.7	1.39	1.03	**71.8**	**77.6**	**83.6**	**87.4**	**0.91**	0.79
E+Ho+Co+Car	64.8	68.4	78.6	81.2	1.58	1.26	63.6	67.2	77.8	80.3	1.61	1.36
E+Ho+Co+Car+‖	69.7	72.6	82.1	84.2	1.52	1.09	63.5	67.4	77.7	80.5	1.49	1.26
All	73.0	76.2	84.4	86.5	1.51	1.08	71.2	76.8	83.2	86.9	1.05	0.84
All+‖	**73.5**	**77.2**	**84.8**	**87.2**	**1.30**	**0.97**	**71.8**	77.5	**83.6**	**87.4**	**0.91**	**0.79**
	Domain shift (train on one dataset, test on the other)											
Road+E+Car	70.0	74.3	82.4	85.2	1.45	1.06	66.0	71.0	79.5	83.0	1.33	0.89
Road+E+Car+‖	70.7	75.2	82.8	85.8	1.30	0.99	66.8	72.0	80.1	83.7	1.18	0.83
E+Ho+Co+Car	69.1	71.5	81.7	83.4	1.73	1.11	59.3	63.5	74.4	77.6	1.63	1.17
E+Ho+Co+Car+‖	70.4	73.4	82.7	84.6	1.43	0.98	62.0	65.7	76.5	79.3	1.57	1.36
All	70.8	75.1	82.8	85.8	1.37	1.02	**67.7**	72.8	**80.7**	84.3	1.20	0.84
All+‖	**71.7**	**76.1**	**83.5**	**86.4**	**1.27**	**0.93**	**67.7**	**73.2**	**80.7**	**84.6**	**1.08**	**0.79**

Table 5. Performance on Bavaria and Aerial KITTI with various features configurations. The IoU and F1 values are in %, while $\overline{\Delta h}, \overline{\Delta y}$ are the mean absolute error of the offset and width measured in meters. The ‖ symbol denotes the overlap potential between parallel roads. The other abbreviations are E: edge feature, Ho: homogeneity feature, Co: context feature.

not very accurate as they are either edited by volunteers without explicit quality control, or computed automatically from GPS trajectories. Furthermore, geo-localization and projection errors make the vertices of the polyline poorly aligned with the center of the road in aerial images. We refer the reader to Fig. 28 for an illustration of the difficulties of the problem. Thus we re-reason about their true location. Given a geo-localized aerial image, we model each road with a set of random variables representing for each vertex of the polyline an offset in the normal direction as well as the width of the road segment. We refer the reader to Fig. 29 for an illustration of our parameterization.

More formally, let $\mathbf{h}^j = \{h_1^j, \cdots, h_{l_j}^j\}$ be a set of random variables encoding the offsets of each vertex of the polyline that defines the j-th road, where l_j is the number of vertices for that road and $h_i^j \in [-30, 30]$ pixels. Our images have a resolution of 13 cm/pixel. Denote $\mathbf{y}^j = \{y_1^j, \cdots, y_{l_j}^j\}$ the width of each segment that compose the j-th road, with $\mathbf{y}^j \in [24, 50]$ pixels. Note that the hypothesis spaces for h and y are defined based on our empirical estimate of maximal road width and OSM projection error. Further, let $\mathbf{h} = \{\mathbf{h}^1, \cdots, \mathbf{h}^L\}$ and $\mathbf{y} = \{\mathbf{y}^1, \cdots, \mathbf{y}^L\}$ be the set of offsets and widths for all roads respectively. Denote $\mathbf{x}$ the input aerial image. We define the energy of our road segmentation as a sum of potentials encoding the image evidence, the presence of car detections, smoothness between widths and offsets of consecutive road segments and overlap constraints between nearby parallel roads

$$\begin{aligned} E(\mathbf{h}, \mathbf{y}) = &\sum_{j=1}^{L} \sum_{i=1}^{l_j} \mathbf{w}_{road}^T \phi_{road}(h_i^j, y_i^j, \mathbf{x}) + \sum_{j=1}^{L} \sum_{i=1}^{l_j} \mathbf{w}_{ap}^T \phi_{ap}(h_i^j, y_i^j, \mathbf{x}) + \sum_{j=1}^{L} \sum_{i=1}^{l_j} \mathbf{w}_{car}^T \phi_{car}(h_i^j, y_i^j, \mathbf{x}) \\ &+ \sum_{j=1}^{L} \sum_{i=1}^{l_j - 1} \mathbf{w}_{sm}^T \phi_{sm}(h_i^j, y_i^j, h_{i+1}^j, y_{i+1}^j) + \sum_{i,j,k,m \in P} \phi_{ol}(h_i^j, y_i^j, h_k^m, y_k^m) \end{aligned} \tag{48}$$

Note that the overlap energy does not have a weight as it is a hard constraint. We use three types of appearance features: distance to edges, homogeneity of the region as well as its context, i.e., $\phi_{ap} = [\phi_{edge}, \phi_{hom}, \phi_{context}]$. We now describe our potentials in more details.

5.2.1.1 Road classifier:

We employ a road classifier to compute for each pixel the likelihood of being road/non-road. The potential for each segment $\phi_{road}(h_i^j, y_i^j)$ is simply the sum of the likelihoods of all pixels in the non-axis aligned rectangle Ω_i^j defined by h_i^j, y_i^j (see Fig. 29 for an example).

$$\phi_{road}(h_i^j, y_i^j) = \sum_{p \in \Omega_i^j(h_i^j, y_i^j)} \varphi(p) \tag{49}$$

with $\varphi(p)$ the classifier score at pixel p. Note that this can be very efficiently computed using non-axis aligned integral images. Since we know the orientation of each segment, only a single integral image is necessary per segment. The integral image is also local to the segment, as the hypothesis space covers regions near the original OSM vertices.

5.2.1.2 Edge:

We expect the boundaries of road segments to match image appearance boundaries. Towards this goal, we compute edges using the line detector of (Dollár and Zitnick, 2013), and define the potential as the distance d from each rectangle boundary pixel to the closest image edge

$$\phi_{edge}(h_i^j, y_i^j) = \sum_{p \in \partial\Omega_i^j(h_i^j, y_i^j)} \min_{e \in \mathcal{E}} d(p, e) \tag{50}$$

with $\partial\Omega$ the boundary of the rectangle Ω, p a pixel and $\mathcal{E}$ the set of all lines returned by the line detector. We adopt the distance transform of (Felzenszwalb and Huttenlocher, 2012) to accelerate the computation.

5.2.1.3 Object detector:

We train a car detector using the detector of (Liu and Mattyus, 2015) described in section 3. Note that this task is extremely challenging as on average a car has only 30×12 pixels. We form a 2D feature for each car by computing $[s \cdot \sin(\Delta\alpha), s \cdot \cos(\Delta\alpha)]$, with $\Delta\alpha$ the angle between the segment and the car and s the confidence of the detector. The car potential ϕ_{car} is simply the sum of the features of all the detected cars that are inside the rectangle. Given the car features, the potentials can be computed efficiently using accumulators in a local region around each segment.

5.2.1.4 Homogeneity:

An important property of roads is that they are typically free of obstacles (otherwise we could not drive on them) and therefore we expect their appearance to be homogeneous. This is violated if there are vehicles, shadows or if our aerial view of the road is obstructed by trees, bridges or tunnels. In those cases we expect the other potentials to correct the mistakes. We capture homogeneity by first transforming the image into Luv space and computing for each channel the standard deviation of the appearance inside the rectangle. This potential can be efficiently computed using two non-axis aligned integral images per channel: one computing the sum of intensities and the other the sum of square intensities. Note that this calculation was used in (Viola and Jones, 2004) to normalize the Haar-like features in a sub-window.

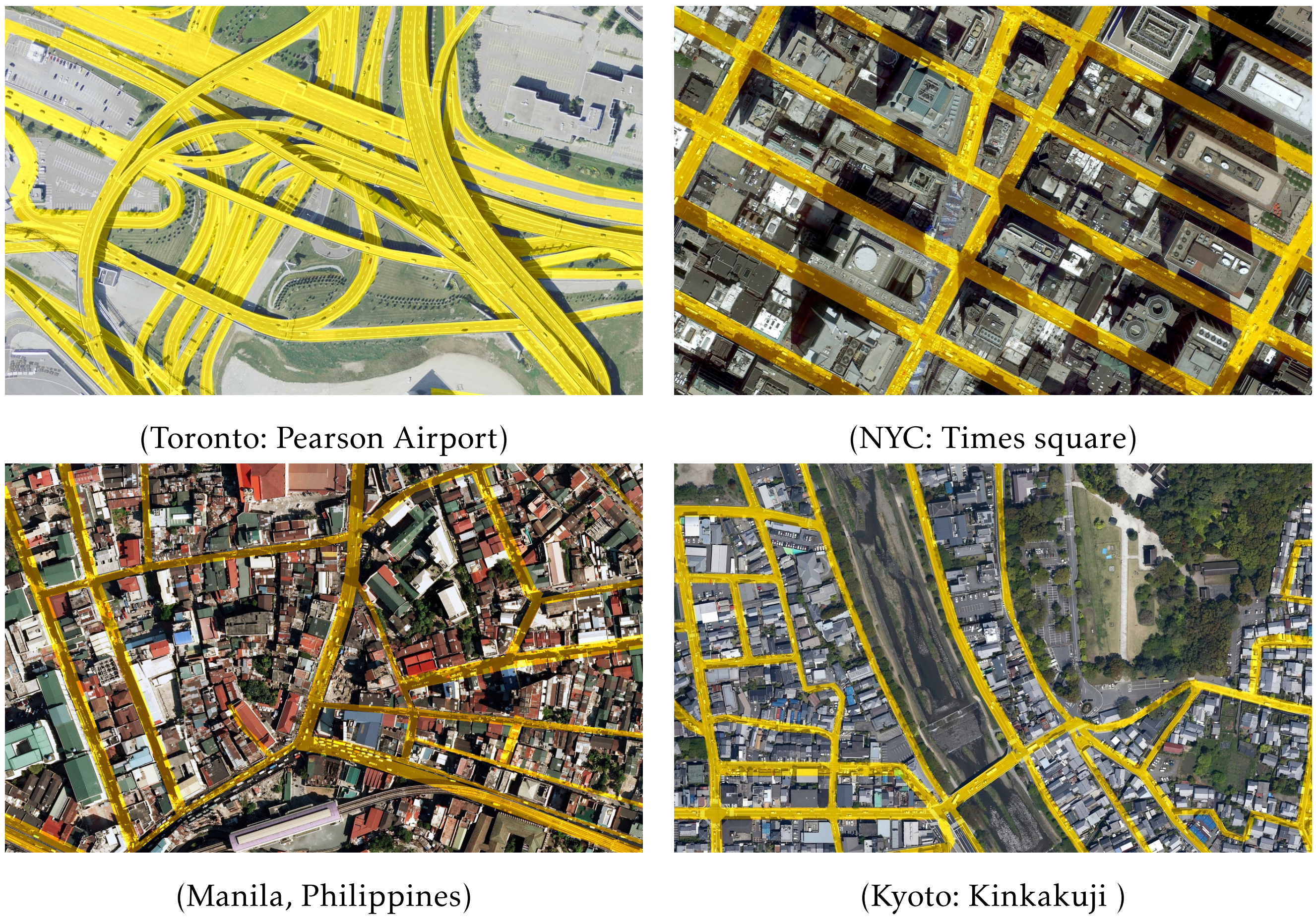


Fig. 30. Segmentation results on several cities over the world using the edge, homogeneity, context, car and overlap features. Note that the MRF was trained only on 1.5 km^2 imagery from the Aerial KITTI dataset.

5.2.1.5 Context features:

This feature encodes the fact that the road looks different than the area around it. Similar to (Viola and Jones, 2004), we compute the difference between the means of pixel intensities in the context and road rectangles, Σ_i^j and Ω_i^j respectively (see Fig. 29). The potential is computed by aggregating the difference across all Luv channels. Again, we use integral images for efficiency.

5.2.1.6 Smoothness:

The widths and offsets along the same road tend to be similar in nearby segments. Our smoothness potentials for both h and y are defined between consecutive segments along the same road as a weighted sum of ℓ_0 and ℓ_2 norms.

5.2.1.7 Overlap:

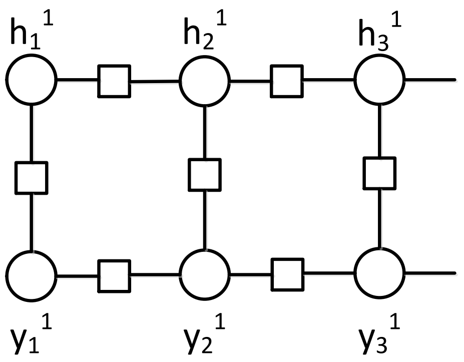
This is a hard constraint encoding the fact that two parallel roads cannot overlap. We enforce this for all roads that have similar orientation (within 20 degrees) and are close enough that they could overlap.

5.2.2 Inference

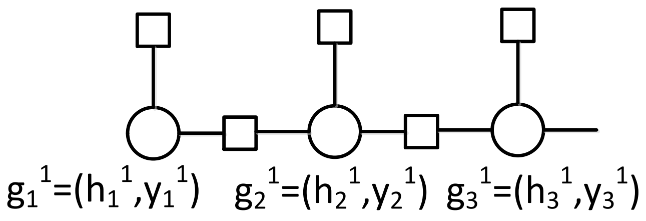
Inference in our model can be done by computing the minimum energy configuration

$$\{\mathbf{h}^*, \mathbf{y}^*\} = \operatorname*{argmin}_{\mathbf{h},\mathbf{y}} E(\mathbf{h}, \mathbf{y}) \tag{51}$$

with $E(\mathbf{h}, \mathbf{y})$ the total energy defined in Eq. (48). Note that due to the overlap constraint, the graphical models might contain loops. As a consequence exact inference is not possible. When there is no overlap, the graphical model can be reduced to a set of chains by merging the variables $g_i^j = (h_i^j, y_i^j)$. This is illustrated on Fig. 31. In a chain dynamic programming solves the inference exactly in polynomial time as described in paragraph 2.1.3.2.1.



(a)



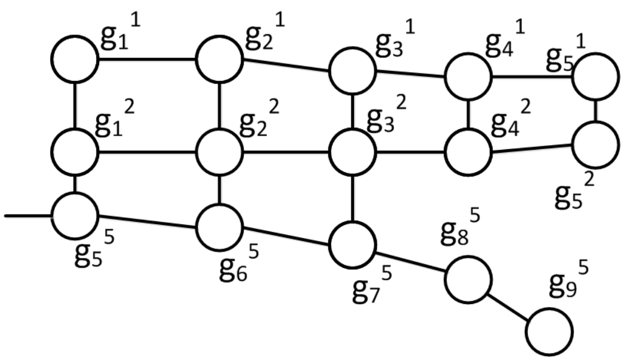
(b)

Fig. 31. The factor graph of the proposed graphical model in case of a single road. The unary potentials are pairwise defining loops in the graph (a). By merging the variables $g_i^j = (h_i^j, y_i^j)$ the graph can be reduced to a chain (b).

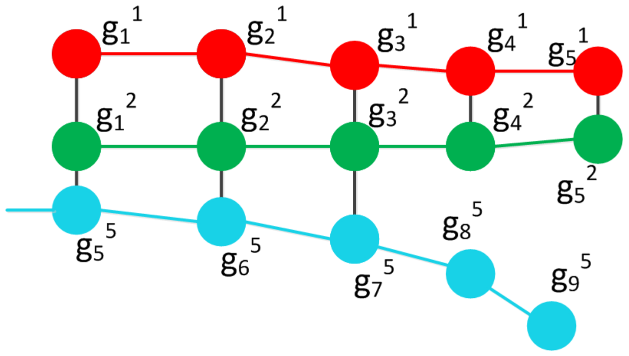
Inspired by the stereo work of (Chen and Koltun, 2014), we employ Block Coordinate Descent (BCD) to perform approximated inference. Towards this goal, we define each block in the BCD to form a chain since we can then solve each step to optimality. We then alternate between going over all horizontal and vertical chains to propagate the information. Note that since we solve each sub-step to optimality this procedure is guaranteed to converge. We refer the reader to Fig. 32 for an illustration, where to simplify the figure we have collapsed the width and offset variables in a single variable $g_i^j = (h_i^j, y_i^j)$. It is important to note that each of the BCD steps (i.e., optimization over a subset of variables) involve conditioning, and thus the pairwise potentials between a variable in the chain and a connected variable not in the chain are folded as unaries. Prior to BCD, we initialize all variables by performing inference along each road chain and ignoring the connections between nearby parallel roads. We refer the reader to Algorithm 3 for more details about the block coordinate descent.

5.2.3 Learning

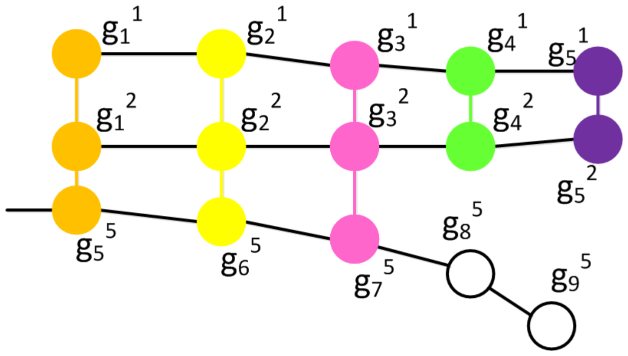
We learn the parameters of the MRF using Structured Support Vector Machine (SSVM)(Tsochantaridis et al., 2005) described in paragraph 2.1.3.3. The learning function



(a)



(b)



(c)

Fig. 32. Illustration of our BCD inference. Note that $g_i^j = (h_i^j, y_i^j)$. (a) Graphical model consisting of 3 roads that have overlapping constraints (i.e., vertical edges). We alternate between performing inference (b) over each road one at a time (red, green, blue), and (c) along chains on the vertical direction encoding the horizontal constraints, also one at a time (orange, yellow, pink, green, purple). Note that these operations involve conditioning, and thus the pairwise potentials between a variable in the chain and a connected variable not in the chain are folded as unaries.

Algorithm 3 Block coordinate descent inference (BCD)

```
1: Initialize $(\mathbf{h}, \mathbf{y})$ by minimizing Eq. (48) ignoring the overlap potentials
2: repeat
3:    for all roads $R_j$ do
4:       Minimize Eq. (48) w.r.t $\mathbf{h}^j, \mathbf{y}^j$ holding the rest fixed.
5:    end for
6:    for all overlap chains $O_i$ do
7:       Minimize Eq. (48) over the variables in the overlap chain
8:    end for
9: until no energy reduction or max number iterations
```

is expressed with slack variables:

$$\mathbf{w}^* = \min_{\mathbf{w} \in \mathbb{R}^D} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{n=1}^{N} \xi_n \tag{52}$$
$$\text{s.t. } E(\mathbf{h}, \mathbf{y}) - E(\hat{\mathbf{h}}^n, \hat{\mathbf{y}}^n) \geq \mathcal{L}(\hat{\mathbf{h}}^n, \hat{\mathbf{y}}^n, \mathbf{h}, \mathbf{y}) - \xi_n, \forall (\mathbf{h}, \mathbf{y}) \in \mathcal{H} \times \mathcal{Y} \setminus (\hat{\mathbf{h}}^n, \hat{\mathbf{y}}^n), \forall n$$

with $\mathcal{H} \times \mathcal{Y}$ the space of all possible labelings for $(\mathbf{h}, \mathbf{y})$. $\hat{\mathbf{h}}^n, \hat{\mathbf{y}}^n$ are the training samples and $\mathcal{L}(\hat{\mathbf{h}}^n, \hat{\mathbf{y}}^n, \mathbf{h}, \mathbf{y})$ is the loss function. Note that our definition is opposite from the one in (Tsochantaridis et al., 2005), as we have defined the features in terms of an energy minimization and not a score maximization. We employ the parallel cutting plane implementation of (Schwing et al., 2013) to learn the parameters. We use the intersection-over-union between the configuration and the ground-truth labels as our task loss. This can be computed as a pairwise term of h_i^j, y_i^j, and thus loss augmented inference can be done efficiently.

5.3 Experimental Evaluation

We perform our experiments on three different datasets: Bavaria, Aerial KITTI and World which were captured with different sensors. Note that we have access to RGB images without any elevation information. We conduct road pixel-wise annotations in all Bavaria and Aerial KITTI images. Note that the parameters not learned by SSVM were set via four-fold cross-validation.

5.3.1 Datasets

5.3.1.1 Bavaria:

This dataset is a collection of ortho-rectified aerial images captured by a DSLR camera mounted on a plane flying around the Bavaria region in Germany [18]. It covers urban, sub-urban and rural areas with motorways. The resolution is 13 cm/pixel on the ground. The total area is 4.95 km^2 containing 103 km of road.

5.3.1.2 Aerial KITTI:

This dataset consists of aerial images downloaded from Google Earth Pro over the city of Karlsruhe, Germany, covering the same area as the KITTI tracking benchmark (Geiger et al., 2013). The total area is 5.96 km^2 with 84 km of road. We resampled the images to be 13 cm/pixel resolution to be consistent with the Bavaria dataset.

5.3.1.3 World:

This dataset consists of aerial images downloaded from Google Earth Pro of landmarks all over the world, including metropolitan areas in Toronto, New York, Sydney, Mexico City, Manaus, etc., as well as rural areas in St. Moritz and Kyoto. For this dataset there is no annotation.

5.3.2 Metrics:

We use four metrics to measure performance: intersection over union, F_1 score, and mean of the absolute error of h and y. We consider two different ground truth labels when evaluating the performance: our human labeled road annotations as well as the maximum achievable score with respect to our model hypothesis, refer to as *Oracle*. The later can be computed by performing our MRF inference, by replacing our unary potentials with ground truth segmentations. For all quantitative experiments we perform four-fold cross-validation.

To compute our road classifier, we first convert the image to opponent Gaussian color space and extract a dense filter response map, with a filterbank composed of 17 edge-like filters (Winn et al., 2005). We oversegment the image using SLIC (Achanta et al., 2012) and calculate the mean and std of the filter responses in each superpixel. We then train a random forest classifier (Dollár and Zitnick, 2013) with this 34D input feature. Note that this road classifier was used in (Wegner et al., 2013) as unary potential.

[18] We will release these images and the ground truth upon publication.

Process step	Time (s) per km			
	Road+Edge+Car	Road+Edge+Car+‖	All	All+‖
Accumulator	0.07	0.069	0.126	0.122
Inference	0.031	0.092	0.032	0.095

Table 6. Running time for feature accumulator calculation and inference under various configurations. In sec per km of road.

5.3.2.1 Comparison to baselines:

We compare our approach to four baselines: The first one is the road classifier unary potential of (Wegner et al., 2013), denoted *Road Unary*. The second baseline, denoted as *OSMxSeg*, is computed by segmenting the image into superpixels using (Felzenszwalb and Huttenlocher, 2004) and labeling each super pixel as road if it is crossed by a road segment in OSM. We also reproduce the state-of-the-art method of (Yuan and Cheriyadat, 2013), denoted as *FSeg*, which also uses the OSM road data. To illustrate the effectiveness of our cartographic prior, the last baseline, denoted *OSMFixed*, projects OSM into the image and utilizes an empirical estimate of the road width. As shown in Table 4 our approach significantly outperforms all baselines in both Bavaria and aerial KITTI datasets. (see qualitative results in Fig. 34). Fig. 36 shows a comparison to (Yuan and Cheriyadat, 2013).

5.3.2.2 Importance of the features:

Table 5 depicts inference results for different combinations of features. Note that every feature contributes, and good performance can be achieved without using a road classifier. As a consequence, we do not need new training data for each different location in the world as the other features are very robust to appearance changes.

5.3.2.3 Segmenting the world:

Fig. 30 shows qualitative results from the *World* dataset with our model trained only on *AKITTI*. Our model works very well under many complex scenarios even with significant appearance changes, illustrating the generalization capabilities of our approach. Note that no re-training is necessary as we do not use the road classifier in our potentials.

5.3.2.4 Domain Adaptation:

We next show our method's domain adaptation ability. Towards this goal, we trained one model on Aerial KITTI and evaluate its performance on Bavaria, and vise versa. As shown in Table 5 our algorithm outperforms all baselines despite the fact that it is trained with different imagery. Furthermore, performance drops less than 5% Intersection over Union (IoU) when compared when we train on the same dataset we test on.

5.3.2.5 Processing time:

We implemented our method in C++ without multi-threading and test it on a laptop with an Intel Core i7-4600M processor. As shown in Table 6, our approach takes less than 0.13 s for computing all feature accumulators per km of road and less than 0.1 s per km for inference. The feature computation (road classifier, edge, car detector) relies on external code which takes around 0.1s per km of road. According to this performance, we estimate our algorithm could approximately segment all the OSM roads in the world in 1 day using a small cluster of 10 machines. We use the parallel cutting-plane SSVM of (Schwing et al., 2013) to learn the parameters of the model. This takes only 1 minute on a desktop computer.

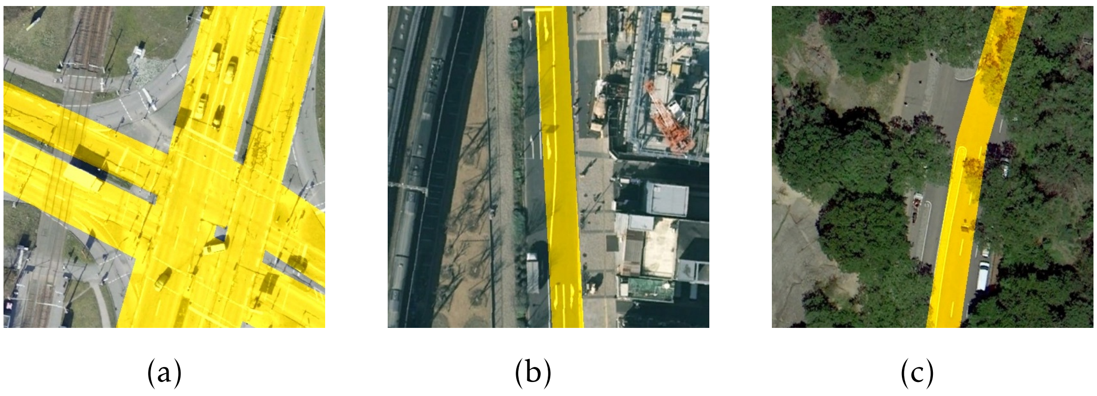


Fig. 33. Failure modes: (a) Missing turn lane intersection. (b) The extracted road is too narrow. (c) Road covered by trees.

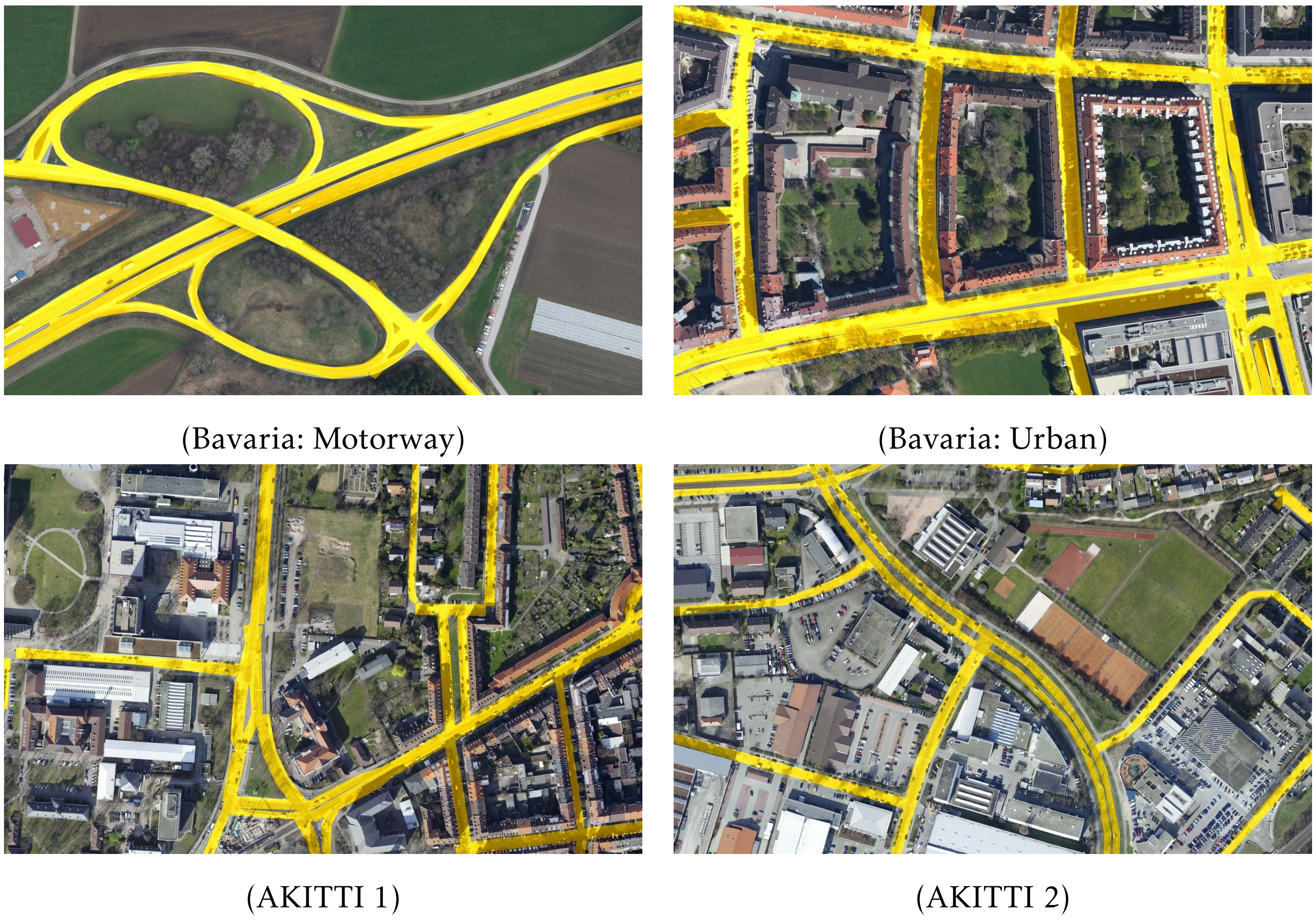


Fig. 34. Results on Bavaria and Aerial KITTI.

5.3.2.6 Ground-level Scene Understanding:

In this experiment we show that our enhanced maps can be used to improve semantic segmentation of ground images from KITTI. Towards this goal, we replace the road prior used in (Wang et al., 2015a) by the estimations of our method. This improves the geographic unary prior for the road class by 15%, see the Table 7. Qualitative results are shown in Fig. 35.

5.3.2.7 Failure modes and limitations:

Fig. 33 depicts failure modes. (a) At intersections the OSM might not include the turn lanes and our model cannot recover from this. (b) In some cases our features/weights are not good for the scene. This is more likely to happen in the strong generalization case. (c) The road can

	Sky	Building	Road	Sidewalk	Vegetation	Car
(Wang et al., 2015a)	32.41	59.25	63.01	36.41	7.36	35.65
Ours	32.41	59.10	**78.71**	**41.96**	7.36	35.65

Table 7. Our method improves the geographic priors of (Wang et al., 2015a). All values are IoU in %.

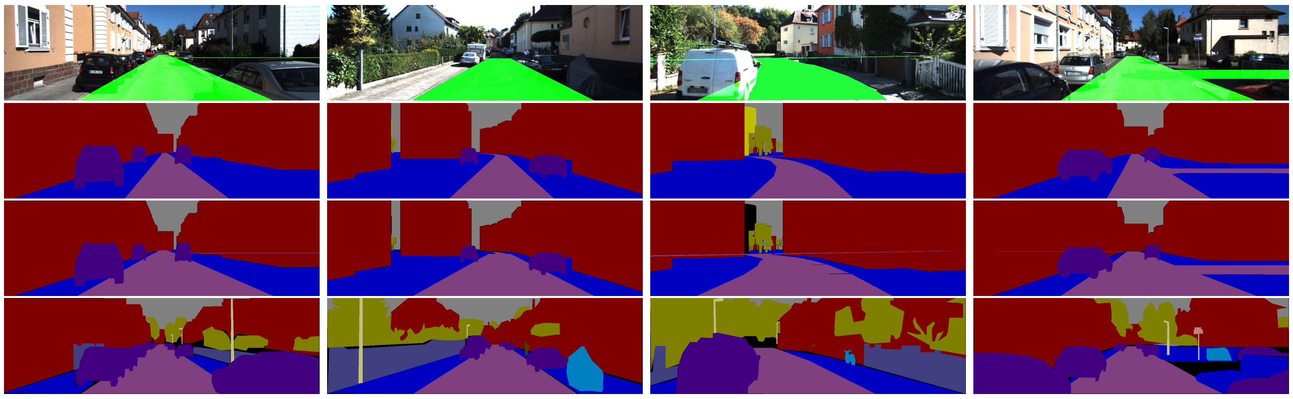


Fig. 35. (Top) Our road extracted from aerial images (green) projected into Kitti ground images. (2nd row): Geographical unary of (Wang et al., 2015a). (3rd row): Geographical unary with our road estimate. (Bottom) Ground truth. Road (pink), sidewalk (blue), building (red), car (purple).

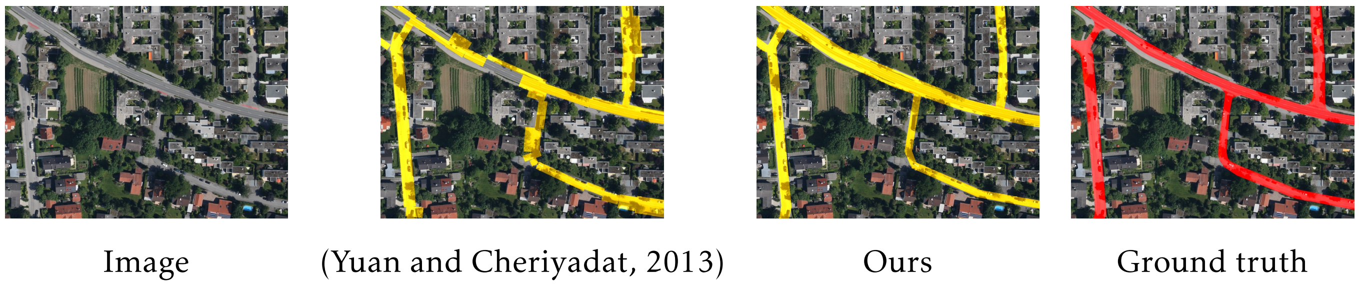


Fig. 36. Comparison to (Yuan and Cheriyadat, 2013): Our approach works significantly better than the baseline.

be (partly) covered, and we only extract the visible part of the road. Additional challenges are posed by historical city centers were the roads might not be visible as well as developing countries, where only satellite images with much lower resolution than aerial images might be available.

6 Fine-grained road segmentation by parsing ground and aerial images

This chapter describes the paper Gellert Mattyus, Shenlong Wang, Sanja Fidler and Raquel Urtasun: *HD Maps: Fine-grained Road Segmentation by Parsing Ground and Aerial Images*, Conference on Computer Vision and Pattern Recognition (CVPR) 2016, (Mattyus et al., 2016). The submitted camera ready paper can be found in the Appendix D.

In this paper an approach is presented to enhance existing maps with fine grained segmentation categories such as parking spots and sidewalk, as well as the number and location of road lanes. This method is able to efficiently estimate these fine grained categories by doing joint inference over both, monocular aerial imagery, as well as ground images taken from a stereo camera pair mounted on top of a car. Important to this is reasoning about the alignment between the two types of imagery, as even when the measurements are taken with sophisticated GPS+IMU systems, this alignment is not sufficiently accurate. We demonstrate the effectiveness of our approach on a new dataset which enhances KITTI Geiger et al. (2013) with aerial images taken with a camera mounted on an airplane and flying around the city of Karlsruhe, Germany. The method for the alignment between aerial and ground images can be also applied later for precise self-localization of the vehicle using the extracted detailed map.

6.1 Related Work

For several decades, researchers from various communities (e.g., vision, remote sensing) have been working on automatic extraction of semantic information from aerial images. In the following, we summarize the approaches most relevant to our work.

6.1.1 Aerial image parsing:

Early approaches employed probabilistic models that aimed to produce topologically connected roads. Barzohar and Cooper (1996) defined a probabilistic model that tiled the image into patches, performed road inference inside each patch via dynamic programming, and then “stitched” together high-confidence patches to ensure road connectivity. Recent work exploits learned classifiers to perform semantic segmentation. (Mnih and Hinton, 2010, 2012) trained a neural net to classify pixels in local patches as road. They employ a post-processing step to ensure a consistent road topology across the patches, which is, however, prone to block-effects. (Wegner et al., 2013) segments the road by defining an MRF on superpixels. High-order cliques are sampled over straight segments or junctions to encourage a road-like network structure. Due to complexity of high order terms a sampling scheme is used to concentrate on more important cliques. (Chai et al., 2013) samples graph junction-points using image consistency and shape priors. A full review of this large field is out of scope of this paper, and we refer the reader to (Mayer et al., 2006) for a detailed review.

6.1.2 Aerial parsing with maps:

While proven useful in many computer vision and robotics applications (Kalogerakis et al., 2009; Matzen and Snavely, 2013; Brubaker et al., 2013; Wang et al., 2015a), few works employ map information for parsing aerial images. Seo et al. (2012a) uses a screenshot of the vector map as a weak source of ground-truth for training a road classifier. (Yuan and Cheriyadat, 2013) exploit road center-lines from OSM maps as a ground-truth road location and performs road segmentation by estimating the width of the road. This is done by finding boundaries of superpixels along the direction of the road, and ignoring dependencies across different line (road) segments. However, the alignment between OSM and aerial images is far from perfect. To solve this problem, (Mattyus et al., 2015) proposed a MRF which reasons about re-positioning the road centerline and estimating the width of the road. Smoothness is incorporated between consecutive line segments by encouraging their widths to be similar. In our work we go beyond this approach by introducing a formulation that reasons about more fine-grained road semantics such as lanes, sidewalks and parking spots, and exploits simultaneously aerial images as well as ground imagery to infer this information.

6.1.3 Fine-grained road parsing:

Very few works exist that extract detailed segmentation. (Porway et al., 2009) propose a hierarchical probabilistic grammar to parse smaller-scale aerial regions into roads, buildings, vehicles and parking lots. Classifiers are first employed to generate object/building/vegetation proposals while the grammar imposes semantic and geometric constraints in order to derive the final parse. Learning and inference are both hard in grammars, and computationally expensive sampling techniques typically need to be employed. In our work, we are aiming at a detailed parsing of the roads into sub-categories. Unlike (Porway et al., 2009), we exploit OSM information in order to derive an efficient formulation.

The work most related to ours is (Seo et al., 2012b) which exploits the map as a screenshot of the road vector map to perform road and lane estimation. The authors take a pipeline approach, where, in the first step, road lane hypotheses are generated based on the output of the road classifier and detected lane markings. In the second step, the authors provide heuristics to "track" the lane hypotheses and connect them into a single lane labeling.

6.1.4 Aerial-to-ground reasoning:

Recent work aims to exploit both aerial and the ground-view, mainly for the problem of geo-localization. In (Lin et al., 2015), a deep neural network is used to match ground images with aerial images in oblique views. The matches come from facade to facade matching and therefore cannot be extended to orthographic aerial images. In (Shan et al., 2014), 3D reconstructions from the ground images are matched to oblique views of aerial images. (Lin et al., 2013) learn cross-view matching between ground images, aerial orthographic photos and land cover attributes. This extends the image geolocalization to areas not covered by ground images. Forster et al. (Forster et al., 2013) match the computed 3D maps of UAVs and ground robots for localization and map augmentation. This method relies on matching 3D information and therefore needs multiview images both from above and on ground. In our work, we exploit the maps as well as ground and aerial imagery to perform fine-grained road parsing. We are not aware of prior work that tackles this problem.

6.2 Fine-grained Semantic Parsing of Roads

We now describe our model that infers fine-grained semantic categories of roads from aerial and ground images. In particular, we are interested in estimating *sidewalks*, *parking*, *road lanes* as well as *background* (e.g., vegetation, buildings). Towards this goal we exploit freely available cartographic maps (we use OSM), that provide us with the topology of the road network in the area of interest. Our approach takes as input an aerial image $\mathbf{x}_A$, a road

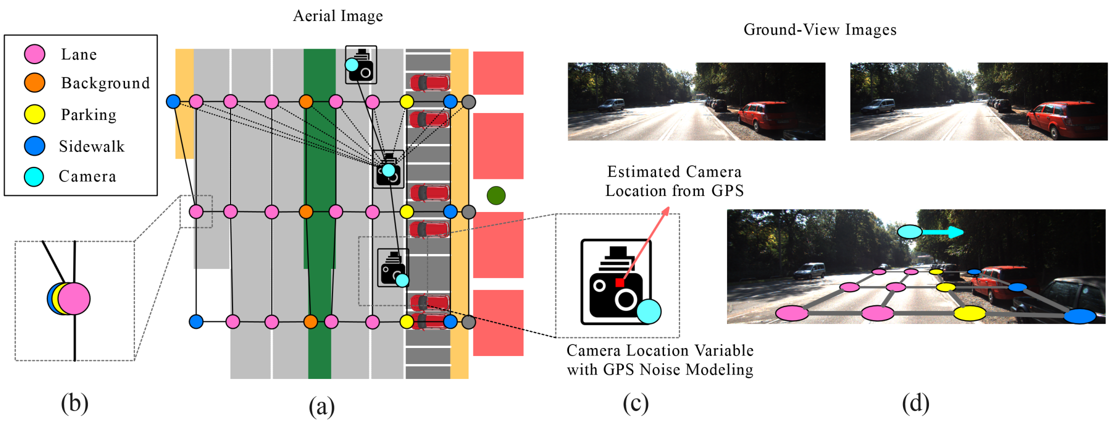

Aerial Image
Lane
Background
Parking
Sidewalk
Camera
Ground-View Images
Estimated Camera Location from GPS
Camera Location Variable with GPS Noise Modeling
(b)
(a)
(c)
(d)

Fig. 37. **Illustration of our model:** (a) Parameterization of our approach. Our random variables are the absolute location of the different region boundaries (e.g., sidewalk) as well as the alignment between air and ground. (b) Our formulation allows a random variable to take the same state as the previous node, collapsing a region to have 0 width. (c). For each ground-view image, a random variable models the alignment noise. (d). Projection of our parameterization on the ground-view.

map $\mathbf{x}_M$ and a set of ground stereo images $\mathbf{x}_G$, which are taken by a calibrated stereo pair mounted on top of a car. The map $\mathbf{x}_M$ is composed of a set of roads, where each road is defined as a piece-wise linear curve representing its centerline.

6.2.1 Model Formulation

We formulate the problem as the one of inference in a Markov Random Field (MRF), which exploits deep features encoding appearance in both aerial and ground images, edge information, smoothness in the direction of the road as well as restrictions between parallel roads to avoid double counting the evidence. Our model encodes each street segment in the aerial image with 15 random variables encoding all possible combinations of *background* (B), *sidewalk* (S), *road lanes* (L) and *parking* (P). In particular,

$$\mathbf{y} = (y_1, \cdots, y_{15}) = (B_1, S_1, B_2, S_2, P_1, L, P_2, S_3, B_3, S_4, B_4)$$

with B_1, B_4 the rightmost (leftmost) border of the background. We model roads with up to 6 lanes, i.e., $L = (L_1, L_2, L_3, L_4, L_5, L_6)$. We allow all variables (but L_6) to take the state of the previous random variable in the sequence (i.e., $y_i = y_{i-1}$), encoding the fact that some of these regions might be absent, e.g., there is no parking or sidewalk. This is not the case for L_6 forcing the fact that at least one lane should be present. We define the states of each random variable to be $[-15, 15]$m from the projection of the OSM centerline in the aerial image (Fig. 37). This discretization represents pixel increments. Note that while there are 15 random variables, $\mathbf{y}$ defines 16 different regions as B_1 and B_4 are not limited on the left (right). Each region width is simply defined by $w_i = y_i - y_{i-1}$, while the width of B_1 is defined as $w_1 = -15m + y_1$, and the width of B_4 as $w_{16} = 15m - y_{16}$, since $-15m$ and $15m$ are the beginning and end of the state space. Note that the combination (B, S, B, S) is necessary (both on the left and right), as there are many bike lines in Germany (where our imagery is captured), and it is not possible to distinguish them from the sidewalk. Fig. 37 illustrates the model.

Each of our ground images comes with a rough alignment with the aerial image as we have access to a GPS+IMU and the cameras are registered w.r.t these sensors. This alignment is, however, noisy with 1.67m error on average. Thus, our model reasons about the alignment when scoring the ground images. Towards this goal, we define $\mathbf{t} = (t_1, \cdots, t_n)$ to be a set of random variables (one per ground image) representing the displacement in the direction perpendicular to the OSM road segment. We define the state space of each misalignment to be $t_i \in (-4m, 4m)$. This is discretized to represent pixel increments.

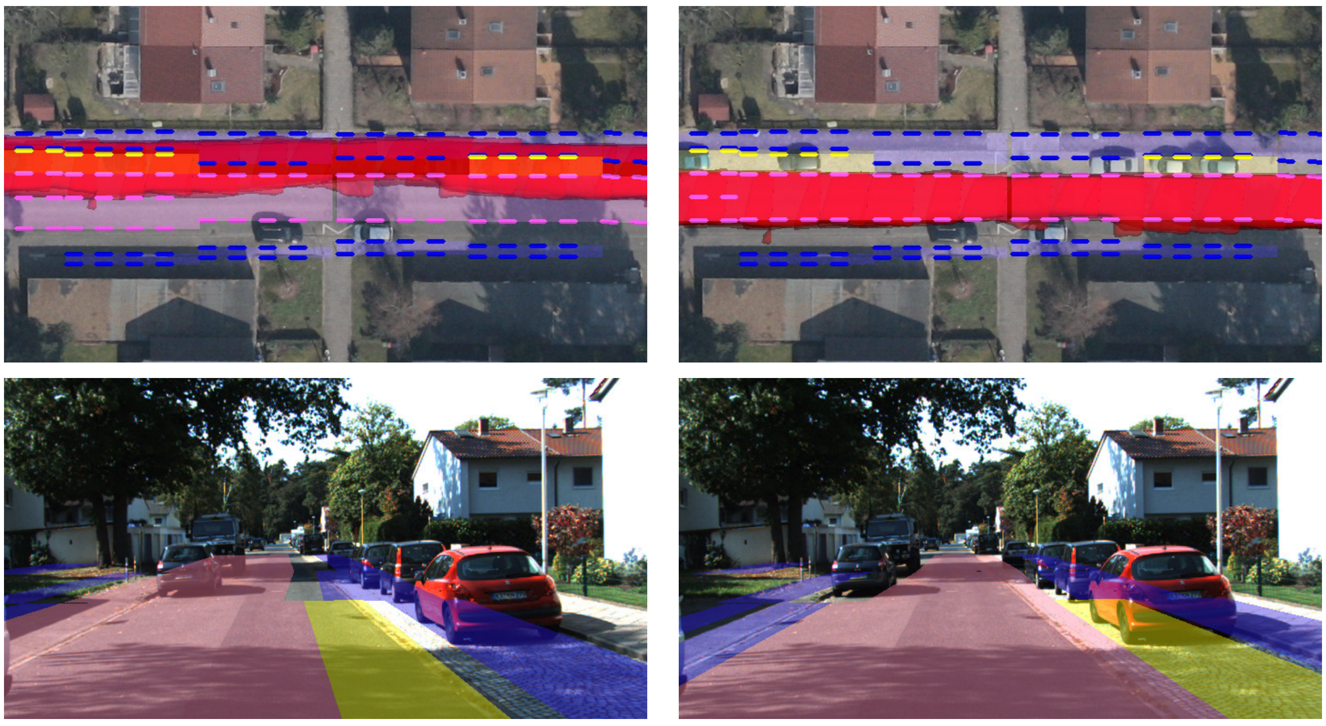


Fig. 38. **Effect of reasoning about alignment: (left)** alignment given by GPS+IMU, **(right)** alignment inferred by our model. **(top)** Ground road classifier projected into the aerial image (shown in red). **(bottom)** Our semantic classes projected on the ground image. Our joint reasoning significantly improves alignment.

We define the energy of the MRF as to encode the information contained in the ground and aerial images as well as smoothness terms and constraints on the possible solutions:

$$E(\mathbf{y},\mathbf{t},\mathbf{x}_A,\mathbf{x}_M,\mathbf{x}_G) = E_{\text{air}}(\mathbf{y},\mathbf{x}_A) + E_{\text{ground}}(\mathbf{y},\mathbf{t},\mathbf{x}_G) + E_{\text{smooth}}(\mathbf{y},\mathbf{t},\mathbf{x}_M) + E_{\text{const}}(\mathbf{y}) \quad (53)$$

We now define the potentials we employ in more detail.

6.2.1.1 Aerial semantics:

We take advantage of deep learning in order to estimate semantic information from aerial images. In particular, we create pixel-wise estimates of 5 semantic categories: *road*, *sidewalk*, *background*, *building* and *parking*. We exploit the CNN for segmentation (Simonyan and Zisserman, 2014; Schwing and Urtasun, 2015) trained on ILSVRC-2014, which we fine-tune for a 5-label classification task: road, parking spot, sidewalk, building and background. To train the network we created training examples by extracting patches centered on the projection of the OSM road segments. If the road segment is too long (i.e., long straight road) we create an example every 20m. We further perform data augmentation by applying small rotations, shifts and flips to the training examples. The output of the soft-max is a downsampled segmentation. To create our features, we upsample the softmax output using linear interpolation as in (Chen et al., 2015). To save computation, we only apply the network in the region of interest (regions of the image that are close to OSM roads). The aerial semantic potential then encodes the fact that our final segmentation should agree with the semantics estimated by the deep net. Towards this goal, we define 5 features for each of our 16 regions, one per label of the deep net. Each feature simply aggregates the output of the softmax in that region. Recall that each region is defined by two consecutive random variables, e.g. the first sidewalk is defined by y_1, y_2, that is B_1, S_1. We refer the reader to Fig. 37 for an illustration. While this potential seems pairwise in nature, we can further decompose it into unary potentials via accumulators $\mathcal{A}$ perpendicular to the road direction. These are simply gener-

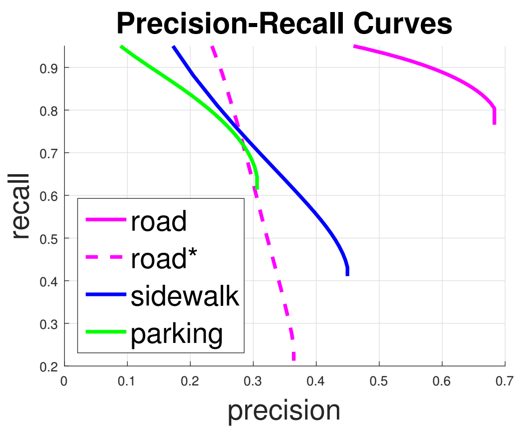


Fig. 39. Precision-Recall curves for our deep classifier and the road classifier of (Mattyus et al., 2015) marked with * and in dashed.

alizations of integral images from axis aligned accumulators to accumulators over arbitrary directions. We thus define

$$\phi_{cl}(y_i^j, y_{i-1}^j) = \sum_{p \in \Omega_i^j(y_i^j, y_{i+1}^j)} \varphi(p) = \mathcal{A}(y_{i+1}^j) - \mathcal{A}(y_i^j) \tag{54}$$

with y_i^j the i-th variable of the j-th road segment, and $\varphi(p)$ the softmax output interpolated at pixel p. To compute this features, we only need 5 accumulators per road segment, one for each semantic class that the deep net predicts.

6.2.1.2 Aerial edges:

This potential encodes the fact that the location of the boundaries between regions should be close to image edges. We thus apply the edge detector of (Dollár and Zitnick, 2013) to detect edges in our aerial images. We then define the potential to be the sum of the edges on the boundary between consecutive regions. To make it more robust we thicken the boundary to be of size 3 pixels.

6.2.1.3 Along the road smoothness:

We encode smoothness along the road by encouraging consecutive road segments to be similar. In particular, we use the ℓ_1 distance between consecutive road estimations in the direction of the road, i.e.

$$\phi_{\text{sm}}(y_i^j, y_i^{j+1}) = |y_i^j - y_i^{j+1}|$$

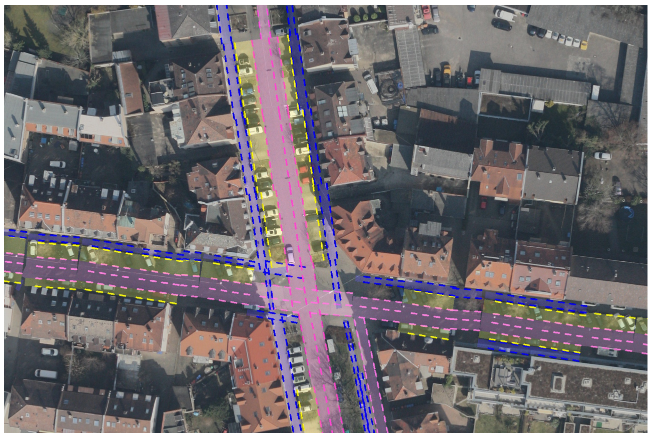
6.2.1.4 Parallel roads:

The regions of close by parallel roads can overlap. To avoid double counting the evidence, we incorporate an additional constraint that forces S_1 of the second road to be bigger or equal to B_4 of the first road or vice versa. We refer the reader to Fig. 37 for an illustration.

6.2.1.5 Road collapse constraints:

We force each variable y_i to have a state higher or equal than the previous variable, so that the order is preserved. Note that equal means that a road can collapse (i.e., does not exist)

$$\phi_{coll}(y_i, y_{i+1}) = \begin{cases} \infty & \text{if } y_{i+1} < y_i \\ 0 & \text{otherwise} \end{cases}$$


(a) Dense urban area.

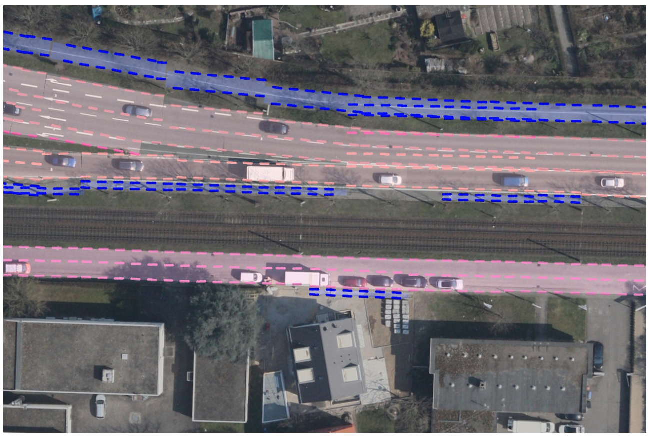

(b) Splitting road plus a bike lane along the street.

Fig. 40. Visualization of our semantic road parsing results using only aerial images. The road lanes are shown with shades of pink, the sidewalk with blue and the parking spots with yellow.

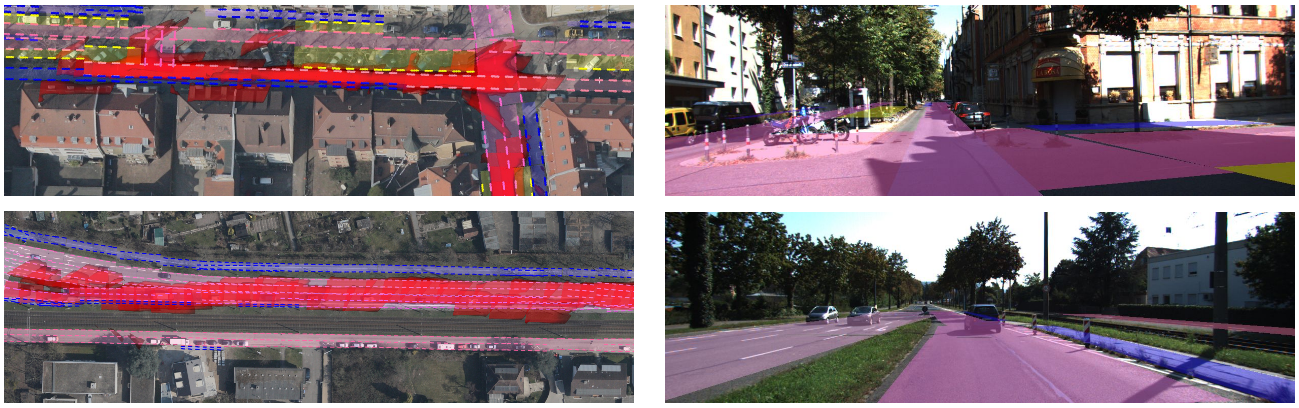


Fig. 41. Left: The ground road detection with red projected into the aerial image after alignment and road layout estimation. Right: The semantic lanes projected back into the aligned ground image. These scenes are all challenging with parallel roads, parking spots and intersections. The bottom image is especially difficult since it is an urban pedestrian area. Note that the aerial and ground images were taken with several years difference in different seasons. Pink is road, blue is sidewalk and yellow marks parking spots.

The only exception is L_6, which we force to have non-zero width as otherwise we could have a road segment without road. Thus

$$\phi_{ex}(L_5, L_6) = \begin{cases} \infty & \text{if } L_6 \leq L_5 \\ 0 & \text{otherwise} \end{cases}$$

6.2.1.6 Lane size constraint:

This constraint forces each region, if present, (i.e., if it is not taking state 0) to have a minimal and maximal size. In particular, we use (1m-3m) for sidewalk, (1.8m-4.5m) for parking and (2.3m-4.6m) for each road lane. Note that width 0 is allowed so that regions can disappear if they are not present in the road segment (e.g., we only have two lanes, there is no sidewalk on the highway). The intervals for the lanes are estimated based on the standards of German roads, while the sidewalk and parking intervals are computed based on empirical estimates.

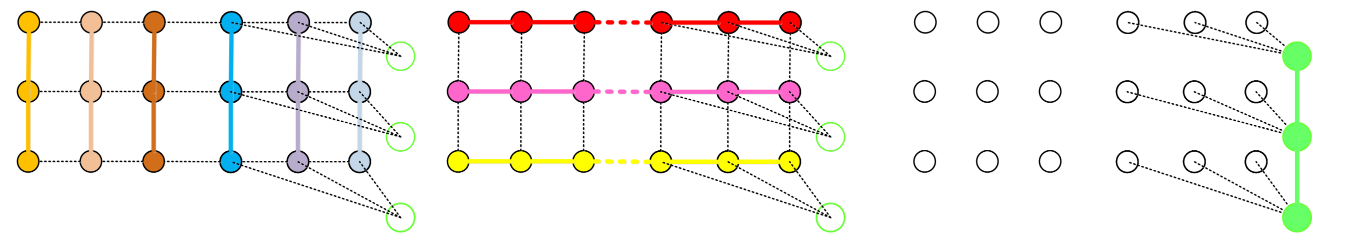


Fig. 42. **BCD:** The graph shows a simplified network with two parallel roads (each with 3 random variables) and one ground image per segment connected to the right road. BCD alternates between three types of updates. (a) Along the road updates: we optimize over each chain with the same color (while holding all other variables fix). The pairwise terms fold to unaries (see dashed black lines). (b) Perpendicular to the road updates: we do inference for the nodes with the same color (holding the rest fix). (c) Along the ground alignments: We minimize only the t variables which are depicted in green. The y variables are fixed and are depicted in black.

6.2.1.7 Centerline prior:

As our images are well registered with OSM, we include a prior that the centerline of our model should be close to the centerline of OSM. In particular,

$$\phi_{cen}(L_3) = \begin{cases} \|L_3 - l\|_2 & \text{if } -7.5 \leq L_3 \leq 7.5 \\ \infty & \text{otherwise} \end{cases}$$

with l the location of the centerline.

6.2.1.8 Ground semantics:

We take advantage of deep learning in order to estimate semantic information from ground images. We exploit the VGG (Simonyan and Zisserman, 2014) implementation of (Schwing and Urtasun, 2015) trained on PASCAL VOC, which we fine-tuned to predict the same 5 classes as the aerial semantics (*road*, *parking*, *sidewalk*, *building* and *background*). We estimate the ground plane from the stereo image and project pixels belonging to this plane to the aerial image via a homography. We then define our ground semantic potential to encourage the segmentation to agree with the aligned ground image segmentation projected to the aerial image. Towards this goal, we define 5 features for each of our road regions, each counting the amount of softmax output for the given class:

$$\phi_{ground}(t_k, y_i^j, y_{i-1}^j) = \mathcal{G}(t_k, y_{i+1}^j) - \mathcal{G}(t_k, y_i^j)$$

Note that via the integral accumulator the 3-way potential decomposes into pairwise terms $\mathcal{G}(t, y)$. In this case we only need 5 integral accumulators per ground image.

6.2.1.9 Ground alignment smoothness:

This potential encodes the fact that two consecutive alignments should be similar.

$$\phi_{gsm}(t_k, t_{k+1}) = |t_k - t_{k+1}|$$

This assumes that GPS+IMU have smooth errors and no outliers.

6.2.2 Inference via Block Coordinate Descent (BCD)

Inference in our model can be performed by minimizing the energy function:

$$\mathbf{y}^*, \mathbf{t}^* = \operatorname*{argmin}_{\mathbf{y},\mathbf{t}} E(\mathbf{y}, \mathbf{t}, \mathbf{x}_A, \mathbf{x}_M, \mathbf{x}_G)$$

Algorithm 4 Block coordinate descent inference (BCD).

```
1:  Set all alignments $\mathbf{t} = 0$, and initialize $\mathbf{y}$ by minimizing Eq. (53) ignoring the along road
    smoothness.
2:  repeat
3:      for for all $\mathbf{y}^j$ do
4:          Minimize Eq. (53) along the road w.r.t $\mathbf{y}^j$, holding the rest fixed.
5:      end for
6:      for all $\mathbf{y}_i$ at one segment of the road do
7:          Minimize Eq. (53) w.r.t $\mathbf{y}_i$, holding the rest fixed.
8:      end for
9:      for all $t$ variables do
10:         Minimize Eq. (53) w.r.t $\mathbf{t}$, holding $\mathbf{y}$ fixed.
11:     end for
12: until no energy reduction or max number iterations
```

with $E(\mathbf{y},\mathbf{t},\mathbf{x}_A,\mathbf{x}_M,\mathbf{x}_G)$ defined as in Eq. (53). Unfortunately, inference in our model is NP-hard, as our graphical model contains many loops. We thus take advantage of block coordinate descent to perform efficient inference. We refer the reader to Alg. 4 and Fig. 42 for inference steps.

Our Block Coordinate Descent algorithm alternates by doing inference in the direction along the road, doing inference in the direction perpendicular to the road and aligning the ground and aerial images. Note that when a road is not connected to a parallel road, the second step results in a graphical model with 15 variables, while when there are k parallel roads, this involves doing inference over a graphical model with $15k$ variables. Note also that in order to minimize the same objective, each of these iterations is performing conditional inference, and the pairwise potentials involving variables that are not optimized collapse to unaries.

6.2.3 Training with SSVM

We employ Structured Support Vector Machines (Tsochantaridis et al., 2005) to learn the weights of the aerial unaries and the smoothness in our model. In particular, we use the parallel cutting plane implementation of (Schwing et al., 2013). We employ a combination of two loss functions. The first is a truncated L_2 loss: $\ell_{\text{data}} = \min(\|y_i^j - \hat{y}_i^j\|^2, 100m^2)$, encouraging our prediction y_i^j to be close to the ground truth $\hat{y}_i^j$. We compute $\hat{y}_i^j$ by performing inference in our model with features computed from the ground truth annotation (segmentation). The second loss term encourages smoothness of the prediction along the road, $\ell_{sm} = |y_i^j - y_i^{j+1}|$. Note that the geometrical constraints in our model are either 0 or ∞ and are not trained.

6.3 Experimental Evaluation

We collected a new dataset which we call *Air-Ground-KITTI*, which is composed of both ground images from the KITTI tracking benchmark (Geiger et al., 2013) and newly acquired orthorectified aerial images over the same area. We neglected the KITTI sequences where the car is mostly static, resulting in 20 KITTI sequences for a total of 7603 ground stereo images. We annotated every 30th ground image with 4 semantic classes (*parking, sidewalk, road, building*). The aerial images were acquired by a DSLR camera mounted on an airplane and projected on the earth surface with 9 cm/pixel GSD. We split the data into 10 training and 10 test aerial image/KITTI sequences, with special care to avoid overlaps in the aerial images.

Model	*Average*		*Road*					*Sidewalk*				*Parking*			
	IoU	F1	IoU	F1	Pr.	R.	*EN*	IoU	F1	Pr.	R.	IoU	F1	Pr.	R.
(Mattyus et al., 2015)	-	-	62.1	76.4	68.0	87.0	-	-	-	-	-	-	-	-	-
(Mattyus et al., 2015) Deep Unary*	-	-	64.4	78.4	66.7	**94.7**	-	-	-	-	-	-	-	-	-
Lane	43.6	59.6	61.9	76.5	82.8	71.0	0.730	31.8	48.3	67.2	37.7	37.0	54.1	58.5	50.3
LaneParallel	44.8	60.3	66.5	79.9	**85.0**	75.4	**0.543**	31.6	48.0	**69.8**	36.6	36.1	53.1	**70.8**	42.4
LaneRoad	45.4	61.6	61.9	76.4	82.7	71.0	0.707	38.3	55.4	62.4	49.7	36.1	53.1	52.2	**54.1**
LaneRoadParallel	**48.6**	**64.3**	**68.0**	**80.9**	83.5	78.5	0.555	**39.5**	**56.6**	63.5	**51.1**	**38.4**	**55.5**	63.8	49.1
LaneRoadParallel**	41.9	58.5	54.9	70.9	**86.9**	59.9	0.559	**34.9**	**51.7**	**68.7**	**41.5**	**35.8**	**52.7**	**69.9**	**42.3**
Full**	**42.0**	**58.6**	**55.3**	**71.2**	86.8	**60.4**	**0.556**	**34.9**	**51.7**	**68.7**	**41.5**	**35.8**	**52.7**	**69.9**	**42.3**

Table 8: Performance for the semantic classes (i.e. *road, parking spot, sidewalk*) with various models and the two baselines. The values are in %, except *EN* which is the average road lane number l_1 error with respect to the oracle. * Marks the method of Mattyus et al. (2015) with our deep road classifier. The last two rows marked with ** evaluate only over areas where ground images are also available.

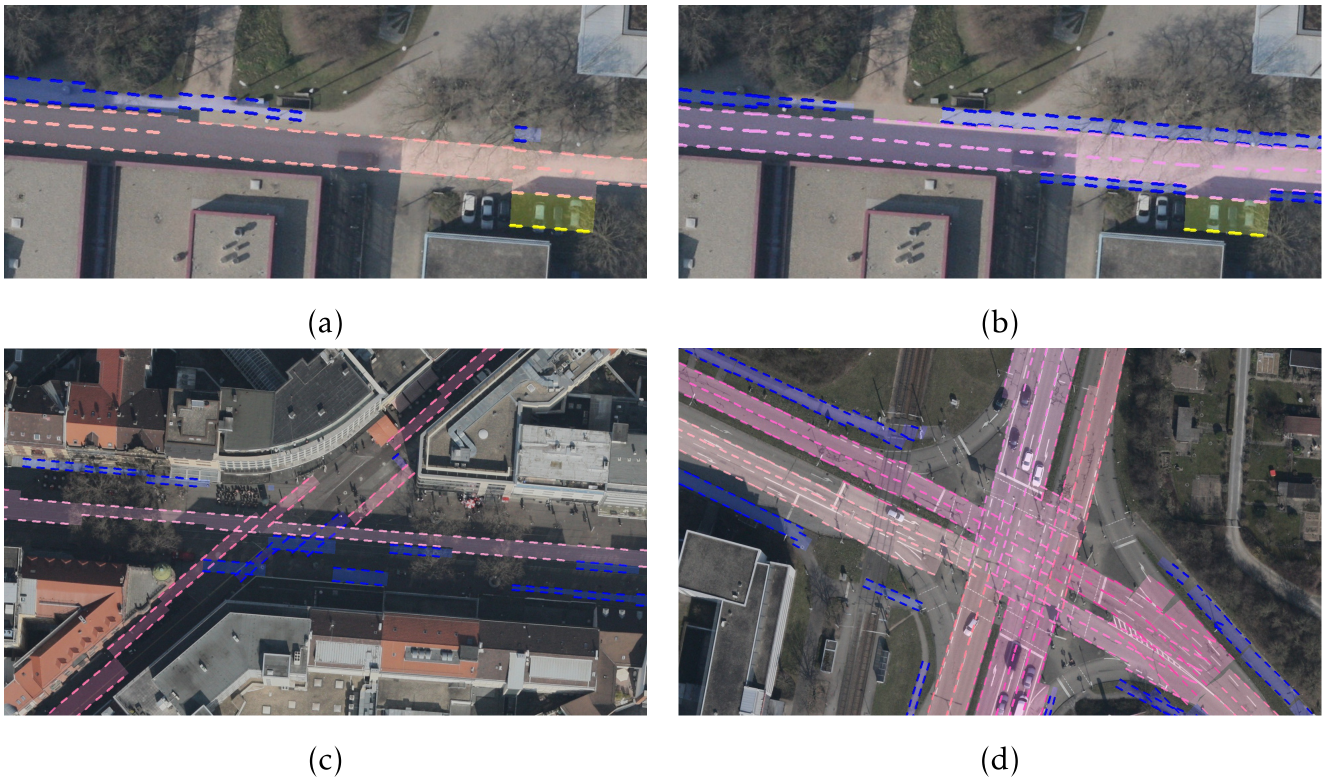


Fig. 43. It is hard to estimate the number of lanes if there are no lane markings. (a) Our method, (b) Oracle (i.e., our method with ground truth potentials). (c) Dense, urban pedestrian streets are difficult to estimate. (d) Our model is not intended for intersections, as it does not reason about turn lanes.

We manually annotated the aerial images with 4 categories (*parking*, *sidewalk*, *road*, *building*) as closed polygons and the lane markings as polylines. This took 70h of annotation, at a mean of $21h/km^2$, the area is 3.23 km^2.

To perform fine-grained segmentation using both aerial and ground images, we estimate a homography that transforms the ground plane in KITTI to the UTM coordinate system based on the KITTI's GPS+IMU measurements and the camera calibration. We assign each ground image to the closest parallel road segment. Our model then refines this estimate in the direction perpendicular to each road segment. We process every 5th ground image in the sequence.

As metrics for the fine-grained segmentation we calculate the pixelwise Intersection over Union (IoU), Precision, Recall and F1 metrics for three classes (i.e. road, parking, sidewalk). Note that we only measure the areas laying in the area of interest (i.e. ±15m around the road map centerline). We consider two parallel roads overlapping over the same area as a serious error. To reflect this, we handle these areas as if they were background. The metrics in Table 8 are calculated according to this.

For the roads, we additionally compute whether we have estimated the correct number of lanes. This is measured as the average ℓ_1 error in terms of number of lanes (*EN*). Note that if there are no lane markings, estimating the number of lanes is very difficult. Fig. 43 (a-b) shows this difficulty.

In our experiments, we compare our approach to the state-of-the-art method of (Mattyus et al., 2015), which uses OSMs to estimate road width. We also tested different model configurations for our approach. We refer to *Lane* as a model that employs *Aerial semantics, Aerial Edges, Road collapse constraints, Lane size constraint and Centerline prior* energy terms. Inference is done independently for each road segment via dynamic programming along the $\mathbf{y}^j = y_1^j, \cdots, y_{15}^j$ chains. We refer by *LaneParallel* to a model where we additionally include the constraint between nearby parallel road. We refer by *LaneRoad* as a model that

	GPS+IMU [m]	Ours [m]
Alignment error	1.67	**0.57**

Table 9. Ground to air image misalignment based on the camera calibrations (GPS+IMU) and after our alignment measured in meters. Using ±4 meter interval.

contains all the potentials in *Lane* plus smoothness along the road. We apply BCD inference by alternating between the chains perpendicular to the road (the lanes) and along the roads (segments). We refer by *LaneRoadParallel* a model that contains all potentials but the ground. Finally, *Ground* contains all potentials. We evaluate this case only where ground images are available.

6.3.1 Comparison to the state-of-the-art:

As shown in Table 8, our method outperforms (Mattyus et al., 2015) in almost all metrics, even when we apply our deep features instead of their road classifier in their method. Furthermore, we retrieve more semantic categories such as sidewalk, individual road lanes and parking. The constraint between parallel roads is important to achieve good results on roads. Without it, our model cannot outperform (Mattyus et al., 2015), which has this constraint.

6.3.2 Deep semantic features in aerial Images:

We show the performance of our Deep Network in Fig. 39. Note that it is much better than the road classifier of (Mattyus et al., 2015).

6.3.3 Alignment between aerial and ground images:

As shown in Table 9 and Fig. 38 reasoning about the alignment between ground and aerial images while doing fine-grained segmentation improves the alignment significantly.

6.3.4 Qualitative Results:

We visualize our results when using only aerial images in Fig. 40, and when using joint aerial and ground reasoning in Fig. 41. Our approach is able to estimate well the lanes, sidewalk and parking as well as the alignment between the ground and the aerial images.

6.3.5 Ablation studies:

As shown in Table 8, the metrics for different versions of our model are fairly similar, however qualitatively, as we add more potentials, the results get better. This is illustrated in Fig. 43 (c), where the *OnlyLane* model moves the middle road to a parallel road resulting in a noncontinuous structure. In contrast, the *LaneRoadParallel* model prevents overlaps and favors smoothness, see the Fig. 43 (d). Including the ground images only slightly improves performance. We believe this could be overcome by using stronger features in the ground images, i.e., leveraging the full 3D point cloud, not just the ground plane. Note that since our approach gives us very precise alignments between the ground and the aerial images it could be used to enhance OSM with object locations, e.g. traffic signs.

6.3.6 Inference time:

Inference in our full model takes 6 seconds per km of road, with a single thread on a laptop computer. Note that BCD can easily be parallelized.

6.3.7 Limitations:

Our model is designed for individual roads and it does not reason about turning lanes connecting different roads at intersections (see Fig. 43 (d)). Dealing with such scenarios is part of our future work. Semantic segmentation from aerial images reasons mainly about the visible parts of the street. Therefore covered areas (e.g. by building, bridges, trees) can be a problem. However, when ground images are available, our approach can handle this problem. Our aerial images were acquired in early spring, and thus trees occluding the roads is not a big problem.

7 Conclusion and Outlook

7.1 Summary

This thesis presents four novel approaches to augment information of road maps, aerial and ground images: Vehicles are detected in aerial images. The geolocation of aerial images is estimated using a road map. Aerial images are parsed to enhance the road maps by estimating the road width and correcting misalignments. Aerial and ground images are processed jointly with existing maps to extract the fine-grained road layout and to improve the alignment between the different image types.

The proposed vehicle detection method applies Integral Channel Features in a Soft Cascade structure for a binary detector distinguishing vehicles and background. These allows to employ various expressive features resulting in good detection performance while the computation remains still fast, i.e. 69% recall with 86% precision is achieved on the *DLR 3K Munich dataset* of 21 Megapixel images and the computation takes only 4.4 seconds on a single CPU core. In contrast to other fast methods, this approach does not need a prior about the streets in the images and hence it can work on original frame images without any orthorectification and geolocation. This is a practical property, since it allows the detection of the vehicles outside of the known roads or, as shown by this thesis, the tracks of the detected vehicles can be leveraged to geolocalize aerial images. Aggregating classifiers to handle different orientations of the vehicles delivers better performance and since the computed features can be reused the computation increase is sublinear by the number of extra classifiers, i.e. 2.7 s is needed for a single classifier and 4.4 s for 8 classifiers. The detected vehicles can be further classified by their type. A small neural network processing HOG features can distinguish between cars and trucks with 98% accuracy in the *DLR 3K Munich dataset* containing several thousand vehicles. A similar neural network can estimate the orientation of the vehicles with 84% correctness on the same data. The proposed detector delivering both robust performance with limited computational demand is suitable for being applied in real-time airborne road traffic extraction system.

Moving cars can be reliably detected and tracked in aerial image sequences, because vehicle detectors (e.g. the detector presented in this thesis) can deliver decent performance and the motion information is a strong a cue enabling the rejection of false positives. By assuming that cars drive on roads, the car trajectories can be considered an incomplete representation of the road network and features can be calculated, e.g. the distances between roads. The presented results suggest that these road features tend to be so unique that they can be applied to recognize a certain spot in the road network (defined by the road map) and thus function as geolocalization. The proposed Polyline Based Geometric Hashing (PLBGHashing) method is selective for the distances between different roads, while being invariant to rotation and translation, it is able to handle the ambiguity of the assignment between car trajectories and the line segments of roads, it can handle missing line segments during query and it allows fast search also over a larger area. In the conducted experiments, the geometric hashing based search could rank the correct location in the *top 100* 13 times out of 20 over a search area of 22500 km^2 with 32000 km road length. The shortlist provided by the PLBGHashing can be verified by robust chamfer matching between the road map and the extracted vehicle trajectories. As a result the correct location is ranked as the *top 1* 15 times out of 20 on the search area of 22500 km^2. On a smaller, 49 km^2, search area the location is found 18 times correctly.

Enhancing the maps with the road width information by parsing aerial images needs pixel level analysis of the images and the possible misalignments between the map and the aerial images have to be handled as well. Considering the problem as semantic segmentation would need to reason about many variables and the priors typical for roads (e.g. smoothness along the road, straight lines) could be expressed only by higher order potentials leading to complex inference problems. The presented approach employs the map data to formulate the problem as one of inference in a MRF reasoning directly about the width of the road and the alignment between the imagery and the map. This approach preserves the topology of the road inherently, the data terms of the MRF can be calculated efficiently via non axis-aligned integral images and the smoothness of the roads can be expressed as a pairwise term, as well as the constraint that two parallel roads should not overlap. The graph of one road in this formulation is a chain enabling efficient and exact inference via dynamic programming. In case of parallel roads, fast BCD approximate inference is performed. The method can achieve 72% IoU in a challenging dataset, while the inference only takes 0.1 second per km of road. Importantly, the parameters of the MRF are trained efficiently by SSVMs and the learned model can generalize well to the whole world.

The fine-grained road layout is extracted in form of the number and width of road lanes, as well as the presence and size of sidewalks and parking spots by analyzing aerial and ground images and leveraging existing road maps defining the road centerline. The problem is formulated as one of inference in an MRF reasoning about the road layout as well as the alignment between the aerial image, the map and the ground image sequence in a joint energy function. The road layout is defined by the distances of the boundaries between lanes. This defines a chain perpendicular to the road for each segment of the road. The semantic information of the images is extracted by a deep neural network providing pixelwise predictions for 5 classes: road, sidewalk, parking spot, building and background. These pixelwise predictions are summed inside the lanes. The lane boundaries are expected to be aligned with edges in the image, therefore an edge detector is applied and formulated as unary term. The prior knowledge about the width of the lanes and the constraints of the road layout are expressed as pairwise terms enabling robust estimation also in case of missing visual evidence (missing lane markings). The smoothness of the lanes along the road is expressed also by pairwise terms. The inference problem is solved by BCD along chains perpendicular and parallel to the roads as well as in the ground image sequence. The approach is evaluated on a dataset augmenting KITTI (Geiger et al., 2012) by annotated aerial images: the IoU metric is 68% for the road lanes and 39%, 38% for the challenging sidewalk and parking spot categories respectively. When ground images are applied jointly with aerial images the MRF reasons also about the alignment between the two perspectives. The image evidence in the two imagery for the road layout is aligned by a translation perpendicular to the road. Instead of registering the images directly, they are both aligned to a joint road layout. This method is robust to visual appearance differences reducing the misalignment between the two imagery to 0.6 meters, significantly lower than the 1.7 m provided by sophisticated GPS+IMU systems. This allows the application of the detailed road layout later for precise, lane-wise visual self-localization within the road.

7.2 Improvements and future work

The individual methods could be enhanced by various improvements, for instance:

◇ The performance of the vehicle detector could be further improved by using a deep neural network after the binary detector like R-CNN in (Girshick et al., 2014). Since this has to be applied only to a fraction of the image, the speed of the detector would be still fast. Alternatively, a complexity aware cascade could be created as in (Cai et al., 2015) which could achieve higher performance with similar or even faster speed than simpler detectors.

◇ The presented method to geolocalize aerial images could be extended to address more general scenarios. A road detector could be applied to enable localization also in the case of missing traffic. The matching could be extended to additional object categories which are covered by maps as well, e.g. building, rivers, forests, etc. The 3D structure extracted from the images could be also applied as a future, especially in areas with considerable relief.
◇ The fine-grained road layout parsing could be extended with more detailed categories, e.g. building footprints, driving direction on the road, etc., extend the method to reason about the turning lanes at intersections and add additional objects visible only in the ground images, e.g. traffic signs, traffic lights.

For the creation and maintenance of detailed, up-to-date maps multiple image sensors are available from satellite images, through aerial and UAV images to imagery acquired by vehicles or smart phones. These sensors have different spatial resolution, spatial coverage, acquisition frequency (revisit time), different perspective and geometric accuracy, but they can be applied jointly for improving the maps. A crucial question is the precise registration of these different sensors. As presented by this thesis, detailed maps are not only the goal of analyzing diverse images, but they can also serve as the feature enabling the registration of the different sensor data.

In the current standard approaches the main image processing steps are handled independently, e.g. stereo matching, orthorectification, semantic analysis. However, these tasks are correlated, for instance building segmentation and stereo matching in aerial images. A system solving these problems jointly, in a holistic approach, would probably deliver better results than systems tackling the different problems independently. Creating an efficient holistic system solving the relevant image analysis tasks by using different sensor inputs and maps jointly will be an important goal and challenge (probably for a longer time) in both the remote sensing and the computer vision community.

List of Abbreviations

ACF Aggregated Channel Feature
ADAS Advanced Driver Assistance Systems
BCD Block Coordinate Descent
CNN Convolutional Neural Network
CPU Central Processing Unit
CRF Conditional Random Field
DARPA Defense Advanced Research Projects Agency
DPM Deformable Part Model
DSLR Digital Single-Lens Reflex
DSM Digital Surface Model
GCP Ground Control Point
GIS Geographic Information System
GNSS Global Navigation Satellite System
GPS Global Positioning System
GPU Graphics Processing Unit
GSD Ground Sampling Distance
HMM Hidden Markov Model
HOG Histogram of Oriented Gradients
ICF Integral Channel Feature
IID *independent and identically distributed*
IMU Inertial Measurement Unit
IoU Intersection over Union
ISPRS International Society for Photogrammetry and Remote Sensing
LIDAR Light Detection And Ranging
MAP Maximum a prosteriori
MRF Markov Random Field
OSM OpenStreetMap
PCL Point Cloud
PLBGHashing Polyline Based Geometric Hashing
PR Precision-Recall
ReLu Rectified Linear Unit
ROI Region of Interest
SIFT Scale-Invariant Feature Transform
SRTM Shuttle Radar Topography Mission
SSVM Structured Support Vector Machine
SVM Support Vector Machine
UAV Unmanned Aerial Vehicle
USGS United States Geological Survey
UTM Universal Transverse Mercator
WGS84 World Geodetic System 1984

List of Figures

1 Illustration of the objectives 1 and 2. The detected vehicles are shown with colored bounding boxes, different classes with different color. Black shows false-negative (missing) cars. The bottom image shows aerial images localized over a larger area. The image is transformed into the road map. The red color in the image shows the extracted car tracks used to match to the map. The red dots in the Google Earth image shows the location of the image. 4

2 Illustration of the objectives 3 and 4. The extracted road width is shown by yellow color. The two bottom image shows the road layout extracted jointly in the aerial and the ground (bottom) image. The coloring is pink: road, yellow: parking lane, blue: sidewalk. The detected road from the ground is projected into the aerial image and it is shown by red color. 6

3 An illustration of an SVM when the data is linearly separable (a). If the data is not linearly separable a trade-off is needed between the size of the margin and the mislabeling of the training data (b). 12

4 An illustration of a binary classification tree predicting if the user of a website will click on an advertisement or not. The leaves (squares) are the class label output, the nodes (circles) are the binary decisions over an input feature. 13

5 Common non-linear functions for neural networks. Sigmoid, hyperbolic tangent and rectified linear unit. 15

6 Illustration of the forward pass (prediction) of a neural network with one hidden layer. 16

7 Illustration of the backward pass of a neural network with one hidden layer. The backpropagation of the Loss function gradient is shown in orange, the gradients used for the parameter update are shown in red and the forward pass is in black. The regularization term $\mathcal{R}(\mathbf{w}, \mathbf{b})$ is not shown. View this figure rotated. 17

8 Illustration of a CNN where the first convolution layer has a kernel size of 5 and the second 3. The max pooling layers have a receptive field of 2. The convolution is padded with zeros. The edges in one convolution layer with the same dash type have the same weights. As we move forward in the network the spatial resolution decreases due to the max pooling. 18

9 From (Zeiler and Fergus, 2014). Visualizing different layers of a deep convolutional neural network. The left images show reconstructed patterns which cause high activations in a feature map. The right images show the corresponding image patches. The reconstructed patterns are projected down to pixel space using the deconvolutional network approach of (Zeiler and Fergus, 2014). The first layers extract simpler features (e.g. edges, corners, etc.), while the later layers describe complex shapes with greater variance. 19

10 A MRF with 3 variables defined by a graph (a) and two possible factorizations (b-c). The first (b) defines pairwise potentials $\Psi(y_i, y_j), i \neq j$. The second (c) defines one third order potential $\Psi(y_1, y_2, y_3)$. 21

11 The random variables y_1, y_2 are conditioned on x_1, x_2 in a CRF (a). The potentials $\Psi(y_1, x_1)$ and $\Psi(y_2, x_2)$ in (b) can be simplified to unaries $\Psi(y_1), \Psi(y_2)$ (c). 21

12 Trellis diagram illustration of the MAP inference in a chain with 5 variables each having 4 states. The $\mathbf{y}^* = \text{argmin}_y(E(\mathbf{y}))$ state sequence is the shortest path if the distance is the sum of the unary potentials visited and the pairwise potentials (edges) connecting the variables. The shortest path (i.e. lowest energy) is shown in red. 24

13 The graph is shown in (a). An iteration for updating a row of variables is shown in (b), while (c) depicts an inference iteration in a column. The gray circles show variables which are kept fix. The dashed lines show the pairwise potentials folded to unaries. 25

14 From (Weyand et al., 2016). Worldwide image based localization. At the top the query images are shown. The bottom map shows the ground truth position (yellow), the human location guess (green) and in blue the automatic guess of (Weyand et al., 2016). 31

15 From (Brubaker et al., 2013). Examples for localizing a car by visual odometry and a map. "The left most column shows the full map region for each sequence, followed by zoomed in sections of the map showing the posterior distribution over time. The black line is the GPS trajectory and the concentric circles indicate the current GPS position. Grid lines are every 500m. " 32

16 From (Ren et al., 2015a). Object detection outputs of the method. The bounding boxes of different object classes are shown in different colors, with the class name and a confidence score. 34

17 From (Chen et al., 2015). In each column the left image is the input, the middle image shows the labels produced by the fully convolutional neural network and the right image shows the output of the fully connected CRF. Different colors represent different classes. Note that the coarse output of the deep network if refined by the CRF. 35

18 Proposed vehicle detection framework. The input image is first evaluated by the multi-direction vehicle detector. A sliding window goes along x- and y-axes. Features are extracted from the detection window and sent to trained binary classifier. The binary classifier classify whether current detection window contains a positive object or not. Detected vehicles are then processed for estimating their orientations and categories. 37

19 The training samples for the aggregated classifier method are split into multiple groups. 40

20 (a) Evaluation of the Integral Channel Features. Gradient histogram channels play the most important role while gradient magnitude channel has least affects on the final result. (b) Detection result of aggregated detectors. (c) Performance after rescaling the image with different factors. (d) Orientation estimation error histogram using artificial neural network with 16 output classes. 41

21 Detection results from the DLR test images. Green and cyan bounding boxes are the correct detected samples, representing cars and trucks, respectively. Black bounding boxes are the missed ones and red are the false positives. The results show that our method works well in most scenarios (a)(b)(c), however the complicated rooftops or outdoor swimming pools may lead to false positive detections (d). We also evaluated our method on the dataset presented in Moranduzzo and Melgani (2014a), Moranduzzo and Melgani (2014b), the detection results are shown in (e)(f). 42

22 Illustration. We search the geolocation of aerial image sequences. The red dots on the map show the ground truth positions. The photos can be located by matching the road traffic (in red) in the image to the road network (in black). The matching tracks are marked with yellow circles in the images. The search area can be as large the shown map. The accuracy of the location is around 25 m. 46

23 (a): The blue and green polylines are different roads. The x-y is the original coordinate system of the road database. The line segment p1-p2 defines the basis (u,v) with the center point and the direction. The grid points intersected by line segments are the indices for the hash table. (b): The p points are the road vertices (from the database), the t points are the track vertices extracted in the mosaic image (with different color for different cars). The red arrows are the road bases in the hash table. The basis B0 from the track end points t1-t3 is shifted along the track. At B+2 position the track basis lies in the the same grid row, as the road basis. 50

24 The distance images used for calculating the verification score at *Suburban 1* image sequence. On top the tracks are shown with green (inlier) and red (outlier) in the distance image to the road network. In the bottom the road network (green) is shown in the distance image to classified road pixels in the image. 52

25 Completeness over top N. 53

26 The largest search area for each image sequence where the correct verified geolocation was in the TOPN in km^2 54

27 Geolocalization results. The mosaic images are projected into the street network. The roads are in black, the tracks are highlighted with red lines, the locations of votes from the *PLBGHashing* are marked with yellow circles. (When an image scene is only partially overlapped with a model during the shortlist retrieval, there are no vote circles at every matching track-road locations (e.g. on *Urban 1.*)) On (d) the geolocalized mosaic image with the tracks is overlaid in Google Earth. All these scenes were correctly located on a 22500 km^2 search area except (f). The (f) shows the scene *Industrial 1* located false on a 4 km^2 search area. The causes for this are; the scene contains 2 roads which makes it inherently ambiguous, multiple tracks are extracted on a parking lot, outside of the road network. Since there are more tracks in the parking lot than on the road, the verification matching handles the correct tracks as outliers. The color based road detection also does not work properly, since a parking lot has similar color as the road. The error is approximately 100m. 55

28 Road segmentation is challenging due to shadows, occluding trees and vehicles which make the appearance heterogeneous as well as OSM/projection misalignment errors. 57

29 Illustration of the road centerline with the width parameterized by the center offset h and symmetrical width y. The direction and length of the rectangle Ω_i is defined by the p_i, p_{i-1} points given by the street database. The context is depicted as Σ. 58

30 Segmentation results on several cities over the world using the edge, homogeneity, context, car and overlap features. Note that the MRF was trained only on 1.5 km^2 imagery from the Aerial KITTI dataset. 62

31 The factor graph of the proposed graphical model in case of a single road. The unary potentials are pairwise defining loops in the graph (a). By merging the variables $g_i^j = (h_i^j, y_i^j)$ the graph can be reduced to a chain (b). 63

32 Illustration of our BCD inference. Note that $g_i^j = (h_i^j, y_i^j)$. (a) Graphical model consisting of 3 roads that have overlapping constraints (i.e., vertical edges). We alternate between performing inference (b) over each road one at a time (red, green, blue), and (c) along chains on the vertical direction encoding the horizontal constraints, also one at a time (orange, yellow, pink, green, purple). Note that these operations involve conditioning, and thus the pairwise potentials between a variable in the chain and a connected variable not in the chain are folded as unaries. 64

33 Failure modes: (a) Missing turn lane intersection. (b) The extracted road is too narrow. (c) Road covered by trees. 67

34 Results on Bavaria and Aerial KITTI. 67

35 (Top) Our road extracted from aerial images (green) projected into Kitti ground images. (2nd row): Geographical unary of (Wang et al., 2015a). (3rd row): Geographical unary with our road estimate. (Bottom) Ground truth. Road (pink), sidewalk (blue), building (red), car (purple). 68

36 Comparison to (Yuan and Cheriyadat, 2013): Our approach works significantly better than the baseline. 68

37 **Illustration of our model:** (a) Parameterization of our approach. Our random variables are the absolute location of the different region boundaries (e.g., sidewalk) as well as the alignment between air and ground. (b) Our formulation allows a random variable to take the same state as the previous node, collapsing a region to have 0 width. (c). For each ground-view image, a random variable models the alignment noise. (d). Projection of our parameterization on the ground-view. 71

38 **Effect of reasoning about alignment: (left)** alignment given by GPS+IMU, **(right)** alignment inferred by our model. **(top)** Ground road classifier projected into the aerial image (shown in red). **(bottom)** Our semantic classes projected on the ground image. Our joint reasoning significantly improves alignment. 72

39 Precision-Recall curves for our deep classifier and the road classifier of (Mattyus et al., 2015) marked with * and in dashed. 73

40 Visualization of our semantic road parsing results using only aerial images. The road lanes are shown with shades of pink, the sidewalk with blue and the parking spots with yellow. 74

41 Left: The ground road detection with red projected into the aerial image after alignment and road layout estimation. Right: The semantic lanes projected back into the aligned ground image. These scenes are all challenging with parallel roads, parking spots and intersections. The bottom image is especially difficult since it is an urban pedestrian area. Note that the aerial and ground images were taken with several years difference in different seasons. Pink is road, blue is sidewalk and yellow marks parking spots. 74

42 **BCD:** The graph shows a simplified network with two parallel roads (each with 3 random variables) and one ground image per segment connected to the right road. BCD alternates between three types of updates. (a) Along the road updates: we optimize over each chain with the same color (while holding all other variables fix). The pairwise terms fold to unaries (see dashed black lines). (b) Perpendicular to the road updates: we do inference for the nodes with the same color (holding the rest fix). (c) Along the ground alignments: We minimize only the t variables which are depicted in green. The y variables are fixed and are depicted in black. 75

43 It is hard to estimate the number of lanes if there are no lane markings. (a) Our method, (b) Oracle (i.e., our method with ground truth potentials). (c) Dense, urban pedestrian streets are difficult to estimate. (d) Our model is not intended for intersections, as it does not reason about turn lanes. 78

List of Tables

1 Performance comparison between different methods. The UAV dataset is from Moranduzzo and Melgani (2014a,b) 44

2 Confusion matrices of type classification using different cropping configurations. *a* : Actual class / Predicted class. *b* : The number of truck type is increased by random transformation of the existing samples. 44

3 Comparison of computation times. *Running on the GPU. 45

4 Performance of our method vs baselines. The IoU and F1 values are in %, while $\overline{\Delta h}, \overline{\Delta y}$ are the mean absolute error of the offset and width measured in meters. * the road unary of (Winn et al., 2005) is applied. ** is the method of (Yuan and Cheriyadat, 2013). 59

5 Performance on Bavaria and Aerial KITTI with various features configurations. The IoU and F1 values are in %, while $\overline{\Delta h}, \overline{\Delta y}$ are the mean absolute error of the offset and width measured in meters. The || symbol denotes the overlap potential between parallel roads. The other abbreviations are *E*: edge feature, *Ho*: homogeneity feature, *Co*: context feature. 60

6 Running time for feature accumulator calculation and inference under various configurations. In sec per km of road. 66

7 Our method improves the geographic priors of (Wang et al., 2015a). All values are IoU in %. 68

8 Performance for the semantic classes (i.e. *road, parking spot, sidewalk*) with various models and the two baselines. The values are in %, except *EN* which is the average road lane number l_1 error with respect to the oracle. * Marks the method of Mattyus et al. (2015) with our deep road classifier. The last two rows marked with ** evaluate only over areas where ground images are also available. 77

9 Ground to air image misalignment based on the camera calibrations (GPS+IMU) and after our alignment measured in meters. Using ±4 meter interval. 79

References

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Susstrunk, S., 2012. Slic superpixels compared to state-of-the-art superpixel methods. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) .

Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B. C., Chen, J., Chrzanowski, M., Coates, A., Diamos, G., Elsen, E., Engel, J., Fan, L., Fougner, C., Han, T., Hannun, A. Y., Jun, B., LeGresley, P., Lin, L., Narang, S., Ng, A. Y., Ozair, S., Prenger, R., Raiman, J., Satheesh, S., Seetapun, D., Sengupta, S., Wang, Y., Wang, Z., Wang, C., Xiao, B., Yogatama, D., Zhan, J., Zhu, Z., 2015. Deep speech 2: End-to-end speech recognition in english and mandarin. CoRR abs/1512.02595.

Barron, A. R., 1993. Universal approximation bounds for superpositions of a sigmoidal function. IEEE Transactions on Information Theory 39 (3): 930–945.

Barrow, H. G., Tenenbaum, J. M., Bolles, R. C., Wolf, H. C., 1977. Parametric correspondence and chamfer matching: two new techniques for image matching. In: Proceedings of the 5th international joint conference on Artificial intelligence - Volume 2, IJCAI'77, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 659–663.

Barzohar, M., Cooper, D., 1996. Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) .

Bishop, C. M., 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Borgefors, G., 1986. Distance transformations in digital images. Comput. Vision Graph. Image Process. 34 (3): 344–371.

Boser, B. E., Guyon, I. M., Vapnik, V. N., 1992. A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92, ACM, New York, NY, USA, 144–152.

Bourdev, L., Brandt, J., 2005. Robust object detection via soft cascade. In: Conference on Computer Vision and Pattern Recognition (CVPR), 2005., Vol. 2, IEEE, 236–243.

Bramer, M., 2007. Principles of Data Mining (Undergraduate Topics in Computer Science). Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Breiman, L., 2001. Random forests. Machine learning 45 (1): 5–32.

Brubaker, M. A., Geiger, A., Urtasun, R., 2013. Lost! leveraging the crowd for probabilistic visual self-localization. In: Conference on Computer Vision and Pattern Recognition (CVPR), 2013, 3057–3064.

Cai, Z., Saberian, M., Vasconcelos, N., 2015. Learning complexity-aware cascades for deep pedestrian detection. In: International Conference on Computer Vision (ICCV) 2015.

Canny, J., 1986. A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 8 (6): 679–698.

Chai, D., Forstner, W., Lafarge, F., 2013. Recovering line-networks in images by junction-point processes. In: Conference on Computer Vision and Pattern Recognition (CVPR), 2013.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A. L., 2015. Semantic image segmentation with deep convolutional nets and fully connected crfs. In: International Conference on Learning Representations (ICLR) 2015.

Chen, Q., Koltun, V., 2014. Fast mrf optimization with application to depth reconstruction. In: Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

Chen, X., Xiang, S., Liu, C., Pan, C., 2014. Vehicle detection in satellite images by hybrid deep convolutional neural networks. Geoscience and Remote Sensing Letters, IEEE 11 (10): 1797–1801.

Cortes, C., Vapnik, V., 1995. Support-vector networks. Machine Learning 20 (3): 273–297.

Dahl, G. E., 2015. Deep learning approaches to problems in speech recognition, computational chemistry, and natural language text processing. Ph.D. thesis, University of Toronto.

Dalal, N., Triggs, B., 2005. Histograms of oriented gradients for human detection. In: Conference on Computer Vision and Pattern Recognition (CVPR), 2005, Vol. 1, IEEE, 886–893.

Dollár, P., Appel, R., Belongie, S., Perona, P., 2014. Fast feature pyramids for object detection. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) .

Dollár, P., Tu, Z., Perona, P., Belongie, S., 2009. Integral channel features. In: British Machine Vision Conference (BMVC), 2009., Vol. 2, 5.

Dollár, P., Zitnick, C. L., 2013. Structured forests for fast edge detection. In: International Conference on Computer Vision (ICCV), 2013.

Dollár, P., Zitnick, C. L., 2015. Fast edge detection using structured forests. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) .

Duan, K.-B., Keerthi, S. S., 2005. Multiple Classifier Systems: 6th International Workshop, MCS 2005, Seaside, CA, USA, June 13-15, 2005. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, Ch. Which Is the Best Multiclass SVM Method? An Empirical Study, 278–285.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., Zisserman, A., 2010. The pascal visual object classes (voc) challenge. International Journal of Computer Vision (IJCV) 88 (2): 303–338.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., Ramanan, D., 2010. Object detection with discriminatively trained part based models. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 32 (9): 1627–1645.

Felzenszwalb, P. F., Huttenlocher, D. P., 2004. Efficient graph-based image segmentation. International Journal of Computer Vision (IJCV) .

Felzenszwalb, P. F., Huttenlocher, D. P., 2012. Distance transforms of sampled functions. Theory of Computing 8 (19): 415–428.

Forster, C., Pizzoli, M., Scaramuzza, D., 2013. Air-ground localization and map augmentation using monocular dense reconstruction. In: IROS.

Freund, Y., Schapire, R. E., 1996. Experiments with a new boosting algorithm.

Frey, B. J., MacKay, D. J. C., 1997. A revolution: Belief propagation in graphs with cycles. In: Conference on Neural Information Processing Systems (NIPS), 1997, MIT Press, 479–485.

Friedman, J., Hastie, T., Tibshirani, R., 1998. Additive logistic regression: a statistical view of boosting. Annals of Statistics 28: 2000.

Geiger, A., Lenz, P., Stiller, C., Urtasun, R., 2013. Vision meets robotics: The kitti dataset. IJRR .

Geiger, A., Lenz, P., Urtasun, R., 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

Gkioxari, G., Girshick, R., Malik, J., 2015. Contextual action recognition with r*cnn. In: International Conference on Computer Vision (ICCV), 2015.

Glorot, X., Bordes, A., Bengio, Y., 2011. Deep sparse rectifier neural networks. In: Gordon, G. J., Dunson, D. B. (Hrsg.), Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11), Vol. 15, Journal of Machine Learning Research - Workshop and Conference Proceedings, 315–323.

Gros, P., Bournez, O., Boyer, E., 1998. Using local planar geometric invariants to match and model images of line segments. Computer Vision and Image Understanding 69 (2): 135 – 155.

Hancock, T., Jiang, T., Li, M., Tromp, J., 1996. Lower bounds on learning decision lists and trees. Information and Computation 126 (2): 114 – 122.

Hannun, A. Y., Case, C., Casper, J., Catanzaro, B. C., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., Ng, A. Y., 2014. Deep speech: Scaling up end-to-end speech recognition. CoRR abs/1412.5567.

Hartley, R. I., Zisserman, A., 2004. Multiple View Geometry in Computer Vision, 2nd Edition. Cambridge University Press, ISBN: 0521540518.

Hays, J., Efros, A. A., 2008. Im2gps: estimating geographic information from a single image. In: Conference on Computer Vision and Pattern Recognition (CVPR), 2008., 1–8.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Deep residual learning for image recognition. CoRR abs/1512.03385.

Håstad, J., 1989. Almost optimal lower bounds for small depth circuits. In: RANDOMNESS AND COMPUTATION, JAI Press, 6–20.

Hyafil, L., Rivest, R. L., 1976. Constructing optimal binary decision trees is np-complete. Information Processing Letters 5 (1): 15 – 17.

Ian, G., Yoshua, B., Aaron, C., 2016. Deep learning, book in preparation for MIT Press.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., Saul, L. K., 1999. An introduction to variational methods for graphical models. Machine Learning 37 (2): 183–233.

Kalogerakis, E., Vesselova, O., Hays, J., Efros, A. A., Hertzmann, A., 2009. Image sequence geolocation with human travel priors. In: International Conference on Computer Vision (ICCV), 2009.

Kluckner, S., Pacher, G., Grabner, H., Bischof, H., Bauer, J., 2007. A 3d teacher for car detection in aerial images. In: International Conference on Computer Vision (ICCV), 2007, 1–8.

Kolmogorov, V., Zabin, R., 2004. What energy functions can be minimized via graph cuts? IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 26 (2): 147–159.

Kozempel, K., Reulke, R., 2009. Camera orientation based on matching road networks. In: Image and Vision Computing New Zealand, 2009. IVCNZ '09. 24th International Conference, 237–242.

Krähenbühl, P., Koltun, V., 2011. Efficient inference in fully connected crfs with gaussian edge potentials. In: Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., Weinberger, K. Q. (Hrsg.), Conference on Neural Information Processing Systems (NIPS), 2011. Curran Associates, Inc., 109–117.

Kraus, K., Harley, I., Kyle, S., 2007. Photogrammetry: Geometry from Images and Laser Scans. No. Bd. 1 in De Gruyter textbook. Bod Third Party Titles.

Krizhevsky, A., Sutskever, I., Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C. J. C., Bottou, L., Weinberger, K. Q. (Hrsg.), Conference on Neural Information Processing Systems (NIPS), 2012. Curran Associates, Inc., 1097–1105.

Lampert, C. H., Blaschko, M. B., Hofmann, T., 2009. Efficient subwindow search: A branch and bound framework for object localization. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 31 (12): 2129–2142.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521: 436–444.

Lecun, Y., Bottou, L., Orr, G. B., Müller, K. R., 1998. Efficient BackProp. In: Orr, G., Müller, K. (Hrsg.), Neural Networks—Tricks of the Trade. Vol. 1524 of Lecture Notes in Computer Science. Springer Verlag, 5–50.

Leitloff, J., Hinz, S., Stilla, U., 2010. Vehicle detection in very high resolution satellite images of city areas. IEEE Transactions on Geoscience and Remote Sensing 48 (7): 2795–2806.

Leitloff, J., Rosenbaum, D., Kurz, F., Meynberg, O., Reinartz, P., 2014. An operational system for estimating road traffic information from aerial images. Remote Sensing 6 (11): 11315–11341.

Li, S., Kittler, J., Petrou, M., 1992. Matching and recognition of road networks from aerial images. In: Sandini, G. (Hrsg.), European Conference on Computer Vision (ECCV), 1992. Vol. 588 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, 857–861.

Li, Y., Paluri, M., Rehg, J. M., Dollár, P., 2016. Unsupervised learning of edges. Conference on Computer Vision and Pattern Recognition (CVPR), 2016. .

Lin, T.-Y., Belongie, S., Hays, J., 2013. Cross-view image geolocalization. In: Conference on Computer Vision and Pattern Recognition (CVPR), 2013., 891–898.

Lin, T.-Y., Cui, Y., Belongie, S., Hays, J., 2015. Learning deep representations for ground-to-aerial geolocalization. In: Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

Lin, Y., Medioni, G., 2007. Map-enhanced uav image sequence registration and synchronization of multiple image sequences. In: Conference on Computer Vision and Pattern Recognition (CVPR), 2007., 1–7.

Liu, C., Schwing, A., Urtasun, R., Filder, S., 2015. Rent3d: Floor-plan priors for monocular layout estimation. Conference on Computer Vision and Pattern Recognition (CVPR), 2015. .

Liu, K., Mattyus, G., 2015. Fast multiclass vehicle detection on aerial images. IEEE Geoscience and Remote Sensing Letters .

Liu, K., Skibbe, H., Schmidt, T., Blein, T., Palme, K., Brox, T., Ronneberger, O., 2014. Rotation-invariant hog descriptors using fourier analysis in polar and spherical coordinates. International Journal of Computer Vision (IJCV) 106 (3): 342–364.

Lowe, D. G., 2004. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision (IJCV) 60 (2): 91–110.

MacKay, D. J. C., 2002. Information Theory, Inference & Learning Algorithms. Cambridge University Press, New York, NY, USA.

Mattyus, G., Wang, S., Fidler, S., Urtasun, R., 2015. Enhancing road maps by parsing aerial images around the world. In: International Conference on Computer Vision (ICCV), 2015.

Mattyus, G., Wang, S., Fidler, S., Urtasun, R., 2016. Hd maps: Fine-grained road segmentation by parsing ground and aerial images. In: Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

Matzen, K., Snavely, N., 2013. Nyc3dcars: A dataset of 3d vehicles in geographic context. In: International Conference on Computer Vision (ICCV), 2013.

Mayer, H., Hinz, S., Bacher, U., Baltsavias, E., 2006. A test of automatic road extraction approaches. In: In International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 209–214.

Mnih, V., Hinton, G. E., 2010. Learning to detect roads in high-resolution aerial images. In: European Conference on Computer Vision (ECCV), 2010.

Mnih, V., Hinton, G. E., 2012. Learning to label aerial images from noisy data. In: International Conference on Machine Learning (ICML), 2012., 567–574.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540): 529–533.

Montoya-Zegarra, J. A., Wegner, J. D., Ladicky, L., Schindler, K., 2014. Mind the gap: Modeling local and global context in (road) networks. In: German Conference on Pattern Recognition, 2014.

Moranduzzo, T., Melgani, F., 2014a. Automatic car counting method for unmanned aerial vehicle images. Geoscience and Remote Sensing, IEEE Transactions on 52 (3): 1635–1647.

Moranduzzo, T., Melgani, F., 2014b. Detecting cars in uav images with a catalog-based approach. Geoscience and Remote Sensing, IEEE Transactions on 52 (10): 6356–6367.

Máttyus, G., Fraundorfer, F., 2016. Aerial image sequence geolocalization with road traffic as invariant feature. Image and Vision Computing 52: 218 – 229.

Müller, R., Krauß, T., Schneider, M., Reinartz, P., 2012. Automated Georeferencing of Optical Satellite Data with Integrated Sensor Model Improvement. Photogrammetric Engineering and Remote Sensing 71 (1): 61–74.

Nowozin, S., Lampert, C. H., 2011. Structured learning and prediction in computer vision. Foundations and Trends in Computer Graphics and Vision 6 (3&#8211;4): 185–365.

Papert, S., 1966. The summer vision project. https://dspace.mit.edu/handle/1721.1/6125.

Porway, J., Wang, Q., Zhu, S., 2009. A hierarchical and contextual model for aerial image parsing. International Journal of Computer Vision (IJCV) 88 (2): 254–283.

Ren, S., He, K., Girshick, R. B., Sun, J., 2015a. Faster R-CNN: towards real-time object detection with region proposal networks. CoRR abs/1506.01497.

Ren, S., He, K., Girshick, R. B., Sun, J., 2015b. Faster R-CNN: towards real-time object detection with region proposal networks. In: Conference on Neural Information Processing Systems (NIPS), 2015., 91–99.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., Fei-Fei, L., 2015. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) 115 (3): 211–252.

Schwing, A. G., Fidler, S., Pollefeys, M., Urtasun, R., 2013. Box In the Box: Joint 3D Layout and Object Reasoning from Single Images. In: International Conference on Computer Vision (ICCV), 2013.

Schwing, A. G., Urtasun, R., 2015. Fully connected deep structured networks. CoRR abs/1503.02351.

Seo, Y.-W., Urmson, C., Wettergreen, D., 2012a. Exploiting publicly available cartographic resources for aerial image analysis. In: SIGSPATIAL.

Seo, Y.-W., Urmson, C., Wettergreen, D., 2012b. Ortho-image analysis for producing lane-level highway maps. Tech. Rep. CMU-RI-TR-12-26, CMU.

Shan, Q., Wu, C., Curless, B., Furukawa, Y., Hernandez, C., Seitz, S., 2014. Accurate geo-registration by ground-to-aerial image matching. In: 3DV.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D., 2016. Mastering the game of go with deep neural networks and tree search. Nature 529 (7587): 484–489.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556.

Snyder, J., Voxland, P., 1989. An Album of Map Projections. No. no. 1453 in An Album of Map Projections. U.S. Government Printing Office.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15: 1929–1958.

Stoica, R., Descombes, X., Zerubia, J., 2004. A gibbs point process for road extraction from remotely sensed images. International Journal of Computer Vision (IJCV) .

Sun, L., Jia, K., Yeung, D.-Y., Shi, B. E., 2015. Human action recognition using factorized spatio-temporal convolutional networks. In: International Conference on Computer Vision (ICCV), 2015.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2014. Going deeper with convolutions. CoRR abs/1409.4842.

Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y., 2005. Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research (JMLR) 6: 1453–1484.

Tuermer, S., Kurz, F., Reinartz, P., Stilla, U., 2013. Airborne vehicle detection in dense urban areas using hog features and disparity maps. Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of 6 (6): 2327–2337.

Turetken, E., Benmansour, F., Andres, B., Pfister, H., Fua, P., 2013. Reconstructing loopy curvilinear structures using integer programming. In: Conference on Computer Vision and Pattern Recognition (CVPR), 2013.

Veeriah, V., Zhuang, N., Qi, G.-J., 2015. Differential recurrent neural networks for action recognition. In: International Conference on Computer Vision (ICCV), 2015.

Viola, P., Jones, M., 2001. Rapid object detection using a boosted cascade of simple features. In: Conference on Computer Vision and Pattern Recognition (CVPR), 2001., Vol. 1, I–511–I–518 vol.1.

Viola, P., Jones, M. J., 2004. Robust real-time face detection. International Journal of Computer Vision (IJCV) 57 (2): 137–154.

Viterbi, A., 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory 13 (2): 260–269.

Waltz, D. L. (Hrsg.), 1982. Proceedings of the National Conference on Artificial Intelligence. Pittsburgh, PA, August 18-20, 1982. AAAI Press.

Wang, C., Stefanidis, A., Agouris, P., 2007. Relaxation matching for georegistration of aerial and satellite imagery. In: IEEE International Conference on Image Processing (ICIP), 2007., Vol. 5, 449–452.

Wang, S., Fidler, S., Urtasun, R., 2015a. Holistic 3d scene understanding from a single geo-tagged image. In: Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

Wang, S., Fidler, S., Urtasun, R., 2015b. Lost shopping! monocular localization in large indoor spaces. In: International Conference on Computer Vision (ICCV), 2015., 2695–2703.

Wegner, J. D., Montoya-Zegarra, J. A., Schindler, K., 2013. A higher-order crf model for road network extraction. In: Conference on Computer Vision and Pattern Recognition (CVPR), 2013., 1698–1705.

Weyand, T., Kostrikov, I., Philbin, J., 2016. Planet - photo geolocation with convolutional neural networks. CoRR abs/1602.05314.

Wilson, R., Hancock, E., 1993. Relaxation matching of road networks in aerial images using topological constraints. Sensor Fusion 2059: 444–455.

Winn, J., Criminisi, A., Minka, T., 2005. Object categorization by learned universal visual dictionary. In: International Conference on Computer Vision (ICCV), 2005.

Wolfson, H., Rigoutsos, I., 1997. Geometric hashing: an overview. Computational Science Engineering, IEEE 4 (4): 10–21.

Workman, S., Souvenir, R., Jacobs, N., 2015. Wide-area image geolocalization with aerial reference imagery. In: International Conference on Computer Vision (ICCV), 2015.

Wu, C., 2013. Towards linear-time incremental structure from motion. In: 3DTV-Conference, 2013 International Conference on, 127–134.

Wu, C., Fraundorfer, F., Frahm, J.-M., Snoeyink, J., Pollefeys, M., 2008. Image localization in satellite imagery with feature-based indexing. In: Commission III, ISPRS Congress 2008 Beijing, Vol. XXXVII, 197–202.

Yu, F., Koltun, V., 2015. Multi-scale context aggregation by dilated convolutions. CoRR abs/1511.07122.

Yuan, J., Cheriyadat, A., 2013. Road segmentation in aerial images by exploiting road vector data. In: COM.geo.

Zeiler, M. D., Fergus, R., 2014. Visualizing and understanding convolutional networks. In: European Conference on Computer Vision (ECCV), 2014., 818–833.

Zhang, T., 2004. Statistical behavior and consistency of classification methods based on convex risk minimization. The Annals of Statistics 32: 56–134.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P. H. S., 2015. Conditional random fields as recurrent neural networks. In: International Conference on Computer Vision (ICCV), 2015.

Acknowledgments

The writing and completion of this thesis would not have been possible without the support and help of many generous people.

I feel fortunate that I could spend the years working on my PhD at the German Aerospace Center (DLR), a dream institute of many children and adults, embodying the vision of mankind reaching the stars. I would like to truly thank Prof. Peter Reinartz,- the head of the Photogrammetry and Image Analysis (PBA) Department-, who gave me this opportunity and supported me to work on my PhD; Prof. Richard Bamler,- my professor at the Technical University of Munich and the head of the Remote Sensing Technology Institute (IMF)-, who supported me writing and submitting this thesis. I would like to express my gratitude to Prof. Friedrich Fraundorfer for his supervision and excellent help in various forms. I have really enjoyed our discussions and our collaboration.

I would thank my team leader, Rupert Müller, the team of the project Vabene: Dr. Franz Kurz, Dr. Dominik Rosenbaum, Oliver Meynberg, Dr. Stefan Auer, Dr. Janja Avbelj, Vronika Gsteiger, Dr. Shiyong Cui, Dr. Elenora Vig and all others colleagues and students with whom I worked together during my 5 years at DLR. I am grateful to my colleague Dr. Eleonora Vig for proof reading my thesis and Dr. Franz Kurz for helping me translate the abstract to German.

I feel particularly lucky that I had the opportunity to visit the Machine Learning Group at the Computer Science Department of the University of Toronto (UofT) multiple times. I would like to especially thank Prof. Raquel Urtasun,- the head of the Machine Learning Group at UofT -, who accepted me as a visiting student and invited me back several times to deepen our cooperation. I find our collaboration with Prof. Raquel Urtasun, Prof. Sanja Fidler (UofT) and Shenlong Wang (also UofT) extremely inspiring, raising the level of my research enormously. I am very grateful for all their help. I truly enjoyed our discussions, and writing our papers. I offer thanks to all other members of the Machine Learning Group at UofT who received me very friendly and made me feel home in Toronto.

I would like to thank Kang Liu for all his efforts enabling the writing of our paper.

I hope to stay in contact with all of you in the future.

The years of reading papers, researching, writing papers, getting rejected or accepted contains both successful, encouraging, motivating moments and depressing, futureless periods with fails and disappointments. It would not have been possible to overcome these difficult times without the support of my parents Dr. Piroska Róna and Dr. István Máttyus and my sisters Kinga Máttyus and Zita Máttyus. I feel truly grateful for their help and warm-hearted care allowing me to concentrate on my studies and work. I thank Kinga extra for proof reading parts of my thesis.

Last but not at least, I would like to thank the many friends for causing a lot of happy moments, particularly my flatmates who were bearing me before deadlines. :)

Appendices

A Liu K., Mattyus, G., 2015. Fast Multiclass Vehicle Detection on Aerial Images. IEEE Geoscience and Remote Sensing Letters 12(9): 1938-1942

http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7122912

Fast Multiclass Vehicle Detection on Aerial Images

Kang Liu and Gellert Mattyus

***Abstract*—Detecting vehicles in aerial images provides important information for traffic management and urban planning. Detecting the cars in the images is challenging due to the relatively small size of the target objects and the complex background in man-made areas. It is particularly challenging if the goal is near-real-time detection, i.e., within few seconds, on large images without any additional information, e.g., road database and accurate target size. We present a method that can detect the vehicles on a 21-MPixel original frame image without accurate scale information within seconds on a laptop single threaded. In addition to the bounding box of the vehicles, we extract also orientation and type (car/truck) information. First, we apply a fast binary detector using integral channel features in a soft-cascade structure. In the next step, we apply a multiclass classifier on the output of the binary detector, which gives the orientation and type of the vehicles. We evaluate our method on a challenging data set of original aerial images over Munich and a data set captured from an unmanned aerial vehicle (UAV).**

***Index Terms*—Classification, near real-time, vehicle detection.**

I. Introduction

The detection of vehicles in aerial images is important for various applications, e.g., traffic management, parking lot utilization, urban planning, etc. Collecting traffic and parking data from an airborne platform gives fast coverage over a larger area. Getting the same coverage by terrestrial sensors would need the deployment of more sensors and more manual work and, thus, higher costs.

A good example for an airborne road traffic measuring system is the one in the project *Vabene*[1] [1] of the German Aerospace Center (DLR). In this real-time system, aerial images are captured over roads, and the vehicles are detected and tracked across multiple consecutive frames. This gives fast and comprehensive information of the traffic situation by providing the number of vehicles and their position and speed. Fig. 1 provides the overview of our work flow and illustration of the output. The detection is a challenging problem due to the small size of the vehicles (a car might be only 30×12 pixels) and the complex background of man-made objects, which appear visually similar to the cars. Providing both the position and the orientation of the detected objects supports the tracking by giving constraints on the motion of the vehicles. This is particularly important in dense traffic scenes where the object assignment is more challenging. The utilization of roads and parking lots depends also on the type of the vehicle (e.g., a truck impacts the traffic flow different as a personal car). A system having access to this richer information can manage the infrastructure better. In a real-time system, as in [1], the processing time (and computing power) is limited. Therefore, the processing method should be as fast as possible.

Our vehicle detection method provides both robust performance and fast speed and vehicle orientation and type information fully automatically based only on the input image.

We detect the bounding box of the vehicles by a very fast binary sliding-window detector using integral channel features (ICFs) and an AdaBoost classifier in a soft-cascade structure. The bounding boxes are further classified to different orientations and vehicle type based on histogram of oriented gradients (HOG) features [2].

We test and evaluate our method on a challenging data set over the city of Munich, Germany, and another data set collected by a UAV. These data sets contain original and nonorthorectified frame images, which makes the problem more challenging since the exact GSD[2] is unknown (we have only an approximate prior). To make our results better comparable to other methods, we release the Munich images with the ground truth[3] (see Table I). To show the robustness of the method, we also present qualitative results on images downloaded from Google Earth around the world in the supplementary material.

Our main contributions are presented as follows: 1) The presented method uses features that can be calculated rapidly in a soft-cascade structure. This makes the detection very fast, i.e., it takes only a few seconds on a 21-MPixel image on a laptop single threaded. 2) Our method also works on a single original frame image without any georeferencing, exact GSD, street database, or 3-D image information. 3) In addition to the location, we also estimate the orientation and type of the vehicles.

II. Related Work

The vehicle detection in aerial images has a large literature; here, we mention only a few important recent papers.

Moranduzzo and Melagni [3], [4] process very high-resolution (2-cm GSD) UAV images for car detection. In [3], a feature point detector and support vector machine (SVM) classification of scale-invariant feature transform descriptors is applied, while the method in [4] uses a catalog of HOG descriptors and later an orientation estimation.

In [5], the cars are detected by a deep neural network running on the graphics processor unit in a sliding-window approach on a known constant scale. In [6], the vehicles are detected with online boosting on Haar-like features, local binary patterns, and

Manuscript received December 24, 2014; revised March 19, 2015 and May 21, 2015; accepted May 23, 2015. Date of publication June 12, 2015; date of current version August 7, 2015. This work was supported by the German Aerospace Center (DLR) Project *Vabene*++.

The authors are with the Remote Sensing Technology Institute, German Aerospace Center, 82234 Wessling, Germany.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LGRS.2015.2439517

[1]http://www.dlr.de/vabene/

[2]Ground Sampling Distance.

[3]http://www.dlr.de/eoc/desktopdefault.aspx/tabid-5431/9230_read-42467/

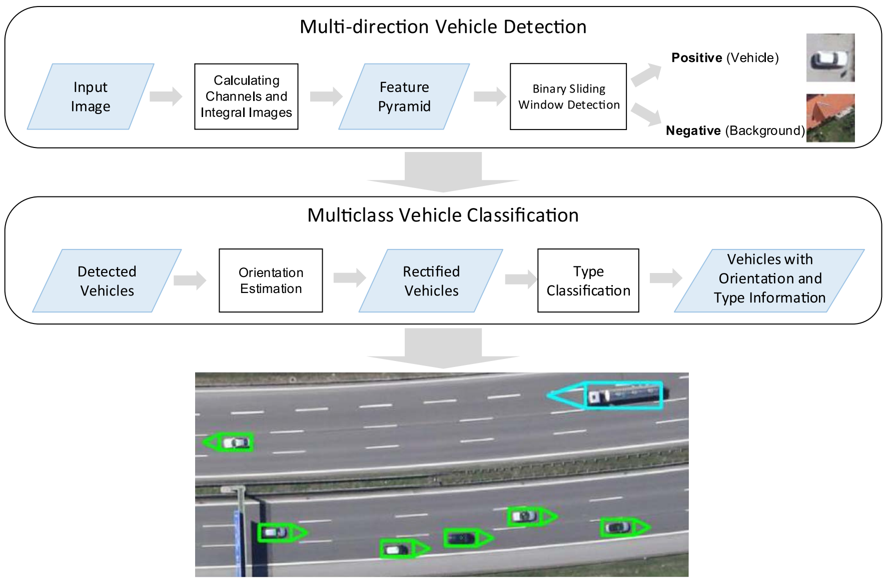

Multi-direction Vehicle Detection
Input Image
Calculating Channels and Integral Images
Feature Pyramid
Binary Sliding Window Detection
Positive (Vehicle)
Negative (Background)
Multiclass Vehicle Classification
Detected Vehicles
Orientation Estimation
Rectified Vehicles
Type Classification
Vehicles with Orientation and Type Information

Fig. 1. Proposed vehicle detection framework. The input image is first evaluated by the multidirection vehicle detector. A sliding window goes along the x- and y-axes. Features are extracted from the detection window and sent to a trained binary classifier. The binary classifier identifies whether the current detection window contains a positive object or not. Detected vehicles are then processed for estimating their orientations and categories.

TABLE I
PERFORMANCE COMPARISON BETWEEN DIFFERENT METHODS

Method	**Ground Truth**	**True Positive**	**False Positve**	**Recall Rate**	**Precision Rate**
Munich dataset					
Viola-Jones	5892	3237	1467	54.9%	68.8%
Ours	5892	**4085**	**619**	**69.3%**	**86.8%**
UAV dataset from [3] [4]					
[3]	119	88	143	73.95%	38.1%
[4]	119	87	111	73.1%	43.4%
Ours	119	**94**	**6**	**79.0%**	**94.0%**

orientation histograms. They train the detector for cars in one direction, and during testing, they rotate the image in a 15° step. This detector is trained for a known object size of 35 × 70 pixels and tested on images with the same scale.

Leitloff *et al.* [1] use a two-stage approach for the detection of cars: first an AdaBoost classifier with Haar-like features and then an SVM on various geometric and radiometric features. They use the road database as a prior to detect only along the roads in a certain direction. The method achieves good results, running fast on a CPU; however, it is limited to orthorectified images and areas covered by the road database.

Tuermer *et al.* [7] utilize the road map and stereo matching to limit the search area to roads and exclude buildings. The HOG features with an AdaBoost classifier are applied to detect the cars on the selected region. This method is limited to georeferenced image pairs and areas covered by the road database.

III. MULTIDIRECTION VEHICLE DETECTION

We handle the vehicle detection problem in two stages. The first stage is a very fast binary sliding-window object detector, which delivers axis-aligned bounding boxes of the vehicles without type or orientation information. The second stage is a multiclass classifier applied on the bounding boxes, estimating the orientation and the type of the vehicles. The processing steps are shown in Fig. 1.

A. Binary Sliding-Window Detector

For fast detection, both the feature calculation and the classification have to be efficient.

1) Fast Image Features: Viola and Jones [8] introduced the integral image concept with Haar-like features for fast and robust face detection. By using the integral image I_Σ, the pixel intensity I sum of the Haar-like features is calculated by a few operations independent of the area of the feature. The value $I_\Sigma(x, y)$ at the (x, y) location in an integral image is the sum of the pixels above and to the left of (x, y) as follows:

$$I_\Sigma(x, y) = \sum_{i=0}^{i\le x}\sum_{j=0}^{j\le y} I(i, j). \tag{1}$$

The integral f_I within an axis-aligned rectangle defined by its upper left corner x_0, y_0, width w, and height h is calculated as $f_I = I_\Sigma(x_0 + w, y_0 + h) + I_\Sigma(x_0, y_0) - I_\Sigma(x_0 + w, y_0) - I_\Sigma(x_0, y_0 + h)$.

This idea is generalized by the ICFs in the work of Dollar *et al.* [9]. Instead of working on pixel intensity values, as in [8], an *ICF* can be constructed on top of an arbitrary feature channel (i.e., the transformation of the original image). Features are defined as linear combinations of sums over local rectangular regions in the channels. By using the concept of integral images, an integral channel can be precomputed for each feature channel, so that the computation of the sum over the rectangle is very fast. The most commonly used channels are the color intensities, the gradient magnitude, and the gradient histogram. The gradient histogram is a weighted histogram where the bin is determined by the gradient orientation. It is given by $Q_\Theta(x, y) = G(x, y)\mathbf{1}[\Theta(x, y) = \theta]$, where $G(x, y)$ is the gradient magnitude, and $\Theta(x, y)$ is the quantized gradient orientation at the x, y image location. The gradient histogram can approximate the powerful and widely used HOG features [2]. If the rectangles are defined as squares, the sum can be aggregated to a single pixel in a downsampled image. In this case, the integral is calculated even faster as a single pixel lookup. This method is also called aggregated channel features [10]. For rapid speed, we apply this method with fast feature pyramid calculation, as described in [10].

2) AdaBoost Classifier in Soft-Cascade Structure: The number of ICFs is very large (larger than the number of pixels in the image window) since it is the linear combination of local rectangular regions in the image window. We select only relevant features by the Discrete AdaBoost algorithm [11] for N weak classifiers $h_t(\mathbf{x})$. $h_t(\mathbf{x})$ is a simple classifier, e.g., a threshold or a shallow decision tree of a few features from the input feature vector $\mathbf{x}$. AdaBoost is an iterative algorithm; in each step, it reweights the samples in the training set, according to the classification result from the previous weak classifier. The final strong classifier H is composed of the weighted α_t weak classifiers $h_t(\mathbf{x})$. That is

$$H = \mathrm{sgn} \sum_{t=1}^{N} \alpha_t h_t(\mathbf{x}). \qquad (2)$$

At numerous sliding-window positions (e.g., homogeneous regions), not all the weak classifiers have to be evaluated to classify the image as nonvehicle. To leverage this property for speed improvement, we form a soft cascade [12] from the weak classifiers. During the training, a threshold r_t is set for all the weighted weak classifiers $c_t = \alpha_t h_t(\mathbf{x})$. If the cumulative sum $H_t(\mathbf{x}) = \sum_{i=1,\ldots,t} c_i(\mathbf{x})$ of the first t output functions is $H_t(\mathbf{x}) \geq r_t$, then the input sample is passed to the subsequent evaluation process; otherwise, it is classified as negative and rejected immediately.

B. Multidirection Detection

The orientation of the vehicles in aerial images can be arbitrary. This increases the intraclass variation of the appearance in the axis-aligned sliding windows. A straightforward but computationally expensive solution, which was used in [6], is to train the detector for one specific direction and rotate the input image and do detection for each rotation. This would need the computation of the integral images separately for each direction and would result in slow processing speed. To overcome this, we propose two methods: one is to train a single classifier, which is able to detect differently oriented vehicles; the other is to aggregate several simple classifiers, where each is only sensitive to specific directions.

1) Single Classifier Method: A single binary classifier is trained with samples covering all the directions. The training process has to deal with the high intraclass variety and find the common part of all the positive samples. When the detector is applied on the input image, vehicles in any directions can be classified as positive samples.

2) Aggregated Classifier Method: Alternatively, the intraclass variety is reduced by splitting the training to different orientations. Multiple binary classifiers are trained, each for specific vehicle orientations. These classifiers are employed in sequence during the detection phase, and the results from each classifier are aggregated using nonmaximal suppression. The integral image does not need to be calculated multiple times, only the classification.

TABLE II
COMPARISON OF COMPUTATION TIMES

Method	Image Resolution	Detection Time Per Image [s]	Detection Time Per MPixel [s]
Proposed	5616 × 3744	**4.4**	**0.2**
Viola-Jones	5616 × 3744	1160	55.2
[4]	5184 × 3456	14400	803.8
[5] [a]	1368 × 972	8	6.0

[a] Running on the GPU.

The performances of these two methods are examined in Section V.

IV. Multiclass Vehicle Classification

The detector provides the axis-aligned bounding boxes of the vehicles. In this next step, we refine the extracted information by classifying the orientation and the type of the vehicle. We propose a two-step approach containing an orientation estimator and a type classifier. A sample is sent to the orientation estimator first, then rotated to the horizontal direction according to the orientation estimation, and finally processed by the type classifier to identify which type category this vehicle belongs to.

A. Orientation Estimation

We consider the orientation estimation as a multiclass classification problem. The directions are clustered, and each cluster is considered as a class. The ICF features can be calculated fast, but they have a very high number; thus, they are not suitable for multiclass classifiers working on fixed-length feature vectors. Therefore, we apply the powerful HOG feature [2], which has a fixed feature vector length. We use a neural network with one hidden layer as a multiclass classifier [13].

B. Type Classification

The type classifier needs to classify the input image into corresponding categories. We have defined two type classes, i.e., car and truck, but the presented method could be extended to more classes. The object bounding box is rotated to the horizontal direction based on the orientation estimation. Unrelated context is cropped out, and HOG features are again extracted and classified by the type classifier.

V. Experimental Results

We test the multidirection detection and multiclass classification parts in our detection method, respectively, and give quantitative results for the different processing stages. The binary detector is trained with 2048 weak classifiers in each test. We use depth-two decision trees as weak classifiers.

A. Results on Munich Images

The quantitative evaluation is performed on 20 aerial images captured by the *DLR 3K* camera system [1] over the area of Munich, Germany. We use the original nadir images with the

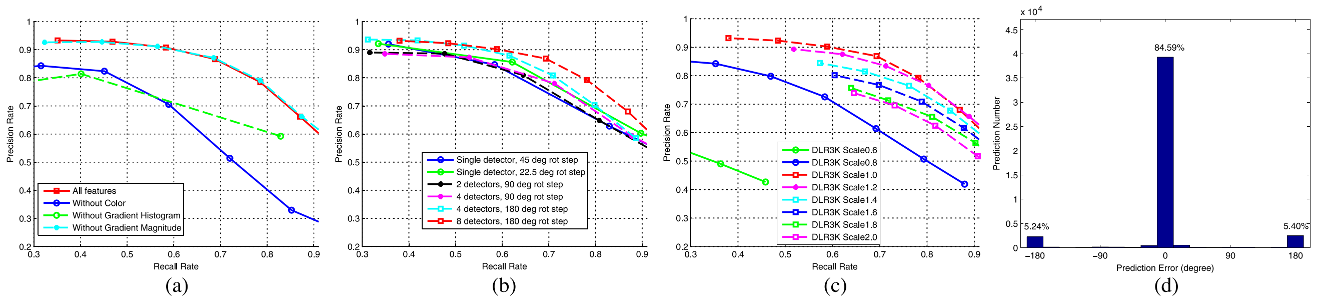


Fig. 2. (a) Evaluation of the ICFs. Gradient histogram channels play the most important role, while the gradient magnitude channel has the least effect on the final result. (b) Detection result of aggregated detectors. (c) Performance after rescaling the image with different factors. (d) Orientation estimation error histogram using an artificial neural network with 16 output classes.

resolution of 5616 × 3744 pixels (see Table II). They are taken at a height of 1000 m above the ground; the approximate GSD is 13 cm. The first ten images are used for training and the other ten for testing. Positive training samples come from 3418 cars and 54 trucks annotated in the training images, while the negatives are randomly picked from the background, i.e., areas without vehicles. To overcome the low number of truck samples, we randomly transformed them additionally 30 times. Fig. 3 shows detection results on the test images. We set the detection window to 48 × 48 pixels. For the ground truth, the vehicles in the images are annotated manually as oriented bounding boxes.

1) Multidirection Vehicle Detection: ICFs contain rich information and can be computed rapidly. They are selected as the features for training and detection. Experiments are performed to evaluate the importance of each feature channel type and the performance of different classifier configurations.

a) Feature channel: We use three types of feature channels: Luv color, gradient magnitude, and gradient histogram. We have evaluated the contribution of each feature channel; the precision–recall (PR) curves are plotted in Fig. 2(a). These curves indicate that gradient histogram channels play the most important role in representing the vehicles, while the gradient magnitude channel affects the final result the least. For the later tests, we use all the feature channels.

b) Multidirection detection methods: We proposed two methods, i.e., single and aggregative classifiers, to detect vehicles in different directions (see Section III-B). The performances are depicted in Fig. 2(b). The PR curve shows that the optimal solution is the "Classifier aggregation method," with each classifier trained using samples in opposite directions (eight detectors with a sample rotation step of 180°). This means eight detectors and, thus, longer computation time. A single detector needs 2.7 s, while the detection with 8 classifiers takes 4.1 s. This is sublinear since the integral images do not have to be calculated again. We use the eight-classifier configuration for the later tests.

c) Detection on images with different scales: To show the ability of our method to detect the cars on images with different scales, we resized the image for the test but not the training. These results are shown in Fig. 2(c). The detector performs best on the same scale as it was trained; if the resolution is increased, the performance remains comparable. However, if we decrease the resolution, we lose information, which leads to a lower performance.

2) Multiclass Vehicle Classification: After the axis-aligned bounding box detection, we classify the orientation and type of

TABLE III
CONFUSION MATRICES OF TYPE CLASSIFICATION USING DIFFERENT CROPPING CONFIGURATIONS

Cropped Size	48 × 48			**48 × 28**		
Confusion Matrix	A/P [a]	Car	Truck	A/P	Car	Truck
	Car	2843	60	Car	**2838**	**65**
	Truck[b]	123	685	Truck	**0**	**808**
Accuracy	95.1%			**98.2%**		

[a] Actual class / Predicted class
[b] The number of truck type is increased by random transformation of the existing samples.

the vehicles. We convert all the bounding boxes to 48 × 48 pixel gray images and calculate HOG features for this image. We get the best performance with 4 × 4 cell size, 1 × 1 block size, and 1 × 1 block stride HOG feature configuration and use this for the later tests. The comparison of different HOG configurations can be found in the supplementary material.

a) Orientation estimation: Orientation classification is performed, according to Section IV-A, with 16 classes (22.5° rotation difference between adjacent sample groups, respectively). The orientation estimation error histogram is depicted in Fig. 2(d). In the supplementary material, we provide results with different number of classes and an additional random forest classifier [14]. The most common error is when the samples are classified in the opposite direction. This is because the front part of a vehicle might be similar to the rear part from the top view in aerial images.

b) Type classification: The detected bounding box is rotated to the horizontal direction according to the orientation estimation. We trim the input image by cropping the upper and lower parts, from 48 × 48 to 48 × 28. In our data set, the number of trucks is much less than the number of cars. We generate new ones from the existing samples using random transformation. The performances with different cropping configurations are compared in Table III and the supplementary material. The optimal type classification can reach 98.2% in accuracy with a one-hidden-layer neural network.

3) Baseline Comparison: As baseline, we use the OpenCV[4] implementation of the Viola–Jones detector [8]. We have trained it on one vehicle direction, while at detection, we rotate the image similar as in [6] and apply the detector for each rotated image. Table I contains the numerical comparison of this method on the Munich data set.

[4]http://opencv.org/

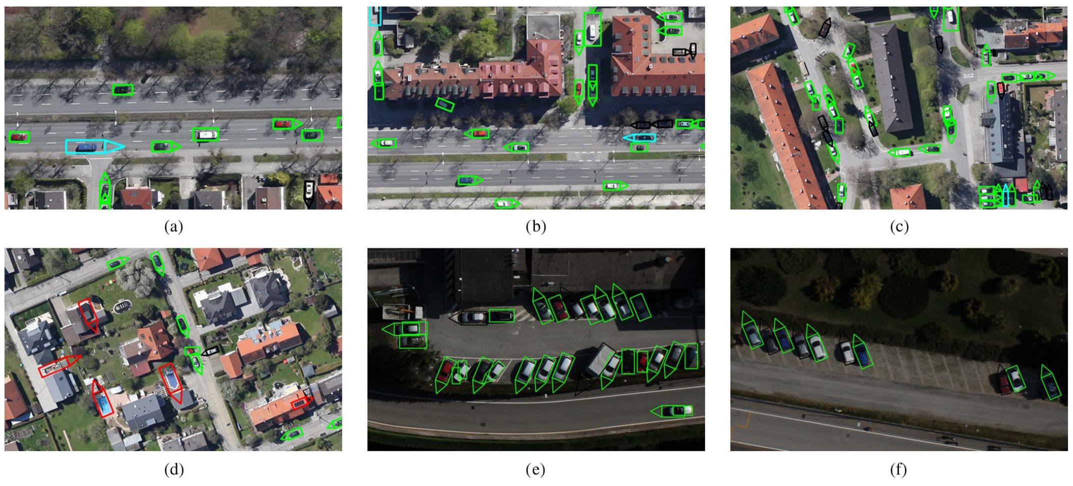


Fig. 3. Detection results from the DLR test images. Green and cyan bounding boxes are the correct detected samples, representing cars and trucks, respectively. Black bounding boxes are the missed ones, and red bounding boxes are the false positives. The results show that our method works well in most scenarios (a)–(c); however, the complicated rooftops or outdoor swimming pools may lead to false positive detections (d). We also evaluated our method on the data set presented in [3] and [4]; the detection results are shown in (e) and (f).

4) Computation Time: Since the processing time is also important for the detector, we compare our method with other methods, where the computation time is provided in this paper. Table II contains the computation times. Our experiments are performed on a laptop with Intel Core i5 processor and 8-GB memory, and our program is running single threaded written in MATLAB and C++. The comparisons show that the speed of our method is considerably faster. This makes our method more suitable for real-time systems, where the computation time is a serious issue. The method in [5] achieves comparable detection performance but on a different data set; therefore, we show only the processing time of the method.

B. Baseline Comparison on UAV Images

We also evaluated our method on the data set presented in [3] and [4] (see Fig. 3) and compared to the results provided without screening. The results can be found in Table I. The precision rate of our method outperforms the other methods, significantly. Due to the higher resolution, we set the detection window to 96×96 pixels for this data set, and we have only car vehicle type (no truck).

C. Qualitative Results From Around the World

To show the robustness of our detector, we also run our detector on images downloaded from Google Earth. These can be found in the supplementary material.

VI. Conclusion

We have presented a method that can detect vehicles with orientation and type information on aerial images in a few seconds on large images. The application of ICFs in a soft-cascade structure results in both good detection performance and fast speed. The detector works on original images where no georeference and resolution information is available. As future work, the performance could be further improved by using a deep neural network after the binary detector, such as Regions with Convolutional Neural Network Features (R-CNN) in [15]. Since this has to be applied only to a fraction of the image, the speed of the detector would be still fast.

Acknowledgment

The authors would like to thank T. Moranduzzo and F. Melgani, who generously provided their image data set with the ground truth.

References

[1] J. Leitloff, D. Rosenbaum, F. Kurz, O. Meynberg, and P. Reinartz, “An operational system for estimating road traffic information from aerial images,” *Remote Sens.*, vol. 6, no. 11, pp. 11 315–11 341, 2014.
[2] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proc. IEEE CVPR*, 2005, vol. 1, pp. 886–893.
[3] T. Moranduzzo and F. Melgani, “Automatic car counting method for unmanned aerial vehicle images,” *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 3, pp. 1635–1647, Mar. 2014.
[4] T. Moranduzzo and F. Melgani, “Detecting cars in UAV images with a catalog-based approach,” *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 10, pp. 6356–6367, Oct. 2014.
[5] X. Chen, S. Xiang, C. Liu, and C. Pan, “Vehicle detection in satellite images by hybrid deep convolutional neural networks,” *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 10, pp. 1797–1801, Oct. 2014.
[6] S. Kluckner, G. Pacher, H. Grabner, H. Bischof, and J. Bauer, “A 3D teacher for car detection in aerial images,” in *Proc. IEEE 11th ICCV*, 2007, pp. 1–8.
[7] S. Tuermer, F. Kurz, P. Reinartz, and U. Stilla, “Airborne vehicle detection in dense urban areas using HoG features and disparity maps,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 6, no. 6, pp. 2327–2337, Dec. 2013.
[8] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proc. IEEE CVPR*, 2001, vol. 1, pp. I-511–I-518.
[9] P. Dollar, Z. Tu, P. Perona, and S. Belongie, “Integral channel features,” in *Proc. BMVC*, 2009, pp. 91.1–91.11.
[10] P. Dollar, R. Appel, S. Belongie, and P. Perona, “Fast feature pyramids for object detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 8, pp. 1532–1545, Aug. 2014.
[11] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: A statistical view of boosting,” *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, 2000.
[12] L. Bourdev and J. Brandt, “Robust object detection via soft cascade,” in *Proc. IEEE CVPR*, 2005, vol. 2, pp. 236–243.
[13] Y. Lecun, L. Bottou, G. B. Orr, and K. R. Müller, “Efficient BackProp,” in *Neural Networks: Tricks of the Trade*. Lecture Notes in Computer Science, G. Orr and K. Müller, Eds. Berlin, Germany: Springer-Verlag, 1998, vol. 1524, pp. 9–50.
[14] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proc. IEEE CVPR*, 2014, pp. 580–587.

B Mattyus G., Fraundorfer, F., 2016. Aerial image sequence geolocalization with road traffic as invariant feature. Image and Vision Computing, Volume 52: 218–229.

`http://www.sciencedirect.com/science/article/pii/S0262885616301056`

This is the preprint version of the paper.

Aerial image sequence geolocalization with road traffic as invariant feature

Gellért Máttyus
German Aerospace Center, Germany, Remote Sensing Technology Institute
gellert.mattyus@dlr.de

Friedrich Fraundorfer
German Aerospace Center, Germany, Remote Sensing Technology Institute
Graz University of Technology, Austria, Institute for Computer Graphics and Vision
fraundorfer@icg.tugraz.at

Abstract

The geolocalization of aerial images is important for extracting geospatial information (e.g. the position of buildings, streets, cars, etc.) and for creating maps. The standard is to use an expensive aerial imaging system equipped with an accurate GPS and IMU and/or do laborious Ground Control Point measurements. In this paper we present a novel method to recognize the geolocation of aerial images automatically without any GPS or IMU. We extract road segments in the image sequence by detecting and tracking cars. We search in a database created from a road network map for the best matches between the road database and the extracted road segments. Geometric hashing is used to retrieve a shortlist of matches. The matches in the shortlist are ranked by a verification process. The highest scoring match gives the location and orientation of the images. We show in the experiments that our method can correctly geolocalize the aerial images in various scenes: e.g. urban, suburban, rural with motorway. Beside the current images only the road map is needed over the search area. We can search an area of 22500 *km^2 containing* 32000 *km of streets within minutes on a single cpu.*

1. Introduction

It is required for numerous applications that images and videos are tagged with their location on the earth's surface (geolocation). The geolocalization of aerial images is particularly essential for extracting geospatial informations (e.g. the position of buildings, streets, cars, etc.) and for creating maps. The aerial imaging systems typically include an accurate (and expensive) GPS and IMU. By avoiding these instruments a simple consumer camera could also be used to create georeferenced aerial images. People would be able to use their own camera during leisure and touristic activities (e.g. hang gliding, air balloon flight, sightseeing flight, glider flight, etc.) to create georeferenced and orthorectified images. A crowd-sourced database of orthorectified aerial images (similar as OpenStreetMap for maps) could be used for mapping applications.

Although nowadays even consumer grade cameras can provide geotagging, it might be needed to find the geolocation of the camera, image and the objects in the scene based only on the visual information. (See the IARPA Finder program [1]). Even with a GPS position tag the orientation of image is usually still unknown. Aerial image camera systems equipped with a very precise GPS and IMU also might lack the geolocation in case of an outage. The automatic geolocalization of these images can reduce expenses by avoiding a new flight or laborous manual work. It is also getting common to acquire images from a unmanned aerial vehiclec (UAV). These can only carry light payloads, limiting the accuracy of the onboard GPS and IMU. Retrieving a more accurate image position and orientation during post processing could be needed.

UAVs also need a backup localization system in case of an outage of the GPS (GNSS). A GPS outage can happen either due to a problem in the device or because of external jamming of the GPS signal. The jamming of the GPS signal is a realistic threat, not only in military environments. The problem of GPS jamming is addressed in research, e.g. in [11].

Our goal is the geolocalization of aerial photo sequences on a large scale by utilizing only the image information and the road network. We propose to match the information acquired from the current aerial image sequence to an object database

[1] http://www.iarpa.gov/Programs/ia/Finder/finder.html

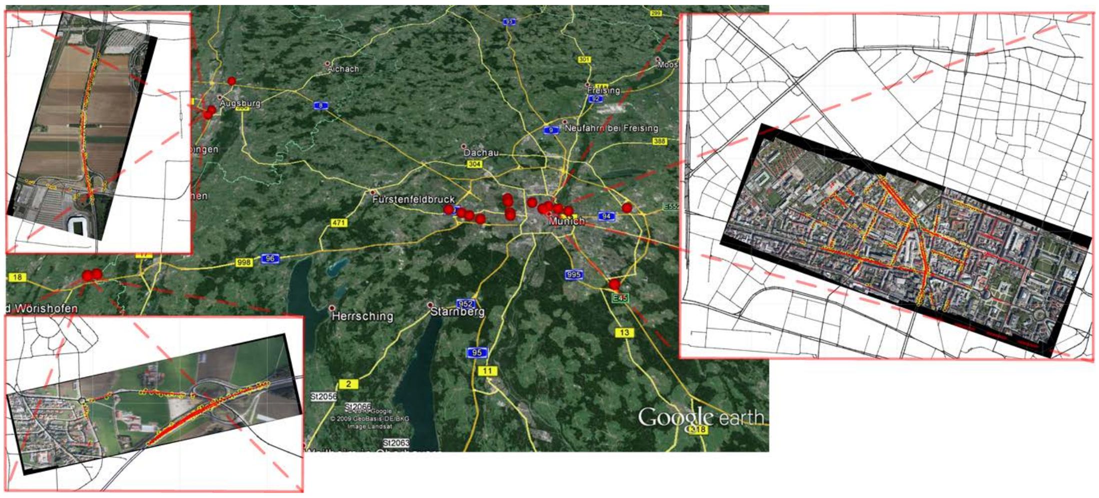


Figure 1. Illustration. We search the geolocation of aerial image sequences. The red dots on the map show the ground truth positions. The photos can be located by matching the road traffic (in red) in the image to the road network (in black). The matching tracks are marked with yellow circles in the images. The search area can be as large the shown map. The accuracy of the location is around 25 m.

invariant to the lighting and weather conditions, the road network map. Since the maintenance of an accurate, up-to-date road map is needed for many other applications (e.g. navigation, administration, etc.), it is more easily accessible than a large image database and requires significantly less storage [2].

We detect the road traffic in the image scene by tracking cars over the frames. By assuming the cars drive on the roads the vehicle trajectories can be interpreted as subsets of the roads, and be matched to the road map.

We propose a method for the fast retrieval of a shortlist containing the possible correct location. This retrieval is based on Geometric Hashing. We call it *Polyline Based Geometric Hashing (PLBGHashing)*, and it can search rapidly over a larger search area. The more complex verification matching needs to be done only on the retrieved shortlist. The car tracks can be considered as a road detector with low completeness but high correctness. We combine this with a simple pixel color based road detector (with high completeness but low correctness) for the verification.

We analyzed the proposed *PLBGHashing* on synthetic data generated from the road network of two large cities. This confirmed that the pattern of the road network is discriminative on a larger scale. An evaluation was done on 20 image sequences captured over urban, suburban, rural with motorway, and industrial scenes by a consumer-grade camera mounted on an airplane, the Figure 1 shows the location of the scenes. In this test the *PLBGHashing* provided the shortlist, while the verification ranked the correct geolocation as the best match in most of the cases.

Our main contributions are: (1)We use car tracks to detect parts of the road network. This does not need a known ground sampling distance (GSD[3]) as standard road detectors. (2) We utilize road networks as appearance invariant features to localize an aerial image sequence over a large area. This also avoids the need for an image database over the search area. (3) We present a geometric hashing method to match partial line segments to line structures. This does not need a complete road detector which detects relations between roads as a graph, e.g. intersections.

2. Related work

The standard for creating georeferenced aerial images is using an accurate GPS and IMU and/or measuring Ground Control Points manually [15]. Other approaches utilize the idea to localize an image from only the image information also on a large scale. The queried image can be matched to a huge pool of landmark photos [32], while the 3D building information might also be utilized [3]. Li et al. [17] determine the absolute (world) camera pose using 3D point clouds. Mueller et al. [21] improve the absolute geometric accuracy of satellite images by automatically extracting Ground Control Points from existing orthorectified reference images. In Wu et al. [30] satellite images are localized with feature-based indexing, but not on a real large scale (maximal 16 km×12 km). In Lin and Medioni [19] UAV images are matched to georeferenced ortho images (map). But the search area is limited by manually labeling several correspondences between the first frame of the

[2]The OpenStreetMap is an open, crowd-sourced map with good coverage and fast updates.
[3]Ground Sample Distance, the distance between pixel centers measured on the ground.

UAV sequences and their corresponding satellite images.

All these methods require an already existing image database, and capturing and maintaining this is laborious. Using crowd-sourced image databases (e.g. *Flickr, Google Picasa)* this difficulty can be overcome. However, the landmark images are unevenly distributed and biased to touristic highlights. The work Lin et al. [18] addresses this problem by matching terrestrial image to remote sensing data, which also provide coverage over locations where no crowd-sourced data are available.

A new direction is the usage of *non-image* databases. Baatz et al. [4] employ the terrain map to geolocalize images from the mountain silhouette in the Alps.

Brubaker et al. [8] utilize the road network and visual odometry to locate a driving vehicle without any other information. They show that a long track on the road network is characteristic enough to obtain the geolocation of the car. Our method shares the idea of exploiting the pattern of the roads in the form of car tracks, but instead of a single long vehicle trajectory we use many short ones.

In Konzempel and Reulke [14] aerial images are orthorectified (projected on the Earth's surface) without an IMU. The orientation is initialized by an accurate GPS measurement and optimized by matching the detected streets in the image to the road network. In comparison to this method, our approach does not require an accurate position information, but only a search area, which might be as large as an entire metropolitan area.

In [25], road networks are represented as graphs, with road intersections as graph vertices. It is assumed that the intersections are detected correctly and the georegistration problem can be solved as a graph matching problem. In comparison to this work, in our approach the road intersections does not need to be detected in the image. We assume that there is no appropriate intersection detector or there might be no intersections in the image (just non intersecting roads).

In Wilson and Hancock [27] the graph structure of the road network (junctions and roads between them) in aerial image is extracted using a relaxational line-finder. Then the graph junctions are matched to junctions in the map using probabilistic relaxation. This method assumes that road junctions are present and they can be detected. In contrast, our method works without junctions, it needs only segments of the road network and we consider a much larger search area.

Li et al. [16] extract and match line segments instead of junctions. They define rotation and translation invariant features between the segments and use these pairwise features to calculate the cost of an assignment between aerial image and map. This combinatorial optimization problem is solved via continuous relaxation labeling. Solving this optimization problem is non trivial, specially for a large number of variables. In [16], the orientation of the image is used to limit the possible matches. In our problem we do not have access to this orientation information. We provide a more detailed analysis on this method in section 4.2.3.

Gros et al. [10] use local, geometric invariant features to match images related by similarity or affine tranformations. For similarity transformations they compute geometric invariants from pairs of line segments having an endpoint in common. The invariants are the angle between the two segments and the ratio of the length of the segments. These invariants are matched between the images, similarity transformations are computed from the matches and they are aggregated in a Hough-transform manner, i.e. clusters are searched in the parameter space. In contrast to this method, in our problem the segment length ratio invariant can not be applied. A track segment can lay anywhere within the map segment. Additionally, our tracks are mostly straight (or with a small curvature), the angle between the consecutive line segments is usually around zero and thus the angle invariant is not discriminative enough.

The COCOA system [1] addressed the tracking of moving objects in aerial videos. Xiao et al. [31] track cars in wide field of view aerial videos, however they need already georefenced images to leverage traffic flow priors from road maps.

3. Image sequence to road database matching

We extract the road traffic from the images and match the car tracks to the road network. Detecting the roads directly would be more straightforward, but this may pose problems due to the high intraclass variety of roads. The methods described in [26], [20] show good results, but they work on images with a constant GSD and the effect of an unknown GSD is not investigated. In our case there is no information about the GSD. Using only the car tracks also highlights that already a fraction of the road network is enough for the geolocalization.

The intraclass variety in visual appearance of cars is much smaller than that of roads. Thus existing robust object detection methods (e.g. Viola-Jones object detection framework [24]) can be applied with satisfying results (e.g. a detector based on boosting is presented in [12]). The features of the car detector are pixel intensity differences, which can be robust against different weather and lighting conditions. The motion information is also a strong cue, enabling object detection in cases where a non-moving object could not be distinguished from the background. We utilize the motion information by using the tracks of only the moving cars to reduce the possible false positive car detections. The length of the tracked cars in the image also gives information about the GSD. The proposed geolocalization method could also work with a suitable road detector (e.g. based on deep learning), or by labeling the roads manually and limiting the GSD search space.

3.1. Track extraction

The track extraction works on a mosaic image compiled from the single images. We use the tool *VisualSFM* [29] to calculate the camera parameters and 3D point coordinates in the scene. The earth surface is assumed a plane, and a homography is calculated between the images from the camera parameters and the equation of the plane.

We detect the cars independently on the image frames. We apply the Viola-Jones object detection framework [24] with the Gentle AdaBoost [9]. The detector is rotation variant, thus we rotate the image in steps to cover all directions and group the independent detections together (it would be more time efficient to use a multi-view detector).

The detections are transformed to the mosaic image. During the tracking we model the motion of the cars with a simple linear model and use Kalman-filtering. In each frame the tracked objects are assigned to detections matching the predicted object positions. We discard the short tracks to avoid false positives. This is a simple method, delivering satisfying results.

3.2. Matching the tracks to the road network

We formulate the aerial image geolocalization as a model recognition and pose estimation task of the road network pattern. The search area is tiled to overlapping square areas (in object recognition terms a tile is a model). An image scene query returns a model and a transformation between the image and the model. Since the absolute world position of the tile (model) is known, the transformation from the image coordinates to the geocoordinates can be recovered. The recognition has to handle geometric transformations for the pose estimation, a high number of models (> 10000) for larger search areas, and partially occlusions since the extracted tracks are just a subset of the road network. The geometric hashing method is suitable for handling these challenges.

3.2.1 Geometric hashing

Geometric hashing is an efficient, low polynomial complexity technique for matching geometric features against a databases of such features. Matching is possible even when the recognizable database objects have undergone transformations or when only partial information is present. We describe briefly the geometric hashing defined for $2D$ feature points. For a more detailed explanation the reader is referred to [28]. In the preprocessing step of geometric hashing a hash table is generated:

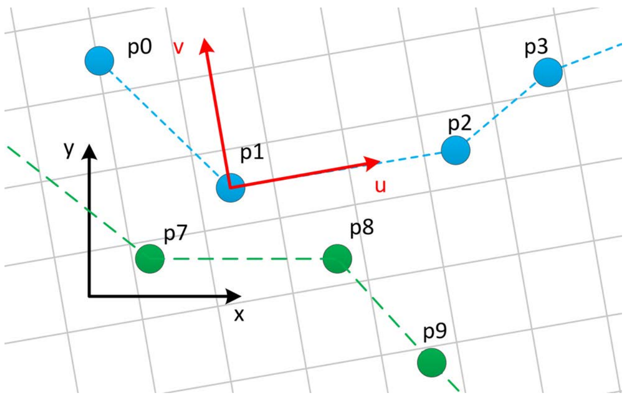
1. Calculate the feature points of the model.
2. For each ordered point pair (basis) calculate the remaining points in the coordinate frame defined by the basis.
3. For each remaining point coordinate in the basis frame quantize the coordinates to a grid. Use the grid coordinate as hash index and insert (into the hash table) an entry containing the identifier of the model and the basis used to generate the hash index.

The matching is done with the following steps:

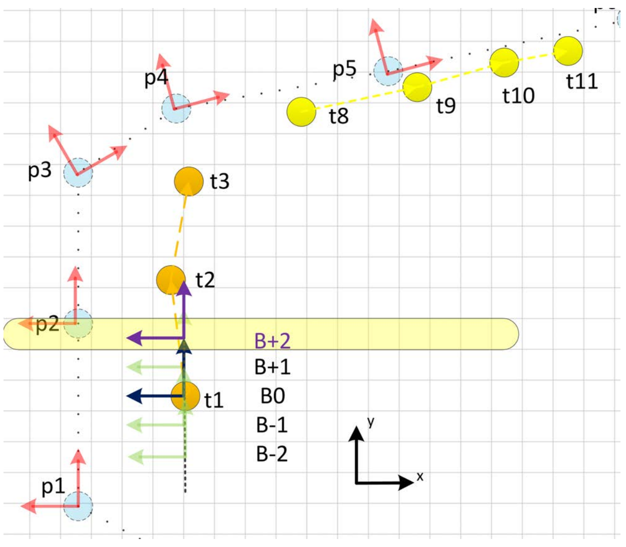
1. Calculate the feature points in the image to be matched.
2. For an arbitrary selected ordered point pair (basis) calculate the remaining points in the coordinate frame defined by the basis.
3. For each remaining point coordinate in the basis frame quantize the coordinates to a grid. Use the grid coordinate as hash index and cast a vote for all the models and the bases present at the hash index.
4. The models and bases with the most votes are match candidates. Select a candidate, recover the transformation between the query image and the reference model (based on the basis correspondence) and verify the transformation. If the verification fails, repeat this step (Step 4) with the next candidate.

Nakai et al. [22] were able to retrieve document images from a database of several thousand images in a fraction of a second. In Stein and Medioni [23] a curvature based geometric hashing is presented. In our case the most important feature is not the shape of the individual roads but the distances between the different roads, therefore we employ a geometric hashing similar as in [28].

Map and track representation The roads in the map are represented as 2D polylines. A polyline contains vertices (points), and the road consist of the line segments between consecutive vertices. The tracks are also polylines, whereas the vertices are the positions in the registered image during different frames. Although the representation of the tracks and roads is identical, there is no strict correspondence between the vertex positions in the road network and the tracks. A track segment can lie anywhere within the road segment.



(a)



(b)

Figure 2. (a): The blue and green polylines are different roads. The x-y is the original coordinate system of the road database. The line segment p1-p2 defines the basis (u,v) with the center point and the direction. The grid points intersected by line segments are the indices for the hash table. (b): The p points are the road vertices (from the database), the t points are the track vertices extracted in the mosaic image (with different color for different cars). The red arrows are the road bases in the hash table. The basis B0 from the track end points t1-t3 is shifted along the track. At B+2 position the track basis lies in the the same grid row, as the road basis.

Polyline Based Geometric Hashing The *PLBGHashing* method compiles the polylines to the hash table. Since we assume the earth's surface to be a plane and the tracks projected on this plane, the perspective transformation between the tracks and the roads can be reduced to a scaling, rotation and translation. As a basis we use a segment in a form of one point and a direction instead of two or more points. The relation of a $\mathbf{p}$ point's original coordinate (x, y) and the coordinate (u, v) in the basis defined by a line segment S with end points $\mathbf{p}_{s_1}, \mathbf{p}_{s_2}$ is described by the equation:

$$\mathbf{p} - \mathbf{p}_{s_1} = u\mathbf{p}_x^s + v\mathbf{p}_y^s \tag{1}$$

where $\mathbf{p}_x^s = \frac{\mathbf{p}_{s_1} - \mathbf{p}_{s_2}}{\|\mathbf{p}_{s_1} - \mathbf{p}_{s_2}\|}$ and $\mathbf{p}_y^s$ is $\mathbf{p}_x^s$ rotated by $90°$. On this basis the 2D hash table is invariant to rotation and translation but not to the scaling. To find the correct GSD we query with multiple scales. By leveraging the low variation of the car lengths $(4 - 6m)$, the number of different scales are kept low. We calculate the maximum in the histogram of the car lengths (the bounding rectangle delivered by the detector) in the tracks. Utilizing this length we needed only 3 scales to search in our experiments to get good geolocalization.

Each hash bin stores a set of entries. An entry contains a model number and a basis. It is important to store a set with unique elements, otherwise some bins could be biased by areas containing roads densely.

Database construction (off-line) The search area is tiled to squares with a fixed side length d_s and overlap. The nodes are interpolated to make the line segments shorter as a defined maximal value d_{max}. Then the hash table is generated by the following steps:

For each polyline (road) P_j in each tile M_i do:

1. For each line segment S_k in the polyline P_j:

(a) Create an entry E_k with the model number i and the basis of S_k.

(b) Solve the (Eq. 1) with S_k to calculate (u_m, v_m) of all the vertex points in M_i.

(c) Quantize the points (u_m, v_m) to a grid (u_m^q, v_m^q).

(d) Calculate the coordinates (u_l^q, v_l^q) of the rasterized line in the grid between the neighboring $(u_{m-1}^q, v_{m-1}^q), (u_m^q, v_m^q)$ vertices. The (u_l^q, v_l^q) line points are indices for the 2D hash table. For each line point (u_l^q, v_l^q) add the entry E_k to the set at the (u_l^q, v_l^q) hash bin.

The Figure 2a illustrates the hash index generation. The complexity of the number of bases is $O(n)$, where n is the number of node points after interpolation in the tile. The complexity of hash table entries per basis is $O(l/d_g)$ where l is the sum of road length and d_g is the grid size for geometric hashing. Since d_g is fixed in an implementation the complexity of hash entries is $O(nl)$.

Shortlist retrieval (on-line) To find the correct geolocation of a scene, a basis (1-point and direction) in the tracks has to be aligned to a similar basis in the correct model with the correct scale. The direction of a road segment can be properly recovered from the end points of a track. We choose long straight tracks, because they define the direction more accurately if the vertex positions are noisy. The correct scale is searched with multiple queries. The list of scales is derived from the car length. Since the line segment length is limited by d_{max}, a street vertex exists in the range $[-d_{max}/2, d_{max}/2]$ along a track vertex, the Figure 2b illustrates this. It is enough to search this range in shifts by the grid size d_g, because then the hash indices are the same at creation and query. We resize the scene with all the search scales. For each resized scene we choose track segments T_i (the line segment between the end points). For each T_i in both directions we do:

1. Shift T_i between $[-d_{max}/2, d_{max}/2]$ in d_g steps.
2. Each shift gives a line segment S_k. For each S_k:
   (a) Solve the (Eq. 1) with S_k to calculate (u_m, v_m) of the vertex points of all the tracks in the scene.
   (b) For each polyline (track) P_i in the scene:
      i. Calculate the rasterized points (u_m^q, v_m^q) of the vertices
      ii. Calculate the coordinates (u_l^q, v_l^q) of the discrete (rasterized) line segment in the grid between the $(u_{m-1}^q, v_{m-1}^q), (u_m^q, v_m^q)$ points.
      iii. Each point (u_l^q, v_l^q) of the line segment is an index for the hash table. For every entry at the hash bin (u_l^q, v_l^q) cast a vote for a recognition (model, basis and scale).

After the query with multiple straight tracks, the votes are aggregated. One recognition gets only one vote from a bin, otherwise a basis might be biased due to many tracks over the same index. A shortlist of matches is created by sorting the recognitions based on the number of received votes. The transformation from image to a world coordinate can be recovered from the basis in the image and in the model, the scale and the world position of the model (tile).

3.3. Verification of the shortlist

The retrieved shortlist also contains false recognitions. We re-sort the shortlist by a stricter criteria, the verification match score S. The quality of the match is measured by the mean distance d_{m_t} of the tracks R_t to the roads in the model, like the Chamfer Matching [5]. The distance image I_d is calculated to the roads [7], and the I_d pixels along the tracks are extracted. Since the tracks may contain outliers, we calculate a trimmed mean, where the 10% of the tracks with the highest distances are discarded. The d_{m_t} does not penalize roads not covered by tracks. To compensate for this, we extract a second road detector with high completeness but low correctness. We assume that the color of the roads is similar all over the scene. The CIELUV color values of the image pixels are grouped to 16 clusters by k-means. Clusters are added to the road class until 85% of the tracks lie over pixels belonging to the road class. A second d_{m_p} mean distance is calculated as the distance of the roads in the model to the roads detected by the pixel based detector (R_p). The Figure 3 shows the distance images, a brighter pixel is a larger distance.

We assign likelihood values L_t (R_t match), and L_p (R_p match) to the two mean distances d_{m_t} and d_{m_p}. The likelihood function $f(x)$ of d_{m_i} for a correct match are modeled by the following function, a normal distribution function $\mathcal{N}(\mu, \sigma)$ if $x \geq \mu$, uniform with the max of the normal distribution if $0 \leq x < \mu$ and 0, otherwise. The normal distribution parameters are set experimentally, for R_t; $\mu_t = 6$m, $\sigma_t = 15$m, for R_p; $\mu_p = 7.5$m, $\sigma_p = 15$m. We set μ_t smaller than μ_p because

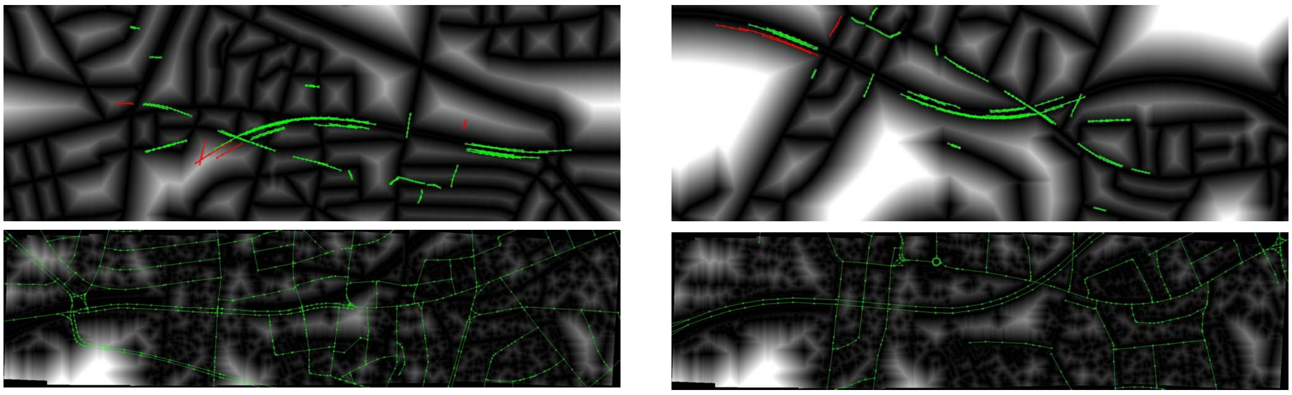


Figure 3. The distance images used for calculating the verification score at *Suburban 1* image sequence. On top the tracks are shown with green (inlier) and red (outliers) in the distance image to the road network. In the bottom the road network (green) is shown in the distance image to classified road pixels in the image.

for d_{m_t} outliers are considered. The likelihood for a mean distance value is $L_d = f(d_m)$. The final score S that the image sequence I matches the roads R is the joint likelihood $S = L_t L_p = f(d_t)f(d_p)$, d_t and d_p are handled as independent. At a correct match both likelihoods L_t and L_p have a high value, thus the final score is also high. For a false match at least one of the likelihoods has to decrease, thus the final score also decreases.

4. Experimental results

4.1. Synthetic data

The purpose of this test was to investigate whether the road map tiles (scenes) and the subsets of the roads in these tiles are unique or ambiguous over a city. We analyzed if the proposed *polyline based geometric hashing* method can distinguish the road network fractions over a city area, thus enabling geolocation recognition. Since the tracks are considered as subsets of the street network, we can generate synthetic tracks with known ground truth from the street map data.

4.1.1 Test steps

- *Database creation:* We generate the hash entries for the search area A_s with a specific grid size as described in 3.2.1. Except the road nodes are not interpolated to limit the segment length to d_{max} since they are virtually identical to the vertices of the tracks.

- *Synthetic track generation:* In the search area A_s an axis aligned square (scene) A_r is extracted with side length d_s. In this A_r area either all streets are used or they are sampled. When sampling roads and line segments are randomly chosen with a ratio parameter defining how many roads r_r and segments r_s are kept. The remaining vertex points are randomly shifted in the line segment. The tracks are transformed by one random 2D rigid transformation T_r to a new area A_t. Since we shift the vertex positions only in the line segments (no noise is added, but the quantization of the geometric hashing can be considered as noise), a perfect location recognition is possible. The question is the ambiguity of the recognition.

- *Geolocation recognition:* A recognition as in 3.2.1 defines a $2D$ rigid transformation T_b. If the T_b transforms the corners of A_t back to their original location A_r the recognition is correct. The geolocation recognition is considered true only if it is correct, it has the highest number of votes, and no wrong recognition has the same number of votes – thus it is unambiguous.

4.1.2 Test configurations and results

We selected two search areas, one in New York City, US with an area of $10.1 \times 11.1\ km^2$ containing 1495 km of streets. The second in Munich, Germany, with an area of $14.1 \times 10.6\ km^2$ and 1560.4 km road length. See the map of the areas[4] on figure 4.

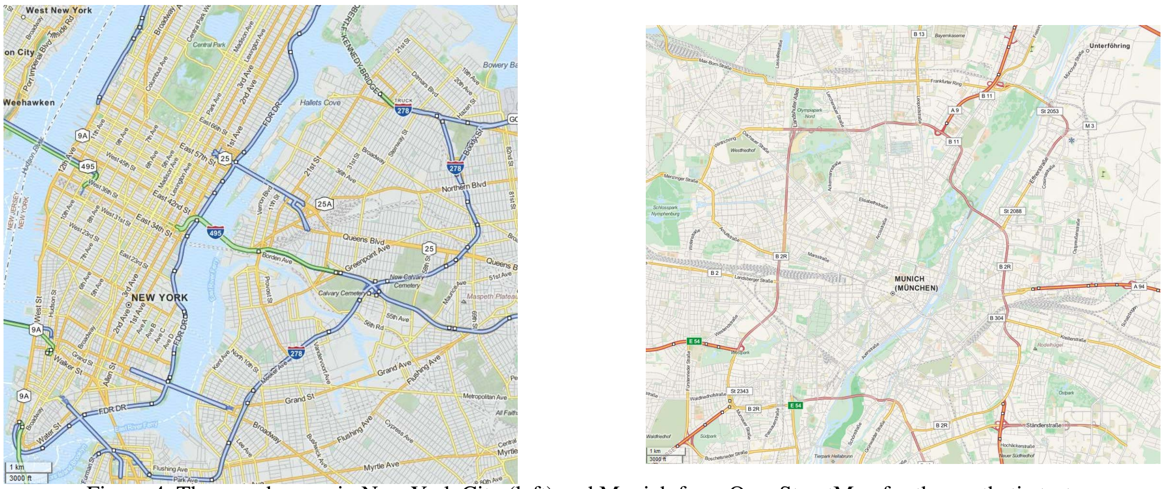
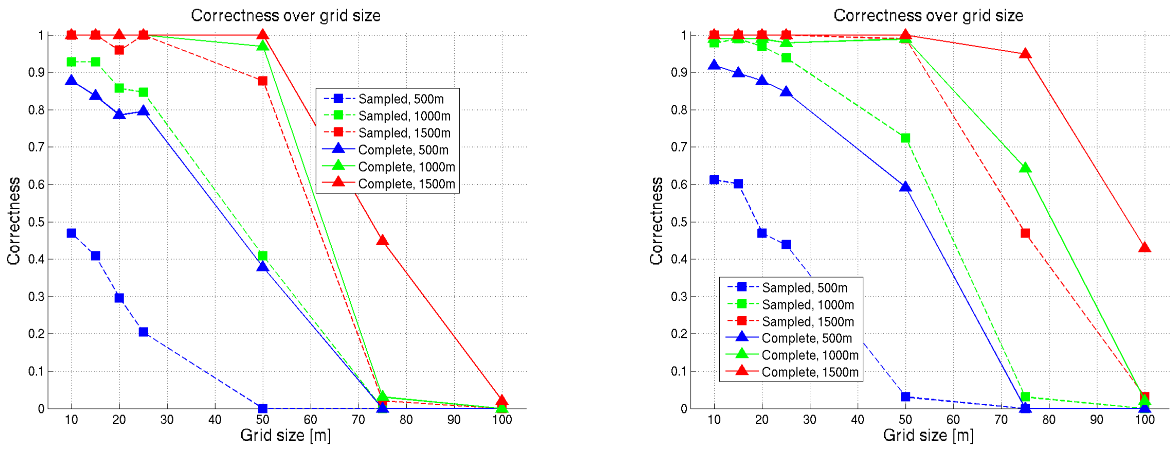


Figure 4. The search areas in New York City (left) and Munich from OpenStreetMap for the synthetic test.



New York Munich

Figure 5. Correctness over grid size with different scene sizes and complete/sampled road network.

Grid Size [m]	10	15	20	25	50	75	100
New York	502.1	323.6	234.4	181.7	79.9	47.4	30.5
Munich	622.0	401.8	292.5	228.4	105.1	64.8	44.2

Table 1. Number of hash entries $[10^6]$ in the database over the grid size.

The tiles of the road database are 2500 m$\times$ 2500 m with 50% overlap. We tested multiple grid sizes; see the table 1 for the values. 100 center points were generated randomly. Around each point scenes were generated with side lengths $d_s = 500, 1000, 1500$ m. We analyzed the complete scene scenario and the more realistic case by sampling the roads with rate $r_r = 0.5$, the line segments with $r_s = 0.3$ and shifting the node along the line segment with maximal 50 m.

[4]source: http://www.openstreetmap.org

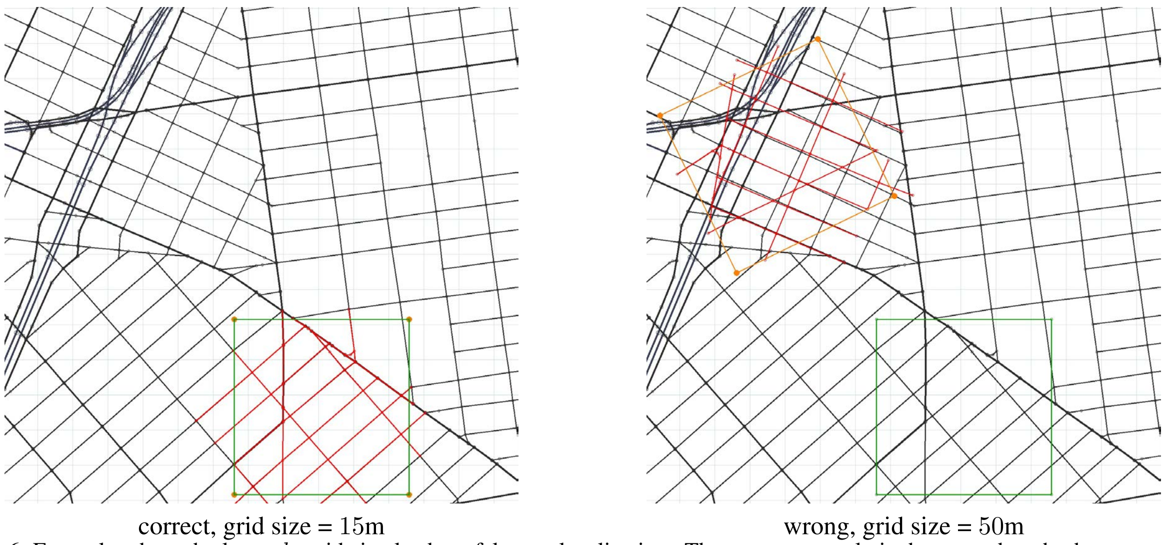


Figure 6. Example where the large d_g grid size leads to false geolocalization. The green rectangle is the ground truth, the orange is the recognized geolocation. The road network is in black, the synthetic tracks are in red. The grid in the image is 100 m. The tracks exceed the scene to preserve the basis points.

We computed the *correctness* $= T/(T+F)$ where T is the number of true and F is the number of false geolocalizations. On figure 5 the correctness was plotted over the grid size for the two cities with different sampling and different scene sizes $d_s \times d_s$.

Since the grid size of the geometric hashing both defines the number of bins and the rasterization of the line segments, it has an impact on the number of hash entries in the database. The number of hash entries over different grid sizes are in table 1. More hash entries imply higher memory demand and longer database retrieval and creation time.

The correctness values derived from our test indicate, as shown in figure 5, that the road network of a larger area is unique, though it might be ambiguous on local scale. The problematic areas occur where the whole street network is very regular like a chessboard or the scene contains a few straight roads. The unambiguity originates not simply from the perpendicular Manhattan World road network, typical in New York, but also from the identical distances between the streets. On a larger area it is more likely that a unique road will appear, which makes the whole road network also unique. The Figure 6 shows an example where the coarser grid size leads to wrong localization.

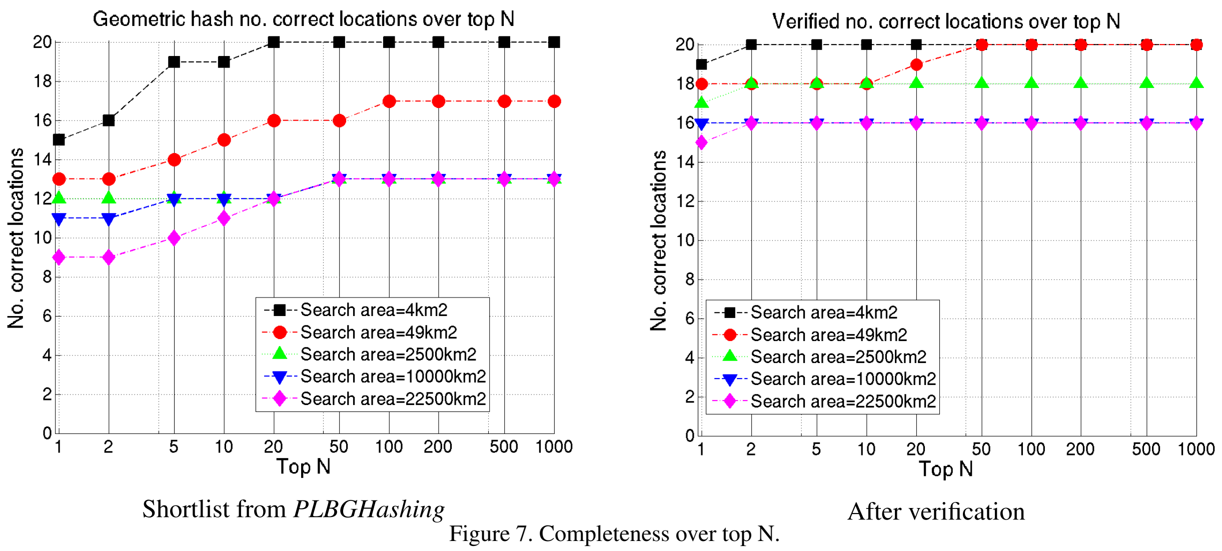
By increasing the scene size (i.e. flying longer over roads) a correct geolocalization becomes possible based only on the road map. With a grid size of 15 m and scene size 1000 m $\times$ 1000 m we get a correctness > 0.9 for all the scenarios and cities we have tested.

4.2. Real data – Aerial image sequences

We have tested the method on real aerial photos over Germany. 20 image sequences were acquired by a 21 MPixel Canon DSLR camera with a fixed focal length mounted in nadir direction on an airplane. The internal camera parameters were known. A GPS device registered the position of each image for the ground truth. Each image sequence contains $17-20$ consecutive images taken with approximately 1 Hz frequency. The altitude over ground was between 1000 and 1500 m, which gives a GSD of $13-19$ cm/pixel. The image sequences contain roads with car traffic. We grouped the scenes as urban (U), suburban (S), industrial (I) and motorway (M). This classification is not strict; we arbitrarily used it for this paper. The scenes were relatively flat, thus the earth surface could be modeled by a plane. Studying mountainous areas is planned for future work. The scenes *Suburban* 1 and 4, 5 and 2, 6 and 3 and *Urban* 3 and 4 are approximately over the same area but with two month difference. The original images were rectified with a 2 parameter radial distortion model. All the processing was done on these rectified images. The tiles of the road database were 2000 m $\times$2000 m with 50% overlap. The grid size d_g for the geometric hashing was 15 m. The 5 longest straight tracks in the scene were used for the shortlist retrieval. The size of the scenes was around 2000 m $\times$ 500 m.

4.2.1 Quantitative evaluation

The *PLBGHashing* delivers a shortlist of locations, while the verification gives a ranking of these locations. To decide if a geolocation was correct, the geolocalization was compared to the GPS values. If the GPS positions were correct, the mosaic image was inspected visually by overlaying it in Google Earth. We extracted a TOPN value, which indicates if there is a correct location in the first N list elements. The number of possible locations and orientations is very high even for a small search area. For a scene of 1000 m $\times$ 1000 m in a search area of 2 km $\times$ 2 km with 15 m translation steps, 5° angle steps, and 3 scales $(\frac{1000}{15})^2 \times \frac{360}{5} \times 3 = 960000$. A search area of 22500 km^2 consists of 22201 overlapped tiles, thus the total number of possibilities is in the 10^{10} scale.



Shortlist from *PLBGHashing* After verification

Figure 7. Completeness over top N.

In Figure 7 the aggregated number of correct localizations is plotted for different N values and search areas. There is a separate plot for the shortlist retrieval and the verification. The increase of Top1 by the verification was significant. We increased the search areas around the GPS position of the camera in steps 4, 9, 25, 49, 100, 225, 400, 625, 1225, 2500, 5625, 10000, 15625, 22500 km^2. The Figure 8 shows for each scene the maximal size of the search area in which it was correctly located in the topN after the verification.

19 scenes out of 20 could be located correctly as top1. The scenes *Industrial 1, 2* contain tracks acquired in large parking lots (the tracking algorithm might extract wrong tracks on dense parked cars) outside the road network. If the number of outliers becomes greater than what the verification accepts, then the verification ranks the locations wrong. Thus even if the correct location was in the shortlist it can not be found. Therefore bad localization. The location of *I1* was wrong already at the smallest search area, while the *I2* could be localized only on 4 km^2. *I2* contains more tracks over streets as *I1*, which has tracks only about one motorway and and industrial area, see the Figure 11. The other scenes show that if we have enough tracks on the roads, then the localization only weakly depends on the search area. On the Figure 10 and 11 we show the geolocalizations by projecting the mosaic images into the street network and as an overlay in Google Earth (the visualization as an image overlay is geometrically not fully accurate).

Some areas might be inherently ambiguous (e.g. chessboard like streets). An indicator for this ambiguity is if there are many candidates with the same number of votes in the shortlist and in the list after validation.

We defined the grid size for the geometric hashing as 15 m. This makes the method robust to small displacements in the tracks and it can handle multi-lane roads. If we assume 3 m wide lanes, then 5 lanes are quantized to the same grid (i.e. the localization is invariant to this displacement). Roads with more then 5 lanes are sparse as large roads (e.g. motorways) in OSM are stored as two separate roads for each direction.

Our method is agnostic to splits or merges of the tracks as the assignment between line segments and tracks is not relevant. A track consisting of 10 line segments has the same effect as if it is split to 10 individual tracks.

The search area 150 km $\times$ 150 km around the *Urban* scenes contain nearly 32000 km of roads, compiling to 2262M entries in the hash which requires 30500 MB memory. For the *Urban 1* scenes containing many tracks the retrieval took 9 minutes on the largest search area, for a scene with less tracks the retrieval is faster, for the *Suburban 1* it took 2 minutes. The

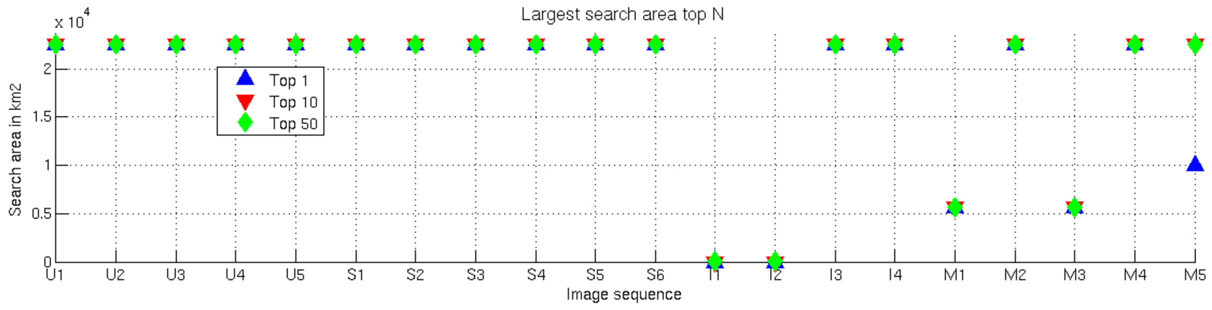


Figure 8. The largest search area for each image sequence where the correct verified geolocation was in the TOPN in km^2

average accuracy of the geolocalization is around 25 m, this also depends on the accuracy of the road data (A more accurate location could be acquired by optimizing the resulting location, but this was not our goal in this paper). Our implementation was written in C++, multi threaded. The experiments run on an Intel Xeon E5-1650v2 processor with 6 cores. The verification took around 36 ms per shortlist element. Since the geometric hashing is very suitable for parallel implementation, the retrieval time on large search areas can be significantly reduced by utilizing more cores or machines.

4.2.2 Comparison to simple chamfer matching

We consider chamfer matching [5] with brute force search as a baseline. Without the *PLBGHashing* for shortlist generation, by only a brute force search and verification of each position the number of required verifications would be very high. Since the processing time of the verification is relatively long (36 ms per location), it would take $960000 \times 0.036s = 9.6$ hours on 2 km $\times$ 2 km search area $10^{10} \times 0.036s = 3.6 \times 10^8 s = 11$ years for the largest search area (on a single thread). This is too long for practical usage.

4.2.3 Comparison to graph matching

To compare our method to graph matching, we reimplemented the method in [16]. We formulated this bipartite graph matching problem with many-to-one matches as one of inference in a Markov random field (MRF). The variables (nodes) are the segments in the tracks, while the states are the segments in the map plus a state defining no match. We define pairwise potentials between all variables. The pairwise potential consists of the sum of differences between 3 binary features (i.e. the relative orientation between lines, the minimum distance between the line endpoints and the distance between the segment center points). The energy function of this MRF is similar as the (2) energy function of [16]. The number of variables is T, the number of pairwise connections is $T(T-1)/2$ where T is the number of track segments. Each variable has $M+1$ states (where M is the number of map segments), thus the pairwise potential is a matrix of $M+1 \times M+1$ values. The MAP inference in this MRF gives the matching between the tracks and the maps. The inference in this MRF is an NP-hard problem as it contains many loops and the pairwise potential can not be guaranteed to be regular. Therefore we have to perform approximate inference. We refer the reader to paper of Kolmogorov and Zabin [13] for the conditions of an MRF inference problem being NP-hard. We applied the AStar inference algorithm [6] of the OpenGM software [2].

We have one difference to the method in [16]. [16] has an approximation for the initial direction of the image and therefore they define possible matches between line segments only to those within a given orientation difference. This can reduce the number of states per variable significantly. For our problem this is not the case, since we have no information about the orientation of our images. We have to match all track segments to all possible map segments.

We used synthetic tracks (the segments of the map randomly transformed) on a limited search area (max 700×700 m^2) to validate if our implementation can provide correct localizations. As long as both the number of map segments (labels) and the track segments (variables) were kept low (< 200) the approximate MAP inference provided a decent segment assignment and thus correct localization. But if we moved to larger search areas (e.g. 1000×1000 m^2) or real tracks which consist of more then 200 segments and has outliers, the inference could not find a solution in a reasonable time (30 minutes) and the memory need for the algorithm became also too high for practical use. We show an example for this on Figure 9.

The increase of T and M leads to a difficult inference problem and the memory consumption becomes also a problem, since the number of values stored for pairwise features is $(M+1)^2 T(T-1)/2$. If $T = 200$ and $M = 1000$ we already need

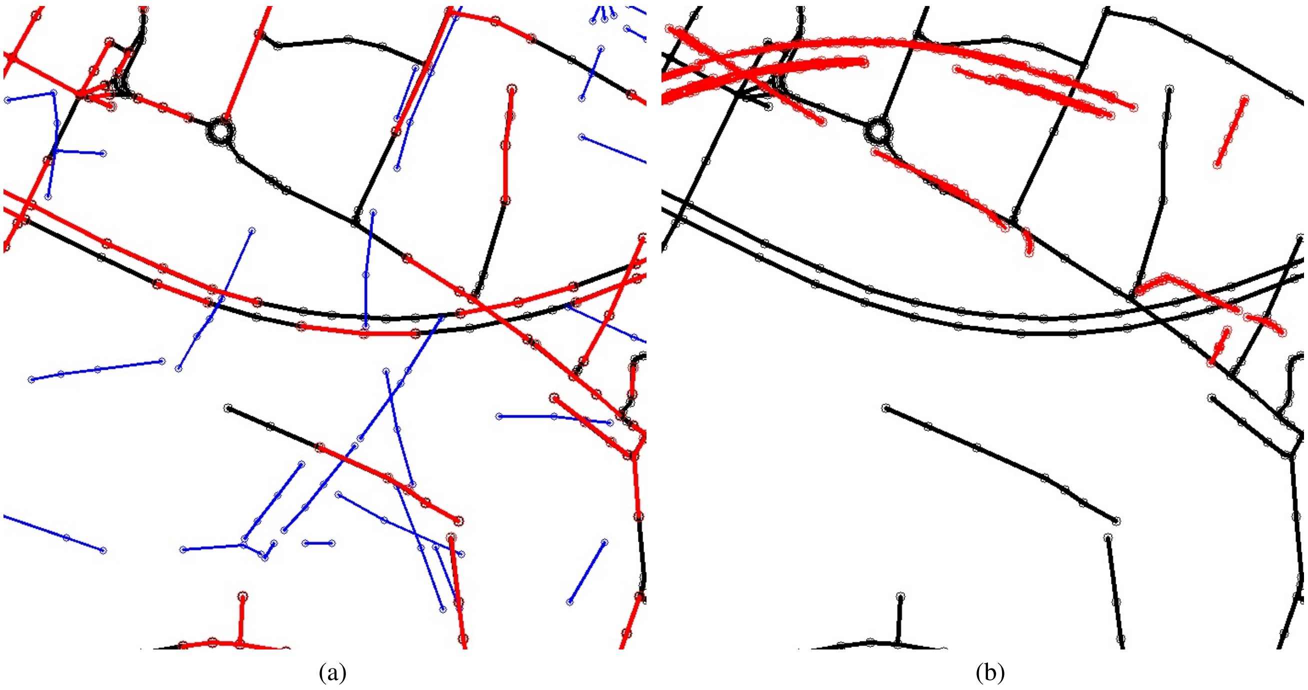


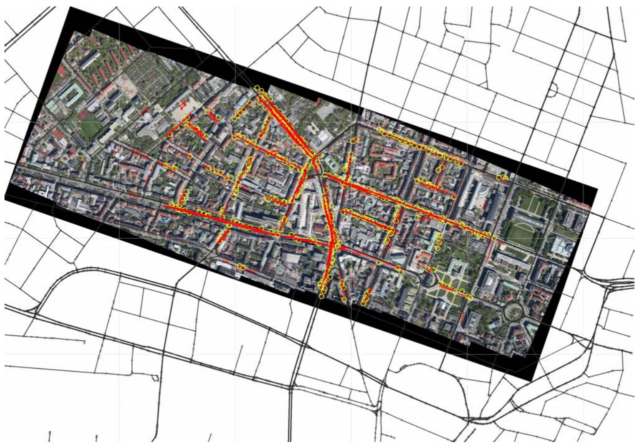
Figure 9. Graph matching based localization [16] on the *Suburban 1* sequence using $700 \times 700m^2$ search area. The (a) shows the case with synthetic tracks (random sampling of the road network). Here we have 110 variables each with 174 states (the number of map segments). The inference gave an assignment between the segments which define a correct localization. The map segments are shown in black, the sampled road segments transformed in blue and the located segments in red. On (b) we use real tracks defining 353 variables. The inference could not find a matching defining a correct geolocalization in reasonable time (30 minutes). The localized tracks after 30 minutes inference running time are shown in red.

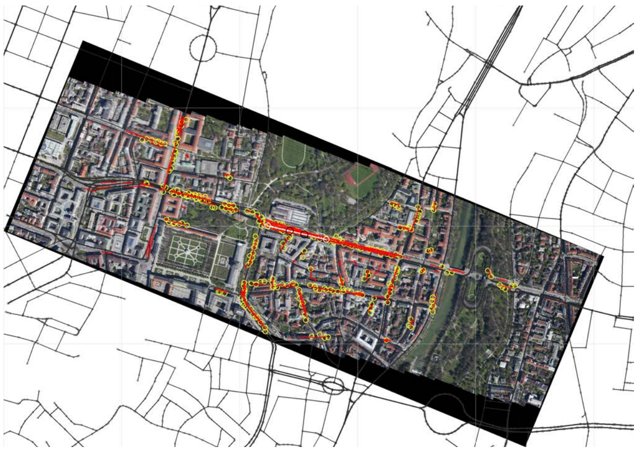
to store $\approx 2 \times 10^{10}$ values.

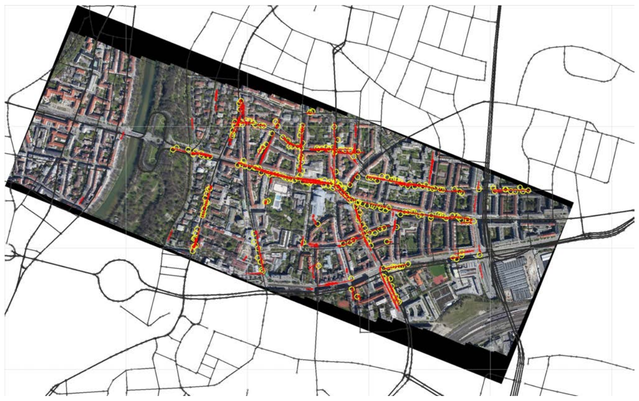
Graph matching as in [16] provides an energy formulation giving a matching invariant to rotation and translation. However the minimization of this energy function is intractable for large scale problems with many variables and states. This prevents the practical application for large scale geolocalization.

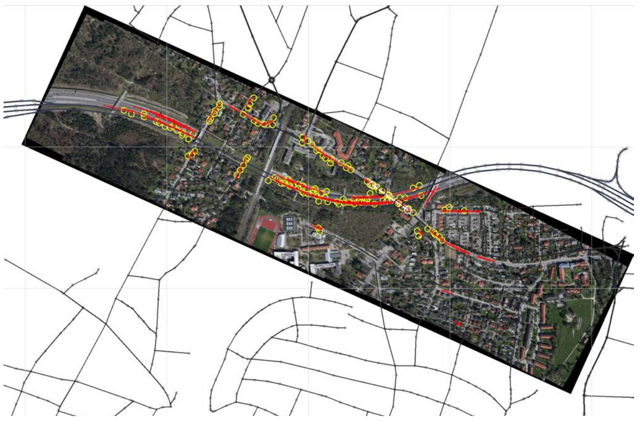
5. Conclusion and outlook

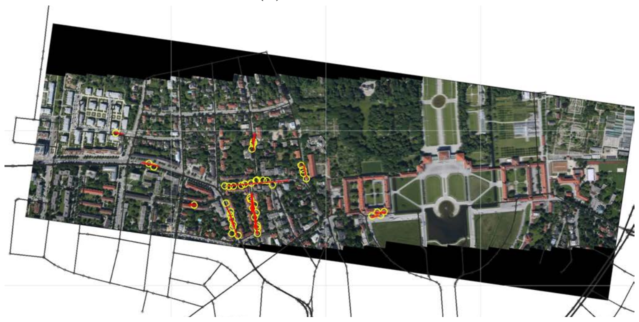
In this paper we have addressed the problem of finding the geolocation of an aerial image sequence over a larger area. Our method utilizes only the visual information and the road map which is easily accessible. We have demonstrated the effectiveness of our method in tests with synthetic data derived from the road network and real photos captured over various scenes (urban, suburban, industrial, motorway). In practice much longer image sequences could be used covering more roads, thus making the localization easier. The limitation is the presence of road traffic at the area and the flat ground assumption. In our future work we plan to address these limitations, e.g. by applying a road detector the method could localize scenes without traffic and also single images, a 3D model of the scene generated by dense stereo could extend the method to areas with relief.


(a) *Urban 1*.


(b) *Urban 2*.


(c) *Urban 3*.


(d) *Suburban 1*.


(c) *Suburban 5* (90° rotated).

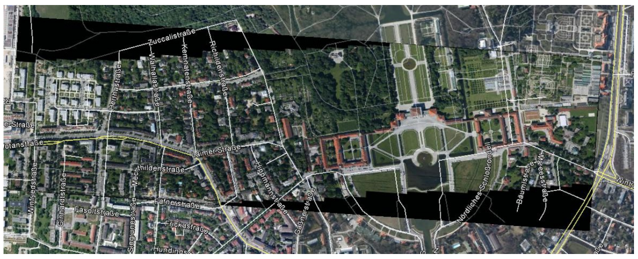
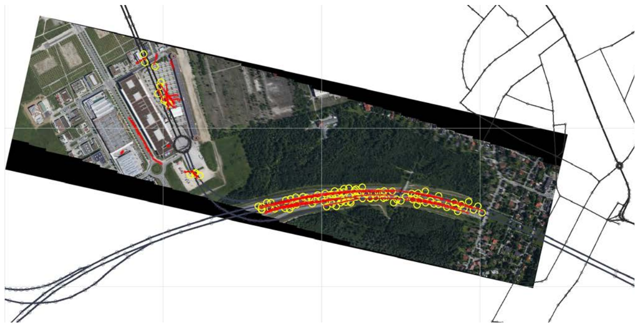
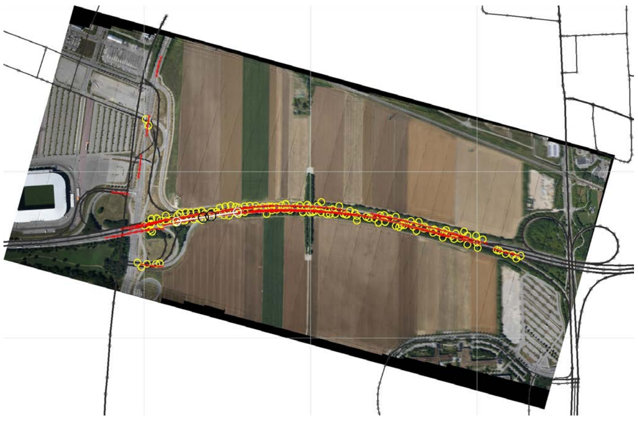

Suburban 5 an an overlay in Google Earth

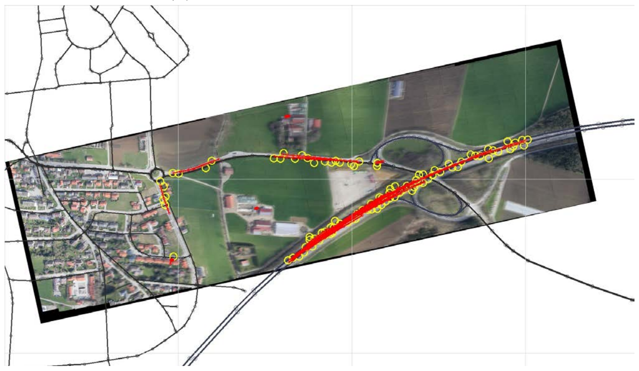
Figure 10. Geolocalization results. The mosaic images are projected into the street network. The roads are in black, the tracks are highlighted with red lines, the locations of votes from the *PLBGHashing* are marked with yellow circles. (When an image scene is only partially overlapped with a model during the shortlist retrieval, there are no vote circles at every matching track-road locations (e.g. on *Urban 1-3*)) On (f) the geolocalized mosaic image with the tracks is overlaid in Google Earth. All these scenes were correctly located on a 22500 km^2 search area.



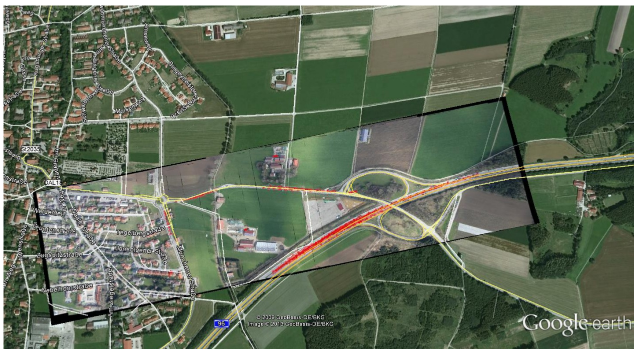
(a) *Industrial 1* false location.



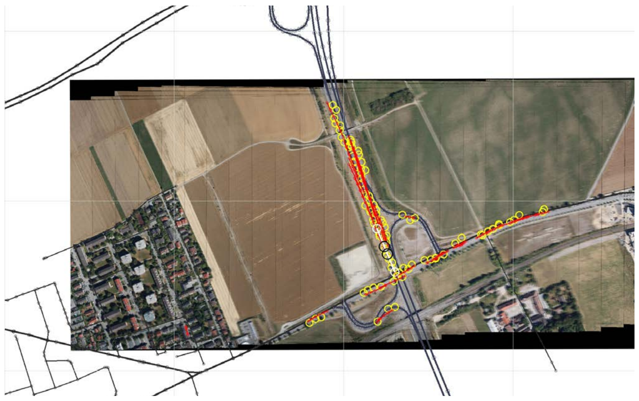
(b) *Industrial 3.* (90° rotated)



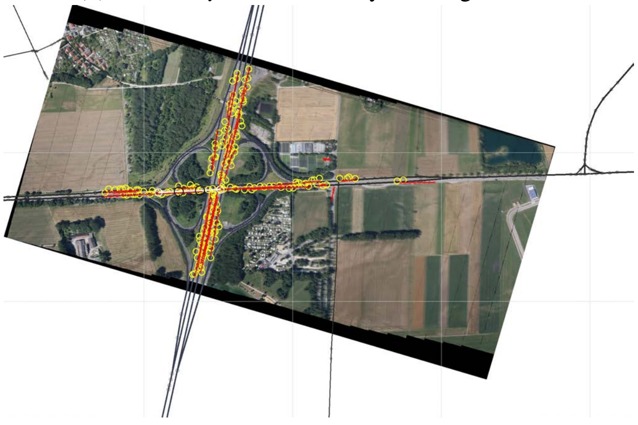
(c) *Motorway 2*.



(d) *Motorway 2* as an overlay in Google Earth.



(e) *Motorway 3*.



(f) *Motorway 5* (90° rotated).

Figure 11. Geolocalization results. The mosaic images are projected into the street network. The roads are in black, the tracks are highlighted with red lines, the locations of votes from the *PLBGHashing* are marked with yellow circles. On (d) the geolocalized mosaic image with the tracks is overlaid in Google Earth. The (a) shows the scene *Industrial 1* located false on a 4 km^2 search area. The causes for this are; the scene contains 2 roads which makes it inherently ambiguous, multiple tracks are extracted on a parking lot, outside of the road network. Since there are more tracks in the parking lot than on the road, the verification matching handles the correct tracks as outliers. The color based road detection also does not work properly, since a parking lot has similar color as the road. The error is approximately 100m.

6. Acknowledgement

This work was funded by the *Vabene++*[5] project of the German Aerospace Center (DLR).

References

[1] Saad Ali and Mubarak Shah. Cocoa - tracking in aerial imagery. In *Proc. Int. Conf. on Computer Vision*, 2005. 3

[2] B. Andres, Beier T., and J. H. Kappes. OpenGM: A C++ library for discrete graphical models. *ArXiv e-prints*, 2012. 11

[3] Georges Baatz, Kevin Köser, David Chen, Radek Grzeszczuk, and Marc Pollefeys. Leveraging 3d city models for rotation invariant place-of-interest recognition. *Int. J. Comput. Vision*, 96(3):315–334, February 2012. 2

[4] Georges Baatz, Olivier Saurer, Kevin Köser, and Marc Pollefeys. Large scale visual geo-localization of images in mountainous terrain. In *Computer Vision ECCV 2012*, Lecture Notes in Computer Science, pages 517–530. Springer, 2012. 3

[5] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric correspondence and chamfer matching: two new techniques for image matching. In *Proceedings of the 5th international joint conference on Artificial intelligence - Volume 2*, IJCAI'77, pages 659–663, San Francisco, CA, USA, 1977. Morgan Kaufmann Publishers Inc. 6, 11

[6] Martin Bergtholdt, Jörg Kappes, Stefan Schmidt, and Christoph Schnörr. A study of parts-based object class detection using complete graphs. *International Journal of Computer Vision*, 87(1):93–117, 2009. 11

[7] Gunilla Borgefors. Distance transformations in digital images. *Comput. Vision Graph. Image Process.*, 34(3):344–371, June 1986. 6

[8] Marcus A. Brubaker, Andreas Geiger, and Raquel Urtasun. Lost! leveraging the crowd for probabilistic visual self-localization. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3057–3064, 2013. 3

[9] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998. 4

[10] Patrick Gros, Olivier Bournez, and Edmond Boyer. Using local planar geometric invariants to match and model images of line segments. *Computer Vision and Image Understanding*, 69(2):135 – 155, 1998. 3

[11] Achim Hornbostel, Manuel Cuntz, Andriy Konovaltsev, Gtz C. Kappen, Christian Httich, Carlos A. Mendes da Costa, and Michael Meurer. Detection and suppression of ppd-jammers and spoofers with a gnss multi-antenna receiver: Experimental analysis. In *European GNSS Conference*, 2013. 1

[12] S. Kluckner, G. Pacher, H. Grabner, H. Bischof, and J. Bauer. A 3d teacher for car detection in aerial images. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, 2007. 3

[13] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159, Feb 2004. 11

[14] K. Kozempel and R. Reulke. Camera orientation based on matching road networks. In *Image and Vision Computing New Zealand, 2009. IVCNZ '09. 24th International Conference*, pages 237–242, 2009. 3

[15] K. Kraus, I. Harley, and S. Kyle. *Photogrammetry: Geometry from Images and Laser Scans*. Number Bd. 1 in De Gruyter textbook. Bod Third Party Titles, 2007. 2

[16] StanZ. Li, Josef Kittler, and Maria Petrou. Matching and recognition of road networks from aerial images. In G. Sandini, editor, *Computer Vision ECCV'92*, volume 588 of *Lecture Notes in Computer Science*, pages 857–861. Springer Berlin Heidelberg, 1992. 3, 11, 12

[17] Yunpeng Li, Noah Snavely, Dan Huttenlocher, and Pascal Fua. Worldwide pose estimation using 3d point clouds. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision ECCV 2012*, volume 7572 of *Lecture Notes in Computer Science*, pages 15–29. Springer Berlin Heidelberg, 2012. 2

[18] Tsung-Yi Lin, Serge Belongie, and James Hays. Cross-view image geolocalization. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 891–898, 2013. 3

[19] Yuping Lin and G. Medioni. Map-enhanced uav image sequence registration and synchronization of multiple image sequences. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–7, 2007. 2

[20] Volodymyr Mnih and Geoffrey E Hinton. Learning to label aerial images from noisy data. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 567–574, 2012. 3

[21] Rupert Müller, Thomas Krauß, Mathias Schneider, and Peter Reinartz. Automated Georeferencing of Optical Satellite Data with Integrated Sensor Model Improvement. *Photogrammetric Engineering and Remote Sensing*, 71(1):61–74, 1 2012. 2

[5] http://www.dlr.de/vabene/

[22] Tomohiro Nakai, Koichi Kise, and Masakazu Iwamura. Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval. In *In Lecture Notes in Computer Science (7th International Workshop DAS2006)*, pages 541–552. Springer, 2006. 4

[23] F. Stein and G. Medioni. Efficient two dimensional object recognition. In *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, volume i, pages 13–17 vol.1, 1990. 4

[24] Paul Viola and Michael J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, May 2004. 3, 4

[25] Caixia Wang, A Stefanidis, and P. Agouris. Relaxation matching for georegistration of aerial and satellite imagery. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 5, pages V – 449–V – 452, Sept 2007. 3

[26] Jan D. Wegner, Javier A. Montoya-Zegarra, and Konrad Schindler. A higher-order crf model for road network extraction. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1698–1705, 2013. 3

[27] R Wilson and E R Hancock. Relaxation matching of road networks in aerial images using topological constraints. *Sensor Fusion*, 2059:444–455, 1993. 3

[28] H.J. Wolfson and I. Rigoutsos. Geometric hashing: an overview. *Computational Science Engineering, IEEE*, 4(4):10–21, 1997. 4

[29] Changchang Wu. Towards linear-time incremental structure from motion. In *3DTV-Conference, 2013 International Conference on*, pages 127–134, 2013. 4

[30] Changchang Wu, Friedrich Fraundorfer, Jan-Michael Frahm, Jack Snoeyink, and Marc Pollefeys. Image localization in satellite imagery with feature-based indexing. In *Commission III, ISPRS Congress 2008 Beijing*, volume XXXVII, pages 197–202, 2008. 2

[31] Jiangjian Xiao, Hui Cheng, H. Sawhney, and Feng Han. Vehicle detection and tracking in wide field-of-view aerial video. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 679–684, June 2010. 3

[32] Yan-Tao Zheng, Ming Zhao, Yang Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T.-S. Chua, and H. Neven. Tour the world: Building a web-scale landmark recognition engine. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1085–1092, 2009. 2

C Máttyus, G., Wang, S., Fidler S. and Urtasun R., 2015. Enhancing Road Maps by Parsing Aerial Images Around the World. 2015 IEEE International Conference on Computer Vision (ICCV), pages 1689–1697.

http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7410554

Enhancing Road Maps by Parsing Aerial Images Around the World

Gellért Máttyus
Remote Sensing Technology Institute
German Aerospace Center
gellert.mattyus@dlr.de

Shenlong Wang, Sanja Fidler and Raquel Urtasun
Department of Computer Science
University of Toronto
{slwang, fidler, urtasun}@cs.toronto.edu

Abstract

In recent years, contextual models that exploit maps have been shown to be very effective for many recognition and localization tasks. In this paper we propose to exploit aerial images in order to enhance freely available world maps. Towards this goal, we make use of OpenStreetMap and formulate the problem as the one of inference in a Markov random field parameterized in terms of the location of the road-segment centerlines as well as their width. This parameterization enables very efficient inference and returns only topologically correct roads. In particular, we can segment all OSM roads in the whole world in a single day using a small cluster of 10 computers. Importantly, our approach generalizes very well; it can be trained using only 1.5 km^2 *aerial imagery and produce very accurate results in any location across the globe. We demonstrate the effectiveness of our approach outperforming the state-of-the-art in two new benchmarks that we collect. We then show how our enhanced maps are beneficial for semantic segmentation of ground images.*

1. Introduction

Over the past decades many contextual models have been developed to improve object recognition [41, 20, 24, 18, 19, 14, 15, 6, 10, 31, 39, 11, 16, 3, 13]. Particularly successful are approaches that use maps to improve localization [22], layout estimation [22] and holistic scene understanding [36]. Most self-driving cars (*e.g*., Google car, participants of the DARPA urban challenge) rely on detailed maps of the environment to facilitate navigation and perception. These maps are typically obtained via costly manual intervention, limiting the applicability of current approaches.

An alternative are online resources such as the OpenStreetMap (OSM) project [1], which contains a cartographic map of the road topology with good coverage over almost the full world, with around 33,968,739 km of road data. This is advantageous as it is freely available on the web and the quality and quantity of the annotations are growing over time, as more users contribute to the project. However, the map information is noisy and partially missing as for example most roads do not contain information about their width.

[1] www.openstreetmap.org

In this paper we proposed to exploit aerial images in order to enhance open-source maps (*e.g*., with road geometry). This is not an easy task as despite decades of research, large-scale automatic road segmentation from aerial images remains an open problem. Most approaches either do not deliver a topologically correct road network and/or rely on classifiers that have to be re-trained for each location in order to properly capture appearance variations. As a consequence they require tedious manual annotation for each region of the globe to be segmented. This annotation task takes around 8 hours per km^2, therefore, current approaches focus on a small set of locations.

In contrast, instead of framing the problem as semantic segmentation, we propose to use OpenStreetMap (OSM) to formulate the problem as inference in a Markov random field (MRF) which is directly parameterized in terms of the centerline of each OSM road segment as well as its width. This parameterization enables very efficient inference and returns the same topology as OSM. In particular, we can segment the OSM roads of the whole world in only 1 day when using a small cluster of 10 computers. Furthermore, our approach can be trained using only 1.5 km^2 of aerial imagery over Germany and is able to generalize to the entire world and produce state-of-the-art results without any further manual interaction. As we reason about the location of the centerline, we can handle and correct OSM mistakes as well as geo-localization/projection errors. This is not an easy task as illustrated in Fig. 1 due to shadows, occlusions and misalignments. Our energy encodes the appearance of roads, edge information, car detection, contextual features, relations between nearby roads as well as smoothness between the line segments. All our energy terms can be computed very efficiently via local, non-axis aligned integral images. Learning can also be done very efficiently using structured SVMs [33] taking 1 minute on a desktop

1550-5499/15 $31.00 © 2015 IEEE
DOI 10.1109/ICCV.2015.197

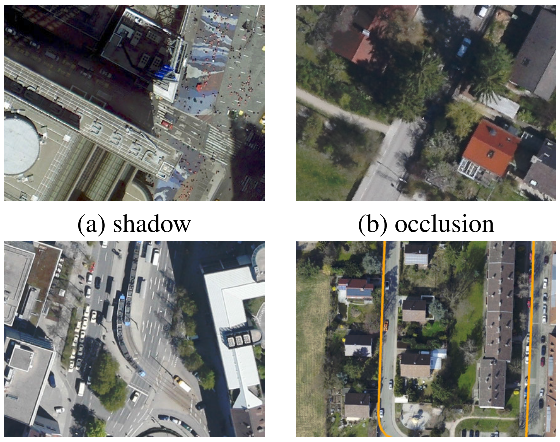

(a) shadow (b) occlusion (c) vehicles (d) misaligned centerline

Figure 1. Road segmentation is challenging due to shadows, occluding trees and vehicles which make the appearance heterogeneous as well as OSM/projection misalignment errors.

computer.

The coverage of OSM is very high in most areas, and thus by employing our parameterization we did not miss roads in our datasets. We had to exclude lower road categories as they include forest tracks and pedestrian areas, which are not sufficiently visible in the aerial images. In other regions of the globe the coverage is not as dense and our approach might miss some roads. We refer the reader to the OSM project[2] for a more detailed explanation of the coverage and its growth, and [3] for a comparison with other maps. Detecting new roads that are missing in OSM is our plan for future work.

We demonstrate the effectiveness of our approach by extracting road information from aerial images from different camera sensors taken around the whole world (*e.g.*, Toronto, Sydney, New York, Manila, Nairobi). Importantly, we only employ 1.5 km^2 imagery over Germany captured by one camera sensor for training, illustrating the ability of our approach to generalize (domain adaptation). The aerial image datasets we are aware of are not labeled with the geometric information we want to extract. They either consider the road as a single centerline or label the other surfaces instead, *e.g.*, the ISPRS [4] contains the "impervious surfaces" class but no roads. Therefore we collect two new datasets namely Bavaria and aerial KITTI, which we manually annotate and show that our approach significantly outperforms all competitors. We then demonstrate the usefulness of our road priors for the task of semantic segmentation on KITTI ground images, and show that we can provide better cartographic priors than [36]. We will release code and datasets to reproduce all results on the paper.

[2]http://wiki.openstreetmap.org/wiki/Stats
[3]http://tools.geofabrik.de/mc
[4]http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html

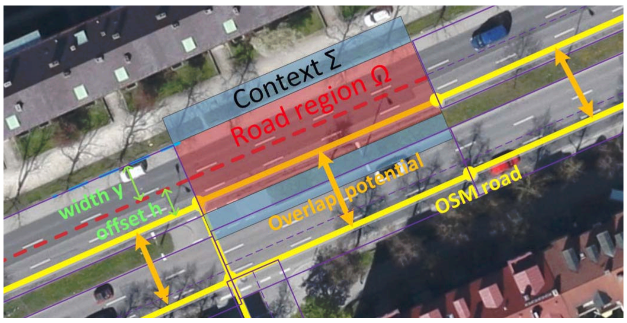


Figure 2. Illustration of the road centerline with the width parameterized by the center offset h and symmetrical width y. The direction and length of the rectangle Ω_i is defined by the p_i, p_{i-1} points given by the street database. The context is depicted as Σ.

2. Related Work

Road segmentation in aerial images has draw a lot attention for decades in the computer vision and remote sensing communities. However, it still remains an open-problem due to the difficulties in handling appearance variations and producing topologically correct segmentations. Early approaches search for objects that fulfill a pre-defined criteria. [2] defines a geometric-stochastic model and estimates the roads by tiling the input image. [32] use a Point Process to simulate and detect a network of connected line segments. We refer the reader to [25] for a detailed literature review and comparison. These approaches, however, share a common drawback: they require manual parameter tuning. Learning based methods have been proposed to be more robust to appearance variations. Mnih and Hinton [26] proposed a two stage approach, where first a neural net is used to label patches independently. Road topology is then corrected using a post-processing step. This was extended in [27] to deal with noisy training labels by employing a robust loss function. However, this method suffers from block effects due to the patch-based prediction. [37] model the road classification as a CRF, where the high-order cliques are sampled over straight segments or junctions to maintain a road-like network structure. In [28] height-field contextual information captured from dense stereo matching is used to improve segmentation. This approach is computationally very expensive and results were shown in a single location. [4] sample graph junction-points using image consistency and shape priors, resulting in long computation times (4 min/image). [34] formulate the delineation of linear loopy structures as an Integer program. However, only simple suburban scenes were tackled.

Map information has been used in both computer vision and robotics communities. Aerial image and land cover attribute maps are exploited in [21] for single image geo-localization. Kalogerakis *et al.* [17] built a human travel prior from maps to geolocalize time-stamped photographs. Brubaker *et al.* [3] use road networks for self-localization.

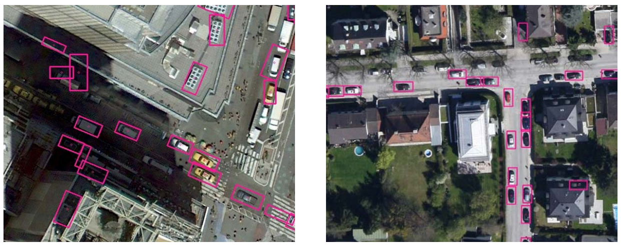


Figure 3. Output of the car detector. This task is challenging due to the small resolution of the target objects. The left image was captured from Google Earth while the right is part of the Bavaria dataset which was used for training the detector.

In [24] various maps of New York city were used to detect and localize cars from ground images. [22] use floor plans to localize and reconstruct in 3D single images in apartments. [36] use OSM to generate a geographic prior for outdoor holistic scene understanding improving performance in 3D object detection, pose estimation, semantic segmentation and depth reconstruction.

In [30] the road segmentation in aerial images is formulated as a weakly supervised classification problem, in which superpixels that overlap with road vector data are adopted as positive samples. However, inaccuracies of the road vector data are not taken into account and the solution does not preserve topology. [40] consider road segmentation as a width estimation problem. By analyzing the spatial distribution of superpixel boundaries along the direction of the road, the road width is retrieved for each line segment independently. However, their approach is not robust to shadows and occlusion.

3. Enhancing Road Maps from Aerial Images

In this section we show how to enhance world maps by parsing aerial images. In particular, we frame the problem as the one of inference in a Markov random field where the noisy cartographic map is employed to directly parameterize the problem. This parameterization is very robust and enables efficient inference.

3.1. Energy Formulation

In OSM, each road centerline is defined as a polyline chain (*i.e.*, piece-wise linear curve) but no information about the road width is typically available. Unfortunately OSM roads are not very accurate as they are either edited by volunteers without explicit quality control, or computed automatically from GPS trajectories. Furthermore, geo-localization and projection errors make the vertices of the polyline poorly aligned with the center of the road in aerial images. We refer the reader to Fig. 1 for an illustration of the difficulties of the problem. Thus we re-reason about their true location. Given a geo-localized aerial image, we model each road with a set of random variables representing for each vertex of the polyline an offset in the normal direction as well as the width of the road segment. We refer the reader to Fig. 2 for an illustration of our parameterization.

More formally, let $\mathbf{h}^j = \{h_1^j, \cdots, h_{l_j}^j\}$ be a set of random variables encoding the offsets of each vertex of the polyline that defines the j-th road, where l_j is the number of vertices for that road and $h_i^j \in [-30, 30]$ pixels. Our images have a resolution of 13 cm/pixel. Denote $\mathbf{y}^j = \{y_1^j, \cdots, y_{l_j}^j\}$ the width of each segment that compose the j-th road, with $\mathbf{y}^j \in [24, 50]$ pixels. Note that the hypothesis spaces for h and y are defined based on our empirical estimate of maximal road width and OSM projection error. Further, let $\mathbf{h} = \{\mathbf{h}^1, \cdots, \mathbf{h}^L\}$ and $\mathbf{y} = \{\mathbf{y}^1, \cdots, \mathbf{y}^L\}$ be the set of offsets and widths for all roads respectively. Denote $\mathbf{x}$ the input aerial image. We define the energy of our road segmentation as a sum of potentials encoding the image evidence, the presence of car detections, smoothness between widths and offsets of consecutive road segments and overlap constraints between nearby parallel roads

$$
\begin{aligned}
E(\mathbf{h},\mathbf{y}) = & \sum_{j=1}^{L}\sum_{i=1}^{l_j} \mathbf{w}_{road}^T \phi_{road}(h_i^j, y_i^j, \mathbf{x}) \\
& + \sum_{j=1}^{L}\sum_{i=1}^{l_j} \mathbf{w}_{ap}^T \phi_{ap}(h_i^j, y_i^j, \mathbf{x}) + \sum_{j=1}^{L}\sum_{i=1}^{l_j} \mathbf{w}_{car}^T \phi_{car}(h_i^j, y_i^j, \mathbf{x}) \\
& + \sum_{j=1}^{L}\sum_{i=1}^{l_j-1} \mathbf{w}_{sm}^T \phi_{sm}(h_i^j, y_i^j, h_{i+1}^j, y_{i+1}^j) \\
& + \sum_{i,j,k,m \in P} \phi_{ol}(h_i^j, y_i^j, h_k^m, y_k^m) \qquad (1)
\end{aligned}
$$

Note that the overlap energy does not have a weight as it is a hard constraint. We use three types of appearance features: distance to edges, homogeneity of the region as well as its context, *i.e.*, $\phi_{app} = [\phi_{edge}, \phi_{hom}, \phi_{context}]$. We now describe our potentials in more details.

Road classifier: We employ a road classifier to compute for each pixel the likelihood of being road/non-road. The potential for each segment $\phi_{road}(h_i^j, y_i^j)$ is simply the sum of the likelihoods of all pixels in the non-axis aligned rectangle Ω_i^j defined by h_i^j, y_i^j (see Fig. 2 for an example).

$$
\phi_{road}(h_i^j, y_i^j) = \sum_{p \in \Omega_i^j(h_i^j, y_i^j)} \varphi(p) \qquad (2)
$$

with $\varphi(p)$ the classifier score at pixel p. Note that this can be very efficiently computed using non-axis aligned integral images. Since we know the orientation of each segment, only a single integral image is necessary per segment. The integral image is also local to the segment, as the hypothesis space covers regions near the original OSM vertices.

Method	*Bavaria*						*Aerial KITTI*					
	IoU		F1		$\overline{\Delta h}$[m]	$\overline{\Delta y}$[m]	IoU		F1		$\overline{\Delta h}$[m]	$\overline{\Delta y}$[m]
	GT	Oracle	GT	Oracle			GT	Oracle	GT	Oracle		
Road Unary [38]	49.7	48.3	66.4	65.1	–	–	32.8	31.2	49.4	47.6	–	–
OSMxSeg	61.6	60.6	76.2	75.5	–	–	50.3	48.8	67.0	65.6	–	–
FSeg [40]	63.0	65.3	77.3	79.0	2.11	1.15	55.4	58.6	71.3	73.9	2.32	1.25
OSMFixed	64.7	66.9	78.6	80.2	1.75	1.45	51.0	53.8	67.6	70.0	2.38	1.21
Ours	**73.5**	**77.2**	**84.8**	**87.2**	**1.30**	**0.97**	**71.8**	**77.5**	**83.6**	**87.4**	**0.91**	**0.79**
Oracle	86.5	100	92.7	100	0	0	84.2	100	91.4	100	0	0

Table 1. Performance of our method vs baselines. The IoU and F1 values are in %, while $\overline{\Delta h}$, $\overline{\Delta y}$ are the mean absolute error of the offset and width measured in meters.

Method	*Bavaria*						*Aerial KITTI*					
	IoU		F1		$\overline{\Delta h}$[m]	$\overline{\Delta y}$[m]	IoU		F1		$\overline{\Delta h}$[m]	$\overline{\Delta y}$[m]
	GT	Oracle	GT	Oracle			GT	Oracle	GT	Oracle		
Road+Edge+Car	72.2	75.7	83.8	86.2	1.57	1.10	70.7	76.3	82.8	86.5	1.05	0.84
Road+Edge+Car+$\|\|$	72.8	76.4	84.2	86.7	1.39	1.03	**71.8**	**77.6**	**83.6**	**87.4**	**0.91**	0.79
Edge+Hom+Context+Car	64.8	68.4	78.6	81.2	1.58	1.26	63.6	67.2	77.8	80.3	1.61	1.36
Edge+Hom+Context+Car+$\|\|$	69.7	72.6	82.1	84.2	1.52	1.09	63.5	67.4	77.7	80.5	1.49	1.26
All	73.0	76.2	84.4	86.5	1.51	1.08	71.2	76.8	83.2	86.9	1.05	0.84
All+$\|\|$	**73.5**	**77.2**	**84.8**	**87.2**	**1.30**	**0.97**	**71.8**	77.5	**83.6**	**87.4**	**0.91**	**0.79**
	Domain shift (train on one dataset, test on the other)											
Road+Edge+Car	70.0	74.3	82.4	85.2	1.45	1.06	66.0	71.0	79.5	83.0	1.33	0.89
Road+Edge+Car+$\|\|$	70.7	75.2	82.8	85.8	1.30	0.99	66.8	72.0	80.1	83.7	1.18	0.83
Edge+Hom+Context+Car	69.1	71.5	81.7	83.4	1.73	1.11	59.3	63.5	74.4	77.6	1.63	1.17
Edge+Hom+Context+Car+$\|\|$	70.4	73.4	82.7	84.6	1.43	0.98	62.0	65.7	76.5	79.3	1.57	1.36
All	70.8	75.1	82.8	85.8	1.37	1.02	**67.7**	72.8	**80.7**	84.3	1.20	0.84
All+$\|\|$	**71.7**	**76.1**	**83.5**	**86.4**	**1.27**	**0.93**	**67.7**	**73.2**	**80.7**	**84.6**	**1.08**	**0.79**

Table 2. Performance on Bavaria and Aerial KITTI with various features configurations. The IoU and F1 values are in %, while $\overline{\Delta h}$, $\overline{\Delta y}$ are the mean absolute error of the offset and width measured in meters. The $||$ symbol denotes the overlap potential between parallel roads.

Edge: We expect the boundaries of road segments to match image appearance boundaries. Towards this goal, we compute edges using the line detector of [7], and define the potential as the distance d from each rectangle boundary pixel to the closest image edge

$$\phi_{edge}(h_i^j, y_i^j) = \sum_{p \in \partial\Omega_i^j(h_i^j, y_i^j)} \min_{e \in \mathcal{E}} d(p, e) \tag{3}$$

with $\partial\Omega$ the boundary of the rectangle Ω, p a pixel and $\mathcal{E}$ the set of all lines returned by the line detector. We adopt the distance transform of [8] to accelerate the computation.

Object detector: We train a car detector using the detector of [23]. Note that this task is extremely challenging as on average a car has only 30×12 pixels (see Fig. 3). We form a 2D feature for each car by computing $[s \cdot \sin(\Delta\alpha), s \cdot \cos(\Delta\alpha)]$, with $\Delta\alpha$ the angle between the segment and the car and s the confidence of the detector. The car potential ϕ_{car} is simply the sum of the features of all the detected cars that are inside the rectangle. Given the car features, the potentials can be computed efficiently using accumulators in a local region around each segment.

Homogeneity: An important property of roads is that they are typically free of obstacles (otherwise we could not drive on them) and therefore we expect their appearance to be homogeneous. This is violated if there are vehicles, shadows or if our aerial view of the road is obstructed by trees, bridges or tunnels. In those cases we expect the other potentials to correct the mistakes. We capture homogeneity by first transforming the image into Luv space and computing for each channel the standard deviation of the appearance inside the rectangle. This potential can be efficiently computed using two non-axis aligned integral images per channel: one computing the sum of intensities and the other the sum of square intensities. Note that this calculation was used in [35] to normalize the Haar-like features in a sub-window.

Context features: This feature encodes the fact that the road looks different than the area around it. Similar to [35], we compute the difference between the means of pixel intensities in the context and road rectangles, Σ_i^j and Ω_i^j respectively (see Fig. 2). The potential is computed by aggregating the difference across all Luv channels. Again, we use integral images for efficiency.

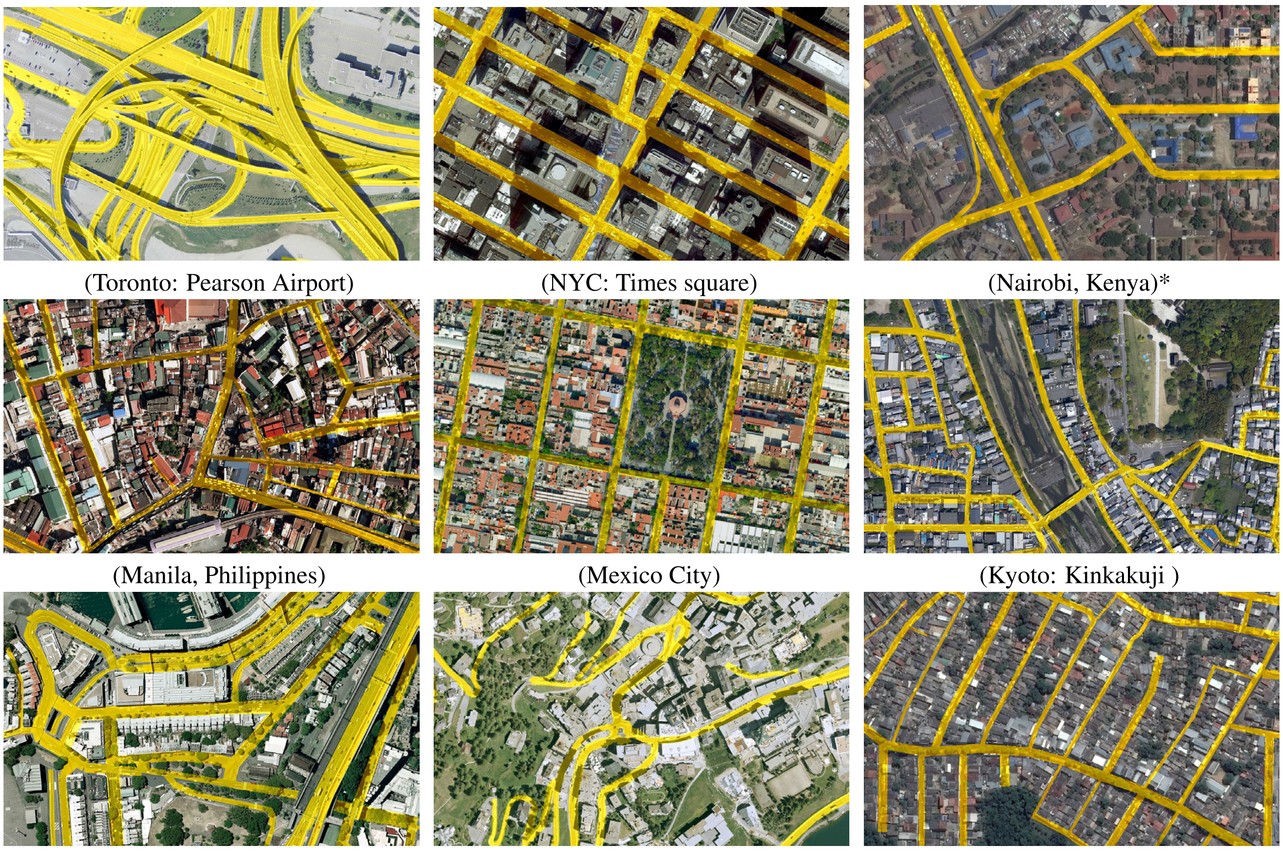


Figure 4. Segmentation results on several cities over the world using the edge, homogeneity, context, car and overlap features. Note that the MRF was trained only on $1.5\ km^2$ imagery from the Aerial KITTI dataset. * Indicates satellite image.

Smoothness: The widths and offsets along the same road tend to be similar in nearby segments. Our smoothness potentials for both h and y are defined between consecutive segments along the same road as a weighted sum of ℓ_0 and ℓ_2 norms.

Overlap: This is a hard constraint encoding the fact that two parallel roads can not overlap. We enforce this for all roads that have similar orientation (within 20 degrees) and are close enough that they could overlap.

3.2. Inference

Inference in our model can be done by computing the minimum energy configuration

$$\{\mathbf{h}^*, \mathbf{y}^*\} = \text{argmin}_{\mathbf{h},\mathbf{y}} E(\mathbf{h}, \mathbf{y}) \tag{4}$$

with $E(\mathbf{h}, \mathbf{y})$ the total energy defined in Eq. (1). Note that due to the overlap constraint, the graphical models might contain loops. As a consequence exact inference is not possible. When there is no overlap, the graphical model is composed of set of chains and dynamic programing yields the exact solution. Inspired by the stereo work of [5], we employ block coordinate descent (BCD) to perform approximated inference. Towards this goal, we define each block in BCD to form a chain since we can then solve each step to optimality. We then alternate between going over all horizontal and vertical chains to propagate the information. Note that since we solve each sub-step to optimality this procedure is guaranteed to converge. We refer the reader to Fig. 5 for an illustration, where to simplify the figure we have collapsed the width and offset variables in a single variable $g_i^j = (h_i^j, y_i^j)$. It is important to note that each of the BCD steps (*i.e.*, optimization over a subset of variables) involve conditioning, and thus the pairwise potentials between a variable in the chain and a connected variable not in the chain are folded as unaries. Prior to BCD, we initialize all variables by performing inference along each road chain and ignoring the connections between nearby parallel roads. We refer the reader to Algorithm 1 for more details about the block coordinate descent.

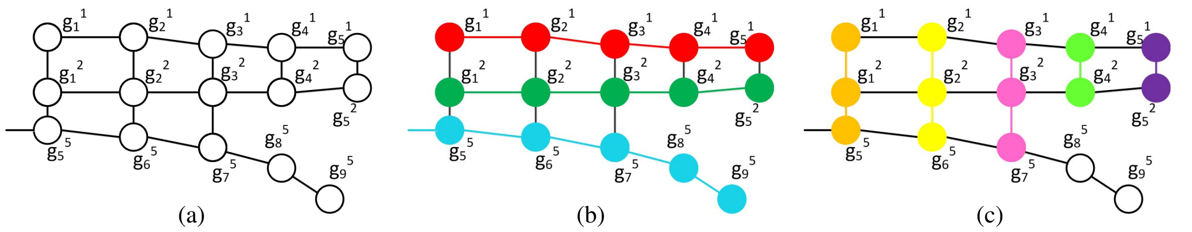


Figure 5. Illustration of our BCD inference. Note that $g_i^j = (h_i^j, y_i^j)$. (a) Graphical model consisting of 3 roads that have overlapping constraints (*i.e*., vertical edges). We alternate between performing inference (b) over each road one at a time (red, green, blue), and (c) along chains on the vertical direction encoding the horizontal constraints, also one at a time (orange, yellow, pink, green, purple). Note that these operations involve conditioning, and thus the pairwise potentials between a variable in the chain and a connected variable not in the chain are folded as unaries.

Algorithm 1 Block coordinate descent inference (BCD)

```
1: Initialize $(\mathbf{h}, \mathbf{y})$ by minimizing Eq. (1) ignoring the overlap potentials
2: repeat
3:     for all roads $R_j$ do
4:         Minimize Eq. (1) w.r.t $\mathbf{h}^j, \mathbf{y}^j$ holding the rest fixed.
5:     end for
6:     for all overlap chains $O_i$ do
7:         Minimize Eq. (1) over the variables in the overlap chain
8:     end for
9: until no energy reduction or max number iterations
```

3.3. Learning

We learn the parameters of the MRF using a structural SVM (S-SVM)[33] by minimizing

$$\min_{\mathbf{w}\in\mathbb{R}^D} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{N}\sum_{n=1}^{N}\xi_n$$
$$\text{s.t. } \delta(\mathbf{h},\mathbf{y}) \geq \Delta(\mathbf{h},\mathbf{y}) - \xi_n, \forall(\mathbf{h},\mathbf{y}) \in \mathcal{H}\times\mathcal{Y}\setminus(y_n,h_n), \forall n \tag{5}$$

with $\delta(\mathbf{h},\mathbf{y}) = E(\mathbf{h},\mathbf{y}) - E(\mathbf{h}_n,\mathbf{y}_n)$ and $\mathcal{H}\times\mathcal{Y}$ the space of all possible labelings for $(\mathbf{h},\mathbf{y})$. Note that our definition is opposite from the one in [33], as we have defined the features in terms of an energy minimization and not a score maximization. We employ the parallel cutting plane implementation of [29] to learn the parameters. We use the intersection-over-union between the configuration and the ground-truth labels as our task loss. This can be computed as a pairwise term, and thus loss augmented inference can be done efficiently.

4. Experimental Evaluation

We perform our experiments on three different datasets: Bavaria, Aerial KITTI and World which were captured with different sensors. Note that we have access to RGB images without any elevation information. We conduct road pixel-wise annotations in all Bavaria and Aerial KITTI images.

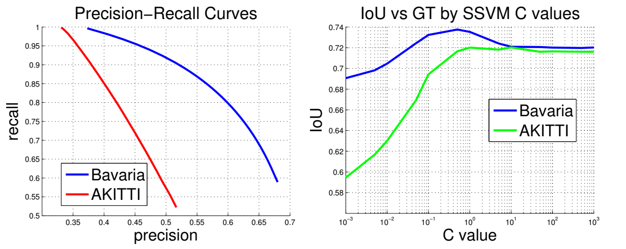


Figure 6. (Left) Precision-recall curve of the road classifier; (Right) Segmentation performance as a function of the structural SVM C parameter.

Note that the parameters not learned by S-SVM were set via four-fold cross-validation.

Bavaria: This dataset is a collection of ortho-rectified aerial images captured by a DSLR camera mounted on a plane flying around the Bavaria region in Germany [5]. It covers urban, suburban and rural areas with motorways. The resolution is 13 cm/pixel on the ground. The total area is 4.95 km^2 containing 103 km of road.

Aerial KITTI: This dataset consists of aerial images downloaded from Google Earth Pro over the city of Karlsruhe, Germany, covering the same area as the KITTI tracking benchmark [12]. The total area is 5.96 km^2 with 84 km of road. We resampled the images to be 13 cm/pixel resolution to be consistent with the Bavaria dataset.

World: This dataset consists of aerial images downloaded from Google Earth Pro of landmarks all over the world, including metropolitan areas in Toronto, New York, Sydney, Mexico City, Manaus, *etc*., as well as rural areas in St. Moritz and Kyoto. For this dataset there is no annotation.

We use four metrics to measure performance: intersection over union, F_1 score, and mean of the absolute error of h and y. We consider two different ground truth labels when evaluating the performance: our human labeled road annotations as well as the maximum achievable score with respect to our model hypothesis, refer to as *Oracle*. The

[5]We will release these images and the ground truth upon publication.

Features	Time (s) per km	
	Accumulator	Inference
Road+Edge+Car	0.07	0.031
Road+Edge+Car+||	0.069	0.092
All	0.126	0.032
All+||	0.122	0.095

Table 3. Running time for feature accumulator calculation and inference under various configurations. In sec per km of road.

later can be computed by performing our MRF inference, by replacing our unary potentials with ground truth segmentations. For all quantitative experiments we perform four-fold cross-validation.

To compute our road classifier, we first convert the image to opponent Gaussian color space and extract a dense filter response map, with a filterbank composed of 17 edge-like filters [38]. We oversegment the image using SLIC [1] and calculate the mean and std of the filter responses in each superpixel. We then train a random forest classifier [7] with this $34D$ input feature. Note that this road classifier was used in [37] as unary potential. Fig. 6 shows the Precision-Recall curve of the road classifier on Bavaria and Aerial KITTI.

Comparison to baselines: We compare our approach to four baselines: The first one is the road classifier unary potential of [37], denoted *Road Unary*. The second baseline, denoted as *OSMxSeg*, is computed by segmenting the image into superpixels using [9] and labeling each super pixel as road if it is crossed by a road segment in OSM. We also reproduce the state-of-the-art method of [40], denoted as *FSeg*, which also uses the OSM road data. To illustrate the effectiveness of our cartographic prior, the last baseline, denoted *OSMFixed*, projects OSM into the image and utilizes an empirical estimate of the road width. As shown in Table 1 our approach significantly outperforms all baselines in both Bavaria and aerial KITTI datasets. (see qualitative results in Fig. 8). Fig. 10 shows a comparison to [40].

Importance of the features: Table 2 depicts inference results for different combinations of features. Note that every feature contributes, and good performance can be achieved without using a road classifier. As a consequence, we do not need new training data for each different location in the world as the other features are very robust to appearance changes.

Segmenting the world: Fig. 4 shows qualitative results from the *World* dataset with our model trained only on *AKITTI*. Our model works very well under many complex scenarios even with significant appearance changes, illustrating the generalization capabilities of our approach. Note that no re-training is necessary as we do not use the road classifier in our potentials.

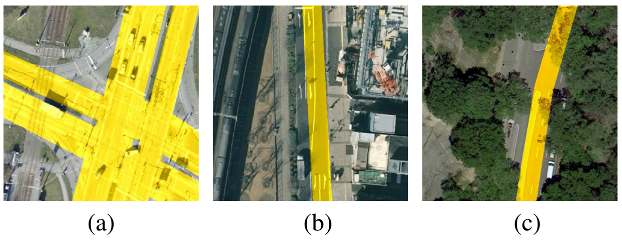


Figure 7. Failure modes: (a) Missing turn lane intersection. (b) The extracted road is too narrow. (c) Road covered by trees.

	Sky	Build	Road	Sidewalk	Vege	Car
[36]	32.41	59.25	63.01	36.41	7.36	35.65
Ours	32.41	59.10	**78.71**	**41.96**	7.36	35.65

Table 4. Our method improves the geographic priors of [36]. All values are IoU in %.

Domain Adaptation: We next show our method's domain adaptation ability. Towards this goal, we trained one model on Aerial KITTI and evaluate its performance on Bavaria, and vise versa. As shown in Table 2 our algorithm outperforms all baselines despite the fact that it is trained with different imagery. Furthermore, performance drops less than 5% IoU when compared when we train on the same dataset we test on.

Processing time: We implemented our method in C++ without multi-threading and test it on a laptop with an Intel Core i7-4600M processor. As shown in Table 3, our approach takes less than 0.13 s for computing all feature accumulators per km of road and less than 0.1 s per km for inference. The feature computation (road classifier, edge, car detector) relies on external code which takes around 0.1s per km of road. According to this performance, we estimate our algorithm could approximately segment all the OSM roads in the world in 1 day using a small cluster of 10 machines. We use the parallel cutting-plane structured SVM of [29] to learn the parameters of the model. This takes only 1 minute on a desktop computer.

Ground-level Scene Understanding: In this experiment we show that our enhanced maps can be used to improve semantic segmentation of ground images from KITTI. Towards this goal, we replace the road prior used in [36] by the estimations of our method. This improves the geographic unary prior for the road class by 15%, see the Table 4. Qualitative results are shown in Fig. 9.

Failure modes and limitations: Fig. 7 depicts failure modes. (a) At intersections the OSM might not include the turn lanes and our model can not recover from this. (b) In some cases our features/weights are not good for the scene. This is more likely to happen in the strong generalization case. (c) The road can be (partly) covered, and we only extract the visible part of the road. Additional challenges are

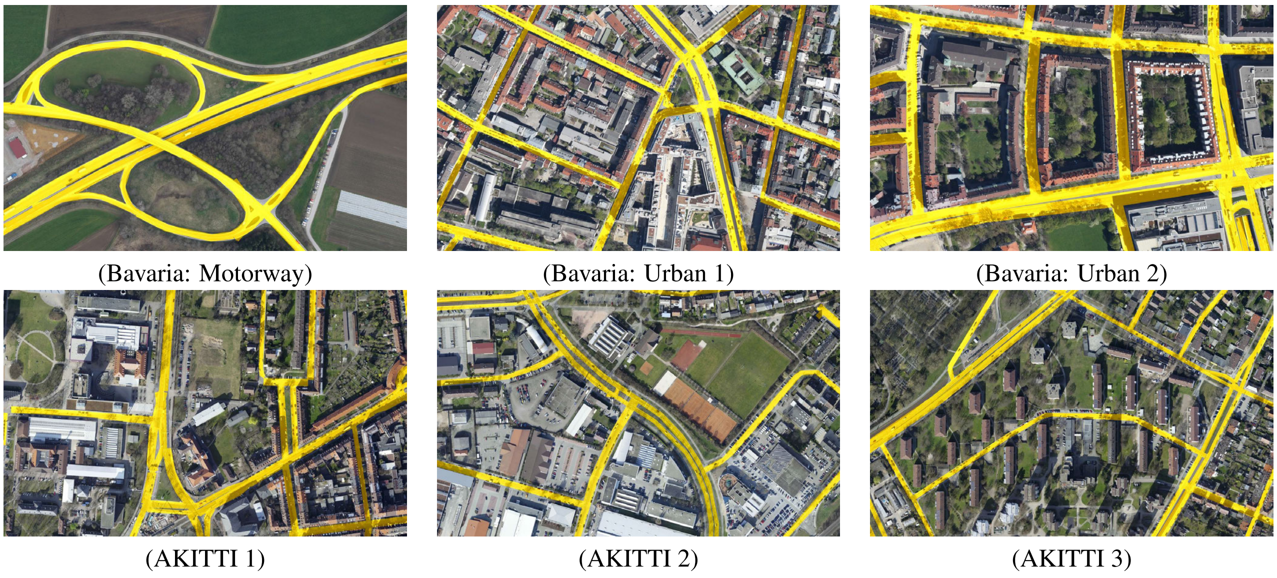


Figure 8. Results on Bavaria and Aerial KITTI.

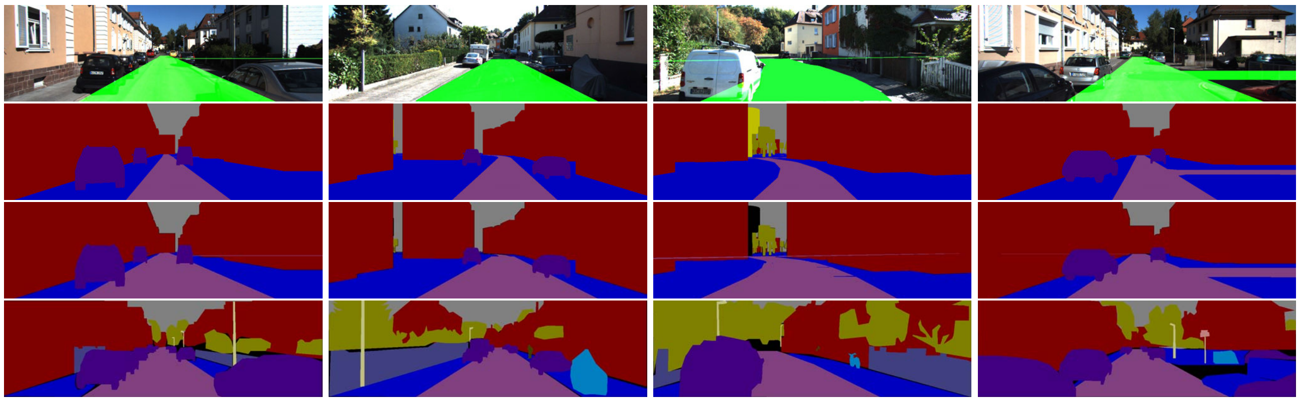


Figure 9. (Top) Our road extracted from aerial images (green) projected into Kitti ground images. (2nd row): Geographical unary of [36]. (3rd row): Geographical unary with our road estimate. (Bottom) Ground truth. Road (pink), sidewalk (blue), building (red), car (purple).

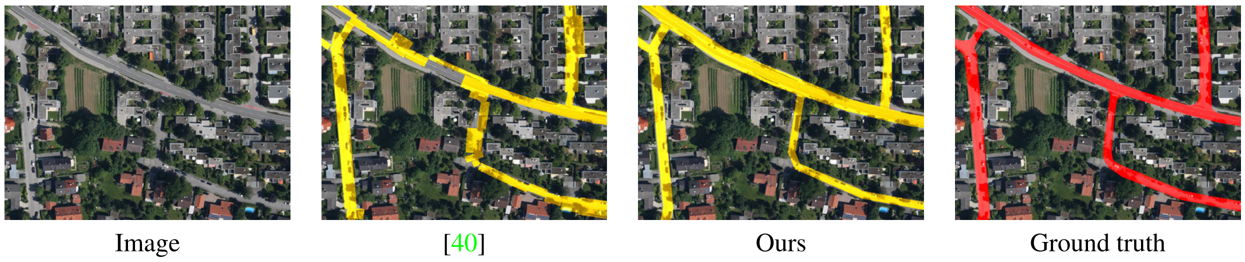


Figure 10. Comparison to [40]: Our approach works significantly better than the baseline.

posed by historical city centers were the roads might not be visible as well as developing countries, where only satellite images with much lower resolution than aerial images might be available.

5. Conclusion

We have presented an approach to enhance world maps by parsing aerial images. By parameterizing the problem in terms of OSM road segment centerlines and widths, we were able to extract road properties very efficiently. In particular, we can process the whole world in a single day using a small cluster. Importantly, our approach can be trained with as little as $1.5\ km^2$ aerial imagery from a single area and it is able to generalize to the full world. We have demonstrated the effectiveness of our approach in three different datasets captured by different sensors in different regions of the world.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *PAMI*, 2012. 7

[2] M. Barzohar and D. Cooper. Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation. *PAMI*, 1996. 2

[3] M. A. Brubaker, A. Geiger, and R. Urtasun. Lost! leveraging the crowd for probabilistic visual self-localization. In *CVPR*, 2013. 1, 2

[4] D. Chai, W. Forstner, and F. Lafarge. Recovering line-networks in images by junction-point processes. In *CVPR*, 2013. 2

[5] Q. Chen and V. Koltun. Fast mrf optimization with application to depth reconstruction. In *CVPR*, 2014. 5

[6] S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert. An empirical study of context in object detection. In *CVPR*, 2009. 1

[7] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013. 4, 7

[8] P. Felzenszwalb and D. Huttenlocher. Distance transforms of sampled functions. Technical report, Cornell University, 2004. 4

[9] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 2004. 7

[10] S. Fidler, S. Dickinson, and R. Urtasun. 3d object detection and viewpoint estimation with a deformable 3d cuboid model. In *NIPS*, 2012. 1

[11] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun. 3d traffic scene understanding from movable platforms. *PAMI*, 2014. 1

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *IJRR*, 2013. 6

[13] J. Hays and A. A. Efros. Im2gps: estimating geographic information from a single image. In *CVPR*, 2008. 1

[14] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009. 1

[15] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *ICCV*, 2005. 1

[16] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. *IJCV*, 2008. 1

[17] E. Kalogerakis, O. Vesselova, J. Hays, A. A. Efros, and A. Hertzmann. Image sequence geolocation with human travel priors. In *ICCV*, 2009. 2

[18] L. Ladickỳ, C. Russell, P. Kohli, and P. H. Torr. Graph cut based inference with co-occurrence statistics. In *ECCV*. 2010. 1

[19] L. Ladickỳ, J. Shi, and M. Pollefeys. Pulling things out of perspective. In *CVPR*, 2014. 1

[20] L. Ladickỳ, P. Sturgess, K. Alahari, C. Russell, and P. H. Torr. What, where and how many? combining object detectors and crfs. In *ECCV*. 2010. 1

[21] T.-Y. Lin, S. Belongie, and J. Hays. Cross-view image geolocalization. In *CVPR*, 2013. 2

[22] C. Liu, A. Schwing, R. Urtasun, and S. Filder. Rent3d: Floor-plan priors for monocular layout estimation. *CVPR*, 2015. 1, 3

[23] K. Liu and G. Mattyus. Fast multiclass vehicle detection on aerial images. *GRSL*, 2015. 4

[24] K. Matzen and N. Snavely. Nyc3dcars: A dataset of 3d vehicles in geographic context. In *ICCV*, 2013. 1, 3

[25] H. Mayer, S. Hinz, U. Bacher, and E. Baltsavias. A test of automatic road extraction approaches. In *ISPRS*, 2006. 2

[26] V. Mnih and G. E. Hinton. Learning to detect roads in high-resolution aerial images. In *ECCV*, 2010. 2

[27] V. Mnih and G. E. Hinton. Learning to label aerial images from noisy data. In *ICML*, 2012. 2

[28] J. A. Montoya-Zegarra, J. D. Wegner, L. Ladicky, and K. Schindler. Mind the gap: Modeling local and global context in (road) networks. In *GCPR*, 2014. 2

[29] A. G. Schwing, S. Fidler, M. Pollefeys, and R. Urtasun. Box In the Box: Joint 3D Layout and Object Reasoning from Single Images. In *ICCV*, 2013. 6, 7

[30] Y.-W. Seo, C. Urmson, and D. Wettergreen. Exploiting publicly available cartographic resources for aerial image analysis. In *SIGSPATIAL*, 2012. 3

[31] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*. 2012. 1

[32] R. Stoica, X. Descombes, and J. Zerubia. A gibbs point process for road extraction from remotely sensed images. *IJCV*, 2004. 2

[33] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 2005. 1, 6

[34] E. Turetken, F. Benmansour, B. Andres, H. Pfister, and P. Fua. Reconstructing loopy curvilinear structures using integer programming. In *CVPR*, 2013. 2

[35] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 2004. 4

[36] S. Wang, S. Fidler, and R. Urtasun. Holistic 3d scene understanding from a single geo-tagged image. In *CVPR*, 2015. 1, 2, 3, 7, 8

[37] J. D. Wegner, J. A. Montoya-Zegarra, and K. Schindler. A higher-order crf model for road network extraction. In *CVPR*, 2013. 2, 7

[38] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, 2005. 4, 7

[39] J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *CVPR*, 2012. 1

[40] J. Yuan and A. Cheriyadat. Road segmentation in aerial images by exploiting road vector data. In *COM.geo*, 2013. 3, 4, 7, 8

[41] M. Z. Zia, M. Stark, K. Schindler, and R. Vision. Are cars just 3d boxes?–jointly estimating the 3d shape of multiple objects. In *CVPR*, 2014. 1

D Máttyus, G., Wang, S., Fidler S. and Urtasun R., 2016. HD Maps: Fine-grained Road Segmentation by Parsing Ground and Aerial Images, Conference on Computer Vision and Pattern Recognition (CVPR) 2016.

This is the submitted camera ready version of the paper.

HD Maps: Fine-grained Road Segmentation by Parsing Ground and Aerial Images

Gellért Máttyus
Remote Sensing Technology Institute
German Aerospace Center
gellert.mattyus@dlr.de

Shenlong Wang, Sanja Fidler, Raquel Urtasun
Department of Computer Science
University of Toronto
{slwang, fidler, urtasun}@cs.toronto.edu

Abstract

In this paper we present an approach to enhance existing maps with fine grained segmentation categories such as parking spots and sidewalk, as well as the number and location of road lanes. Towards this goal, we propose an efficient approach that is able to estimate these fine grained categories by doing joint inference over both, monocular aerial imagery, as well as ground images taken from a stereo camera pair mounted on top of a car. Important to this is reasoning about the alignment between the two types of imagery, as even when the measurements are taken with sophisticated GPS+IMU systems, this alignment is not sufficiently accurate. We demonstrate the effectiveness of our approach on a new dataset which enhances KITTI [8] with aerial images taken with a camera mounted on an airplane and flying around the city of Karlsruhe, Germany.

1. Introduction

We are in an exciting time for computer vision, and more broadly AI, as the development of fully autonomous systems such as self-driving cars seems possible in the near future. These systems have to robustly estimate the scene in 3D, its semantics as well as be able to self-localize at all times. Key to the success of these tasks is the use of maps containing detailed information such as road location, number of lanes, speed limit, traffic signs, parking spots, traffic rules at intersections, etc.

Current maps, however, have been created with the use of semi-automatic systems that employ many man-hours of laborious and tedious labeling. An alternative to this costly labeling is to employ existing maps and correct/enhance them based on ground imagery or LIDAR point clouds, captured, for example, by a Velodyne/cameras mounted on top of a car. Systems like TESLA auto-pilot [1] are currently using their deployed fleet of cars, which are equipped with cameras, to perform such corrections. However, it is difficult to create full coverage of the world as we will need access to imagery/LIDAR from millions of cars in order to reliably enhance maps at a world-scale.

Alternatively, aerial images provide us with full coverage of a significant portion of the world, but at a much lower resolution than ground images. This makes semantic segmentation from aerial images a very difficult task. In this paper, we propose to use both aerial and ground images to jointly infer fine grained segmentation of roads. Towards this goal, we take advantage of the OpenStreetMap (OSM) project, which provides us with freely available maps of the road topology in the form of piece-wise linear road segments. We formulate the problem as energy minimization, inferring the number and location of the lanes for each road segment, parking spots, sidewalks and background, as well as the alignment between the ground and aerial images. We employ deep learning to estimate semantics from both aerial and ground images, and define a set of potentials exploiting these semantic cues, as well as road constraints, relationships between parallel roads, and the smoothness of both the estimations along the road as well as the alignment between consecutive ground frames.

We demonstrate the effectiveness of our approach in a new dataset which covers a wide area of the city of Karlsruhe in Germany, both from the ground and from the air. We provide pixel-level annotations for the aerial images in terms of fine-grained road categories. We call our dataset *Air-Ground-KITTI*. We show that our approach is able to estimate these categories reliably, while significantly reducing the alignment error between the ground and aerial images when compared to a sophisticated GPS+IMU system.

2. Related work

For several decades, researchers from various communities (e.g., vision, remote sensing) have been working on automatic extraction of semantic information from aerial images. In the following, we summarize the approaches most relevant to our work.

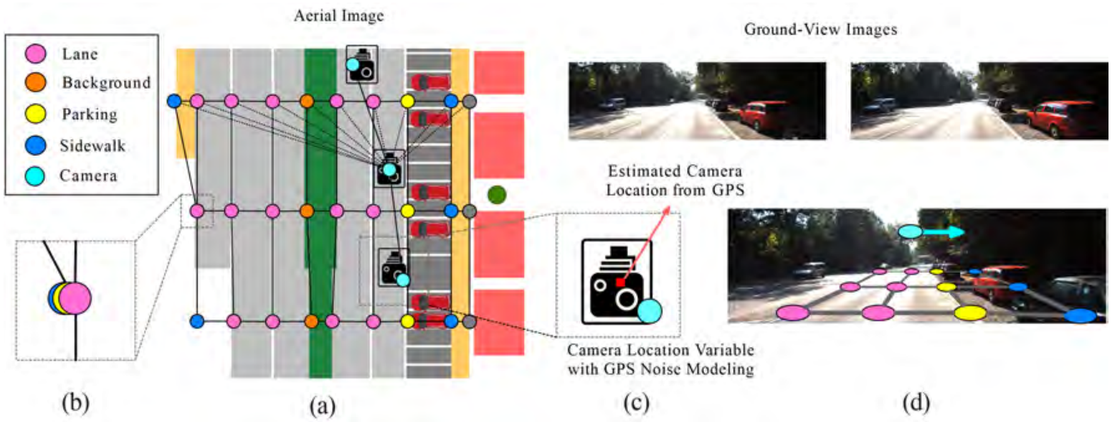

Aerial Image
Ground-View Images
Lane
Background
Parking
Sidewalk
Camera
Estimated Camera Location from GPS
Camera Location Variable with GPS Noise Modeling
(b)
(a)
(c)
(d)

Figure 1. **Illustration of our model:** (a) Parameterization of our approach. Our random variables are the absolute location of the different region boundaries (e.g., sidewalk) as well as the alignment between air and ground. (b) Our formulation allows a random variable to take the same state as the previous node, collapsing a region to have 0 width. (c). For each ground-view image, a random variable models the alignment noise. (d). Projection of our parameterization on the ground-view.

Aerial image parsing: Early approaches employed probabilistic models that aimed to produce topologically connected roads. [2] defined a probabilistic model that tiled the image into patches, performed road inference inside each patch via dynamic programming, and then "stitched" together high-confidence patches to ensure road connectivity. Recent work exploits learned classifiers to perform semantic segmentation. [15, 16] trained a neural net to classify pixels in local patches as road. They employ a post-processing step to ensure a consistent road topology across the patches, which is, however, prone to block-effects. [26] segments the road by defining an MRF on superpixels. High-order cliques are sampled over straight segments or junctions to encourage a road-like network structure. Due to complexity of high order terms a sampling scheme is used to concentrate on more important cliques. [4] samples graph junction-points using image consistency and shape priors. A full review of this large field is out of scope of this paper, and we refer the reader to [14] for a detailed review.

Aerial parsing with maps: While proven useful in many computer vision and robotics applications [9, 13, 3, 25], few works employ map information for parsing aerial images. [20] uses a screenshot of the vector map as a weak source of ground-truth for training a road classifier. [27] exploit road center-lines from OSM maps as a ground-truth road location and performs road segmentation by estimating the width of the road. This is done by finding boundaries of superpixels along the direction of the road, and ignoring dependencies across different line (road) segments. However, the alignment between OSM and aerial images is far from perfect. To solve this problem, [12] proposed a MRF which reasons about re-positioning the road centerline and estimating the width of the road. Smoothness is incorporated between consecutive line segments by encouraging their widths to be similar. In our work we go beyond this approach by introducing a formulation that reasons about more fine-grained road semantics such as lanes, sidewalks and parking spots, and exploits simultaneously aerial images as well as ground imagery to infer this information.

Fine-grained road parsing: Very few works exist that extract detailed segmentation. [17] propose a hierarchical probabilistic grammar to parse smaller-scale aerial regions into roads, buildings, vehicles and parking lots. Classifiers are first employed to generate object/building/vegetation proposals while the grammar imposes semantic and geometric constraints in order to derive the final parse. Learning and inference are both hard in grammars, and computationally expensive sampling techniques typically need to be employed. In our work, we are aiming at a detailed parsing of the roads into sub-categories. Unlike [17], we exploit OSM information in order to derive an efficient formulation.

The work most related to ours is [21] which exploits the map as a screenshot of the road vector map to perform road and lane estimation. The authors take a pipeline approach, where, in the first step, road lane hypotheses are generated based on the output of the road classifier and detected lane markings. In the second step, the authors provide heuristics to "track" the lane hypotheses and connect them into a single lane labeling.

Aerial-to-ground reasoning: Recent work aims to exploit both aerial and the ground-view, mainly for the problem of geo-localization. In [11], a deep neural network is used to match ground images with aerial images in oblique views. The matches come from facade to facade matching and therefore can not be extended to orthographic aerial images. In [22], 3D reconstructions from the ground images are matched to oblique views of aerial images. [10] learn cross-view matching between ground images, aerial orthographic photos and land cover attributes. This extends the image geolocalization to areas not covered by ground images. Forster *et al.* [7] match the computed 3D maps of

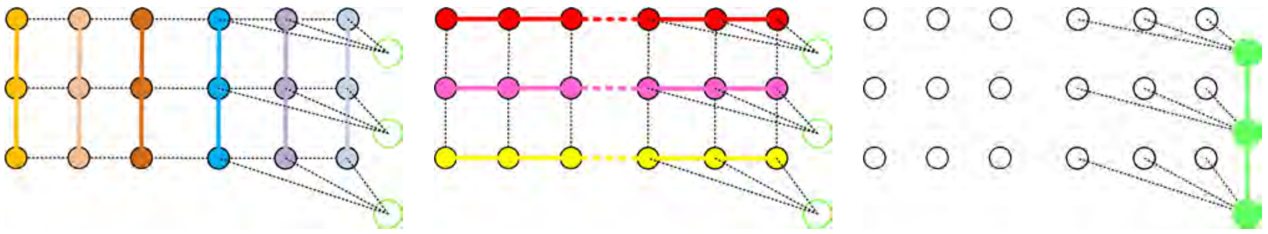

(a) Along Road (b) Perpendicular to Road (c) Along ground image sequence

Figure 2. **BCD:** The graph shows a simplified network with two parallel roads (each with 3 random variables) and one ground image per segment connected to the right road. BCD alternates between three types of updates. (a) Along the road updates: we optimize over each chain with the same color (while holding all other variables fix). The pairwise terms fold to unaries (see dashed black lines). (b) Perpendicular to the road updates: we do inference for the nodes with the same color (holding the rest fix). (c) Along the ground alignments: We minimize only the t variables which are depicted in green. The y variables are fixed and are depicted in black.

MAVs and ground robots for localization and map augmentation. This method relies on matching 3D information and therefore needs multiview images both from above and on ground. In our work, we exploit the maps as well as ground and aerial imagery to perform fine-grained road parsing. We are not aware of prior work that tackles this problem.

3. Fine-grained Semantic Parsing of Roads

We now describe our model that infers fine-grained semantic categories of roads from aerial and ground images. In particular, we are interested in estimating *sidewalks*, *parking*, *road lanes* as well as *background* (e.g., vegetation, buildings). Towards this goal we exploit freely available cartographic maps (we use OSM), that provide us with the topology of the road network in the area of interest. Our approach takes as input an aerial image $\mathbf{x}_A$, a road map $\mathbf{x}_M$ and a set of ground stereo images $\mathbf{x}_G$, which are taken by a calibrated stereo pair mounted on top of a car. The map $\mathbf{x}_M$ is composed of a set of roads, where each road is defined as a piece-wise linear curve representing its centerline.

3.1. Model Formulation

We formulate the problem as the one of inference in a Markov random field (MRF), which exploits deep features encoding appearance in both aerial and ground images, edge information, smoothness in the direction of the road as well as restrictions between parallel roads to avoid double counting the evidence. Our model encodes each street segment in the aerial image with 15 random variables encoding all possible combinations of *background* (B), *sidewalk* (S), *road lanes* (L) and *parking* (P). In particular,

$$\begin{aligned}\mathbf{y} &= (y_1, \cdots, y_{15}) \\ &= (B_1, S_1, B_2, S_2, P_1, L, P_2, S_3, B_3, S_4, B_4)\end{aligned}$$

with B_1, B_4 the rightmost (leftmost) border of the background. We model roads with up to 6 lanes, i.e., $L = (L_1, L_2, L_3, L_4, L_5, L_6)$. We allow all variables (but L_6) to take the state of the previous random variable in the sequence (i.e., $y_i = y_{i-1}$), encoding the fact that some of these regions might be absent, e.g., there is no parking or sidewalk. This is not the case for L_6 forcing the fact that at least one lane should be present. We define the states of each random variable to be $[-15, 15]$m from the projection of the OSM centerline in the aerial image (Fig. 1). This discretization represents pixel increments. Note that while there are 15 random variables, $\mathbf{y}$ defines 16 different regions as B_1 and B_4 are not limited on the left (right). Each region width is simply defined by $w_i = y_i - y_{i-1}$, while the width of B_1 is defined as $w_1 = -15m + y_1$, and the width of B_4 as $w_{16} = 15m - y_{16}$, since $-15m$ and $15m$ are the beginning and end of the state space. Note that the combination (B, S, B, S) is necessary (both on the left and right), as there are many bike lines in Germany (where our imagery is captured), and it is not possible to distinguish them from the sidewalk. Fig. 1 illustrates the model.

Each of our ground images comes with a rough alignment with the aerial image as we have access to a GPS+IMU and the cameras are registered w.r.t these sensors. This alignment is, however, noisy with 1.67m error on average. Thus, our model reasons about the alignment when scoring the ground images. Towards this goal, we define $\mathbf{t} = (t_1, \cdots, t_n)$ to be a set of random variables (one per ground image) representing the displacement in the direction perpendicular to the OSM road segment. We define the state space of each misalignment to be $t_i \in (-4m, 4m)$. This is discretized to represent pixel increments.

We define the energy of the MRF as to encode the information contained in the ground and aerial images as well as smoothness terms and constraints on the possible solutions:

$$\begin{aligned}E(\mathbf{y}, \mathbf{t}, \mathbf{x}_A, \mathbf{x}_M, \mathbf{x}_G) = E_{\text{air}}(\mathbf{y}, \mathbf{x}_A) + E_{\text{ground}}(\mathbf{y}, \mathbf{t}, \mathbf{x}_G) \\ + E_{\text{smooth}}(\mathbf{y}, \mathbf{t}, \mathbf{x}_M) + E_{\text{const}}(\mathbf{y})\end{aligned} \quad (1)$$

We now define the potentials we employ in more detail.

Aerial semantics: We take advantage of deep learning in order to estimate semantic information from aerial images. In particular, we create pixel-wise estimates of 5 semantic categories: *road*, *sidewalk*, *background*, *building* and *parking*. We exploit the CNN for segmentation [23, 19] trained

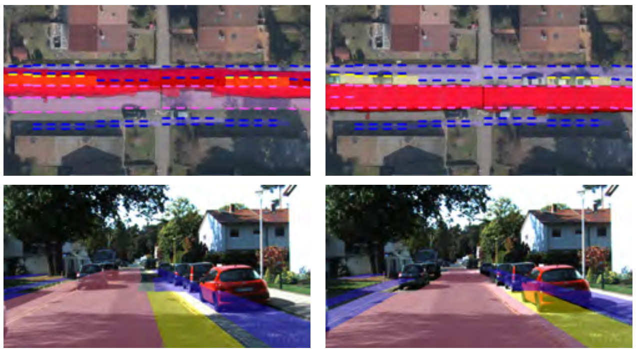


Figure 3. **Effect of reasoning about alignment: (left)** alignment given by GPS+IMU, **(right)** alignment inferred by our model. **(top)** Ground road classifier projected into the aerial image (shown in red). **(bottom)** Our semantic classes projected on the ground image. Our joint reasoning significantly improves alignment.

on ILSVRC-2014, which we fine-tune for a 5-label classification task: road, parking spot, sidewalk, building and background. To train the network we created training examples by extracting patches centered on the projection of the OSM road segments. If the road segment is too long (i.e., long straight road) we create an example every 20m. We further perform data augmentation by applying small rotations, shifts and flips to the training examples. The output of the soft-max is a downsampled segmentation. To create our features, we upsample the softmax output using linear interpolation as in [5]. To save computation, we only apply the network in the region of interest (regions of the image that are close to OSM roads). The aerial semantic potential then encodes the fact that our final segmentation should agree with the semantics estimated by the deep net. Towards this goal, we define 5 features for each of our 16 regions, one per label of the deep net. Each feature simply aggregates the output of the softmax in that region. Recall that each region is defined by two consecutive random variables, e.g. the first sidewalk is defined by y_1, y_2, that is B_1, S_1. We refer the reader to Fig. 1 for an illustration. While this potential seems pairwise in nature, we can further decompose it into unary potentials via accumulators $\mathcal{A}$ perpendicular to the road direction. These are simply generalizations of integral images from axis aligned accumulators to accumulators over arbitrary directions. We thus define

$$\phi_{cl}(y_i^j, y_{i-1}^j) = \sum_{p \in \Omega_i^j(y_i^j, y_{i+1}^j)} \varphi(p) = \mathcal{A}(y_{i+1}^j) - \mathcal{A}(y_i^j)$$

with y_i^j the i-th variable of the j-th road segment, and $\varphi(p)$ the softmax output interpolated at pixel p. To compute this features, we only need 5 accumulators per road segment, one for each semantic class that the deep net predicts.

Aerial edges: This potential encodes the fact that the location of the boundaries between regions should be close

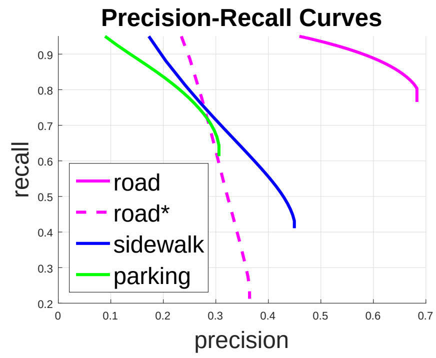


Figure 4. Precision-Recall curves for our deep classifier and the road classifier of [12] marked with * and in dashed.

to image edges. We thus apply the edge detector of [6] to detect edges in our aerial images. We then define the potential to be the sum of the edges on the boundary between consecutive regions. To make it more robust we thicken the boundary to be of size 3 pixels.

Along the road smoothness: We encode smoothness along the road by encouraging consecutive road segments to be similar. In particular, we use the ℓ_1 distance between consecutive road estimations in the direction of the road, i.e.

$$\phi_{\text{sm}}(y_i^j, y_i^{j+1}) = |y_i^j - y_i^{j+1}|$$

Parallel roads: The regions of close by parallel roads can overlap. To avoid double counting the evidence, we incorporate an additional constraint that forces S_1 of the second road to be bigger or equal to B_4 of the first road or vice versa. We refer the reader to Fig. 1 for an illustration.

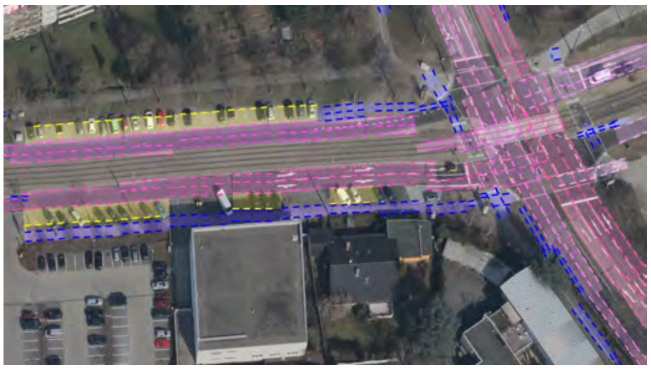
Road collapse constraints: We force each variable y_i to have a state higher or equal than the previous variable, so that the order is preserved. Note that equal means that a road can collapse (i.e., does not exist)

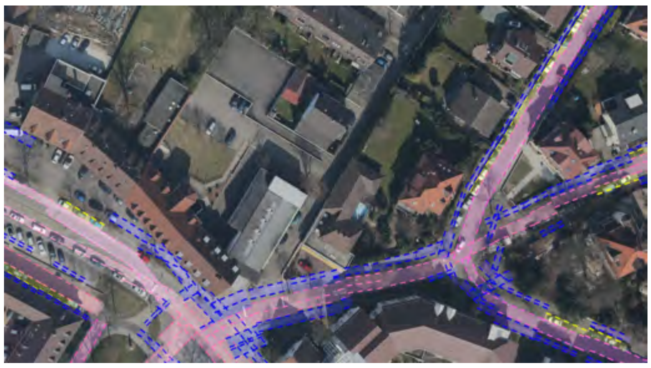
$$\phi_{coll}(y_i, y_{i+1}) = \begin{cases} \infty & \text{if } y_{i+1} < y_i \\ 0 & \text{otherwise} \end{cases}$$

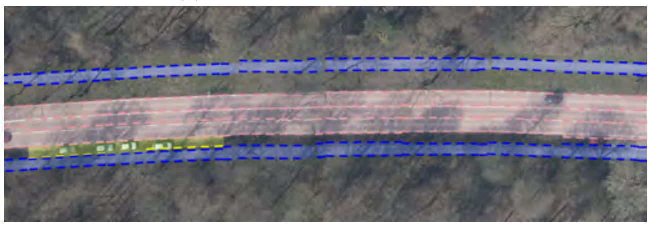
The only exception is L_6, which we force to have non-zero width as otherwise we could have a road segment without road. Thus

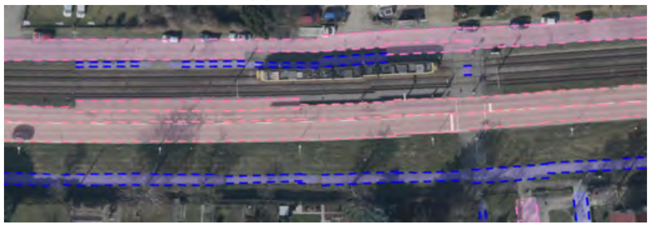
$$\phi_{ex}(L_5, L_6) = \begin{cases} \infty & \text{if } L_6 \leq L_5 \\ 0 & \text{otherwise} \end{cases}$$

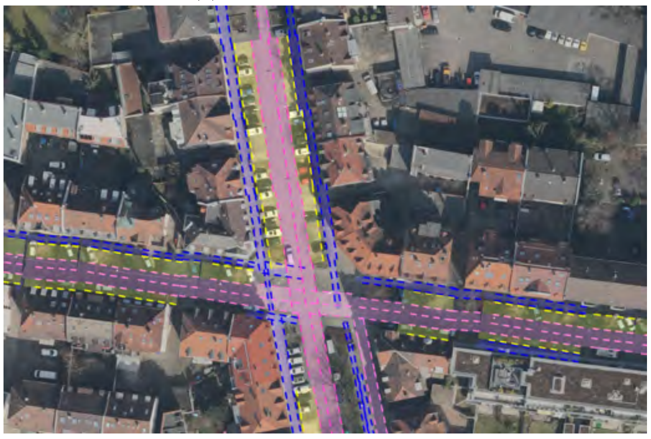
Lane size constraint: This constraint forces each region, if present, (i.e., if it is not taking state 0) to have a minimal and maximal size. In particular, we use (1m-3m) for sidewalk, (1.8m-4.5m) for parking and (2.3m-4.6m) for each road lane. Note that width 0 is allowed so that regions can disappear if they are not present in the road segment (e.g.,


(a) Intersection with tram line.


(b) Small town.


(c) A road with three lanes.


(d) Two roads with tram stop in between.


(e) Dense urban area.

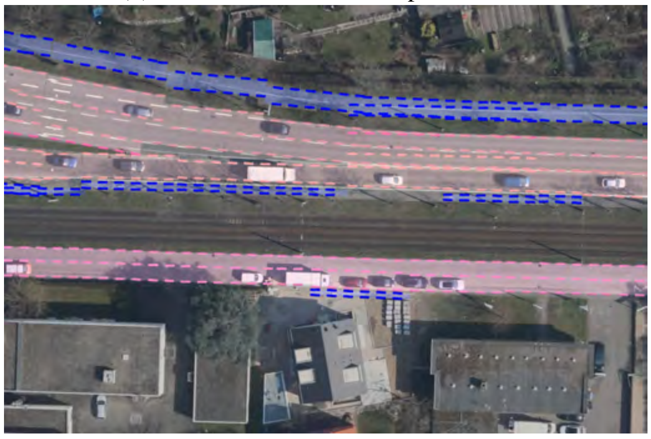

(f) Splitting road plus a bike lane along the street.

Figure 5. Visualization of our semantic road parsing results using only aerial images. The road lanes are shown with shades of pink, the sidewalk with blue and the parking spots with yellow.

we only have two lanes, there is no sidewalk on the highway). The intervals for the lanes are estimated based on the standards of German roads, while the sidewalk and parking intervals are computed based on empirical estimates.

Centerline prior: As our images are well registered with OSM, we include a prior that the centerline of our model should be close to the centerline of OSM. In particular,

$$\phi_{cen}(L_3) = \begin{cases} ||L_3 - l||_2 & \text{if } -7.5 \leq L_3 \leq 7.5 \\ \infty & \text{otherwise} \end{cases}$$

with l the location of the centerline.

Ground semantics: We take advantage of deep learning in order to estimate semantic information from ground images. We exploit the VGG [23] implementation of [19] trained on PASCAL VOC, which we fine-tuned to predict the same 5 classes as the aerial semantics (*road*, *parking*, *sidewalk*, *building* and *background*). We estimate the ground plane from the stereo image and project pixels belonging to this plane to the aerial image via a homography. We then define our ground semantic potential to encourage the segmentation to agree with the aligned ground image segmentation projected to the aerial image. Towards this goal, we define 5 features for each of our road regions, each counting the amount of softmax output for the given class:

$$\phi_{ground}(t_k, y_i^j, y_{i-1}^j) = \mathcal{G}(t_k, y_{i+1}^j) - \mathcal{G}(t_k, y_i^j)$$

Note that via the integral accumulator the 3-way potential decomposes into pairwise terms $\mathcal{G}(t, y)$. In this case we

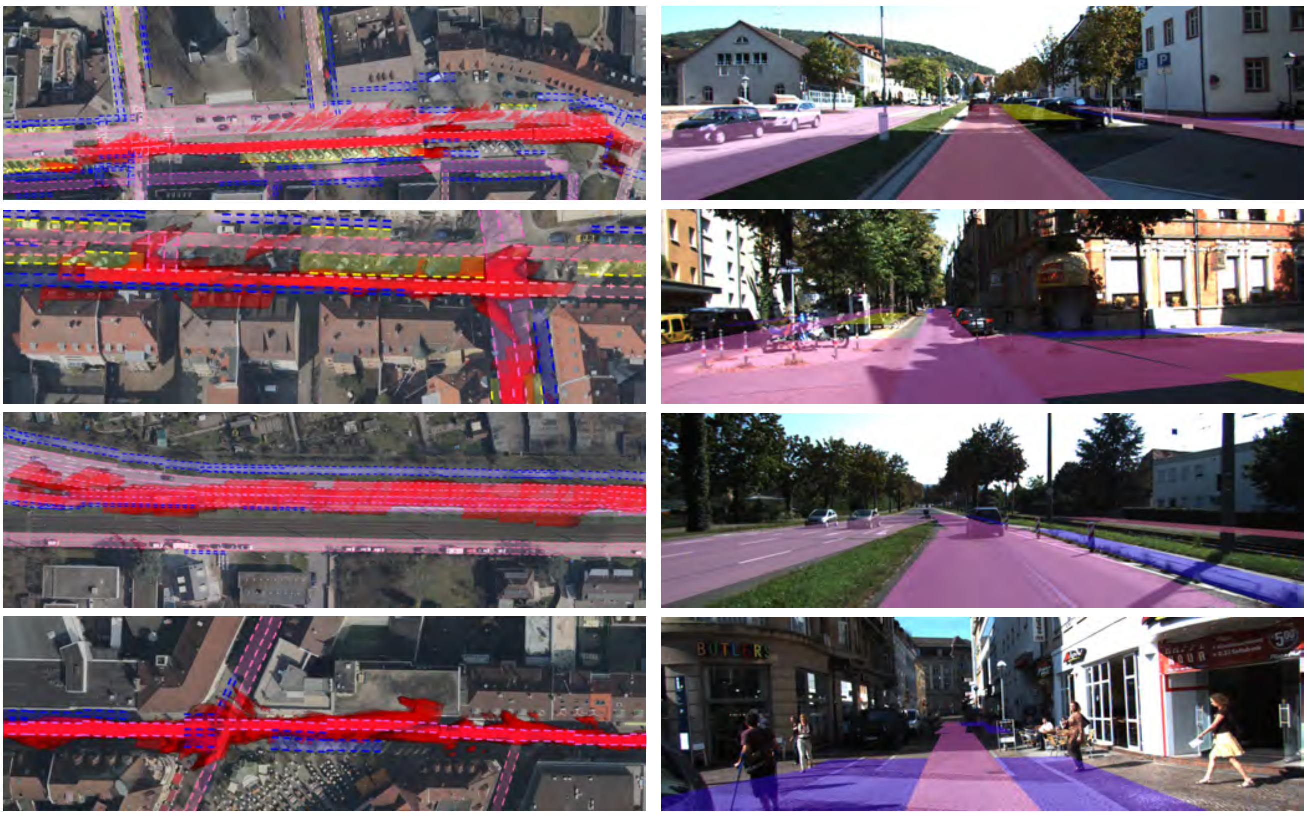


Figure 6. Left: The ground road detection with red projected into the aerial image after alignment and road layout estimation. Right: The semantic lanes projected back into the aligned ground image. These scenes are all challenging with parallel roads, parking spots and intersections. The bottom image is especially difficult since it is an urban pedestrian area. Note that the aerial and ground images were taken with several years difference in different seasons. Pink is road, blue is sidewalk and yellow marks parking spots.

only need 5 integral accumulators per ground image.

Ground alignment smoothness: This potential encodes the fact that two consecutive alignments should be similar.

$$\phi_{gsm}(t_k, t_{k+1}) = |t_k - t_{k+1}|$$

This assumes that GPS+IMU have smooth errors and no outliers.

3.2. Inference via Block Coordinate Descent (BCD)

Inference in our model can be performed by minimizing the energy function:

$$\mathbf{y}^*, \mathbf{t}^* = \underset{\mathbf{y},\mathbf{t}}{\operatorname{argmin}}\, E(\mathbf{y}, \mathbf{t}, \mathbf{x}_A, \mathbf{x}_M, \mathbf{x}_G)$$

with $E(\mathbf{y}, \mathbf{t}, \mathbf{x}_A, \mathbf{x}_M, \mathbf{x}_G)$ defined as in Eq. (1). Unfortunately, inference in our model is NP-hard, as our graphical model contains many loops. We thus take advantage of block coordinate descent to perform efficient inference. We refer the reader to Alg. 1 and Fig. 2 for inference steps.

Our block coordinate descent algorithm (BCD) alternates by doing inference in the direction along the road, doing inference in the direction perpendicular to the road and aligning the ground and aerial images. Note that when a road is not connected to a parallel road, the second step results in a graphical model with 15 variables, while when there are k parallel roads, this involves doing inference over a graphical model with $15k$ variables. Note also that in order to minimize the same objective, each of these iterations is performing conditional inference, and the pairwise potentials involving variables that are not optimized collapse to unaries.

3.3. Training with S-SVM

We employ structured SVM (S-SVM) [24] to learn the weights of the aerial unaries and the smoothness in our model. In particular, we use the parallel cutting plane implementation of [18]. We employ a combination of two loss functions. The first is a truncated L_2 loss: $\ell_{\text{data}} = \min(||y_i^j - \hat{y}_i^j||^2, 100m^2)$, encouraging our prediction y_i^j to be close to the ground truth $\hat{y}_i^j$. We compute $\hat{y}_i^j$ by performing inference in our model with features computed from the ground truth annotation (segmentation). The second loss term encourages smoothness of the prediction along the

Algorithm 1 Block coordinate descent inference (BCD).

```
1:  Set all alignments $\mathbf{t} = 0$, and initialize $\mathbf{y}$ by minimizing
    Eq. (1) ignoring the along road smoothness.
2:  repeat
3:      for for all $\mathbf{y}^j$ do
4:          Minimize Eq. (1) along the road w.r.t $\mathbf{y}^j$, holding
            the rest fixed.
5:      end for
6:      for all $\mathbf{y}_i$ at one segment of the road do
7:          Minimize Eq. (1) w.r.t $\mathbf{y}_i$, holding the rest fixed.
8:      end for
9:      for all $t$ variables do
10:         Minimize Eq. (1) w.r.t $\mathbf{t}$, holding $\mathbf{y}$ fixed.
11:     end for
12: until no energy reduction or max number iterations
```

road, $\ell_{sm} = |y_i^j - y_i^{j+1}|$. Note that the geometrical constraints in our model are either 0 or ∞ and are not trained.

4. Experiments

We collected a new dataset which we call *Air-Ground-KITTI*, which is composed of both ground images from the KITTI tracking benchmark [8] and newly acquired orthorectified aerial images over the same area. We neglected the KITTI sequences where the car is mostly static, resulting in 20 KITTI sequences for a total of 7603 ground stereo images. We annotated every 30th ground image with 4 semantic classes (*parking*, *sidewalk*, *road*, *building*). The aerial images were acquired by a DSLR camera mounted on an airplane and projected on the earth surface with 9 cm/pixel Ground Sampling Distance (GSD). We split the data into 10 training and 10 test aerial image/KITTI sequences, with special care to avoid overlaps in the aerial images. We manually annotated the aerial images with 4 categories (*parking*, *sidewalk*, *road*, *building*) as closed polygons and the lane markings as polylines. This took 70h of annotation, at a mean of $21h/km^2$, the area is 3.23 km^2.

To perform fine-grained segmentation using both aerial and ground images, we estimate a homography that transforms the ground plane in KITTI to the UTM coordinate system based on the KITTI's GPS+IMU measurements and the camera calibration. We assign each ground image to the closest parallel road segment. Our model then refines this estimate in the direction perpendicular to each road segment. We process every 5th ground image in the sequence.

As metrics for the fine-grained segmentation we calculate the pixelwise Intersection over Union (IoU), Precision, Recall and F1 metrics for three classes (*i.e.* road, parking, sidewalk). Note that we only measure the areas laying in the area of interest (*i.e.* ± 15m around the road map centerline). We consider two parallel roads overlapping over the same area as a serious error. To reflect this, we handle these areas as if they were background. The metrics in Table 1 are calculated according to this.

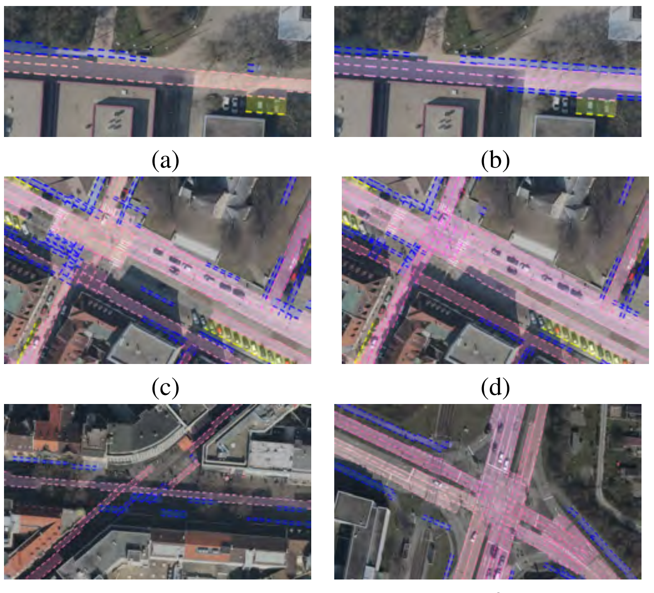


Figure 7. It is hard to estimate the number of lanes if there are no lane markings. (a) Our method, (b) Oracle (i.e., our method with ground truth potentials). (c) The *OnlyLane* model without the parallel constraint allows the road to "jump" to the nearby parallel road. (d) The parallel constraints of *LaneRoadParallel* prevents this from happing. (e) Dense, urban pedestrian streets are difficult to estimate. (f) Our model is not intended for intersections, as it does not reason about turn lanes.

For the roads, we additionally compute whether we have estimated the correct number of lanes. This is measured as the average ℓ_1 error in terms of number of lanes (*EN*). Note that if there are no lane markings, estimating the number of lanes is very difficult. Fig. 7 (a-b) shows this difficulty.

In our experiments, we compare our approach to the state-of-the-art method of [12], which uses OSMs to estimate road width. We also tested different model configurations for our approach. We refer to *Lane* as a model that employs *Aerial semantics, Aerial Edges, Road collapse constraints, Lane size constraint and Centerline prior* energy terms. Inference is done independently for each road segment via dynamic programming along the $\mathbf{y}^j = y_1^j, \cdots, y_{15}^j$ chains. We refer by *LaneParallel* to a model where we additionally include the constraint between nearby parallel road. We refer by *LaneRoad* as a model that contains all the potentials in *Lane* plus smoothness along the road. We apply BCD inference by alternating between the chains perpendicular to the road (the lanes) and along the roads (segments). We refer by *LaneRoadParallel* a model that contains all potentials but the ground. Finally, *Ground* contains all potentials. We evaluate this case only where ground images are available.

Model	*Average*		*Road*					*Sidewalk*				*Parking*			
	IoU	F1	IoU	F1	Pr.	R.	*EN*	IoU	F1	Pr.	R.	IoU	F1	Pr.	R.
Mattyus *et al.* [12]	-	-	62.1	76.4	68.0	87.0	-	-	-	-	-	-	-	-	-
[12] Deep Un*	-	-	64.4	78.4	66.7	**94.7**	-	-	-	-	-	-	-	-	-
Lane	43.6	59.6	61.9	76.5	82.8	71.0	0.730	31.8	48.3	67.2	37.7	37.0	54.1	58.5	50.3
LaneParallel	44.8	60.3	66.5	79.9	**85.0**	75.4	**0.543**	31.6	48.0	**69.8**	36.6	36.1	53.1	**70.8**	42.4
LaneRoad	45.4	61.6	61.9	76.4	82.7	71.0	0.707	38.3	55.4	62.4	49.7	36.1	53.1	52.2	**54.1**
LaneRoadParallel	**48.6**	**64.3**	**68.0**	**80.9**	83.5	78.5	0.555	**39.5**	**56.6**	63.5	**51.1**	**38.4**	**55.5**	63.8	49.1
LaneRoadParallel**	41.9	58.5	54.9	70.9	**86.9**	59.9	0.559	**34.9**	**51.7**	**68.7**	**41.5**	**35.8**	**52.7**	**69.9**	**42.3**
Full**	**42.0**	**58.6**	**55.3**	**71.2**	86.8	**60.4**	**0.556**	**34.9**	**51.7**	**68.7**	**41.5**	**35.8**	**52.7**	**69.9**	**42.3**

Table 1. Performance for the semantic classes (*i.e. road, parking spot, sidewalk*) with various models and the two baselines. The values are in %, except *EN* which is the average road lane number l_1 error with respect to the oracle. * Marks the method of [12] with our deep road classifier. The last two rows marked with ** evaluate only over areas where ground images are also available.

	GPS+IMU [m]	Ours [m]
Alignment error	1.67	**0.57**

Table 2. Ground to air image misalignment based on the camera calibrations (GPS+IMU) and after our alignment measured in meters. Using ±4 meter interval.

Comparison to the state-of-the-art: As shown in Table 1, our method outperforms [12] in almost all metrics, even when we apply our deep features instead of their road classifier in their method. Furthermore, we retrieve more semantic categories such as sidewalk, individual road lanes and parking. The constraint between parallel roads is important to achieve good results on roads. Without it, our model cannot outperform [12], which has this constraint.

Deep semantic features in aerial Images: We show the performance of our Deep Network in Fig. 4. Note that it is much better than the road classifier of [12].

Alignment between aerial and ground images: As shown in Table 2 and Fig. 3 reasoning about the alignment between ground and aerial images while doing fine-grained segmentation improves the alignment significantly.

Qualitative Results: We visualize our results when using only aerial images in Fig. 5, and when using joint aerial and ground reasoning in Fig. 6. Our approach is able to estimate well the lanes, sidewalk and parking as well as the alignment between the ground and the aerial images.

Ablation studies: As shown in Table 1, the metrics for different versions of our model are fairly similar, however qualitatively, as we add more potentials, the results get better. This is illustrated in Fig. 7 (c), where the *OnlyLane* model moves the middle road to a parallel road resulting in a noncontinuous structure. In contrast, the *LaneRoadParallel* model prevents overlaps and favors smoothness, see the Fig. 7 (d). Including the ground images only slightly improves performance. We believe this could be overcome by using stronger features in the ground images, i.e., leveraging the full 3D point cloud, not just the ground plane. Note that since our approach gives us very precise alignments between the ground and the aerial images it could be used to enhance OSM with object locations, e.g. traffic signs.

Inference time: Inference in our full model takes 6 seconds per km of road, with a single thread on a laptop computer. Note that BCD can easily be parallelized.

Limitations: Our model is designed for individual roads and it does not reason about turning lanes connecting different roads at intersections (see Fig. 7 (f)). Dealing with such scenarios is part of our future work. Semantic segmentation from aerial images reasons mainly about the visible parts of the street. Therefore covered areas (*e.g.* by building, bridges, trees) can be a problem. However, when ground images are available, our approach can handle this problem. Our aerial images were acquired in early spring, and thus trees occluding the roads is not a big problem.

5. Conclusion

We proposed an approach to enhance existing freely available maps with fine grained segmentation categories such as parking spots and sidewalk, as well as the number and location of road lanes. Towards this goal, we proposed an efficient method that produces very accurate estimates by performing joint inference over both, monocular aerial imagery captured by a plane and ground images taken from a stereo pair mounted on top of a car. We have demonstrated the effectiveness of our approach on a new dataset which enhances KITTI with aerial images taken with a camera mounted on an airplane and flying around the city of Karlsruhe. In the future, we plan to reason about other fine grained categories such as traffic signs in order to further enhance the maps. As our method reasons about the accurate alignment between the map and the ground images, we envision its use for precise, lane-wise self localization of the vehicle on the road.

Acknowledgments

We would like to thank Viktoria Zekoll, Stefan Turzer and Márk Bagdy for making the laborious annotation work.

References

[1] http://fortune.com/2015/10/16/how-tesla-autopilot-learns/. 1

[2] M. Barzohar and D. Cooper. Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation. *PAMI*, 1996. 2

[3] M. A. Brubaker, A. Geiger, and R. Urtasun. Lost! leveraging the crowd for probabilistic visual self-localization. In *CVPR*, 2013. 2

[4] D. Chai, W. Forstner, and F. Lafarge. Recovering line-networks in images by junction-point processes. In *CVPR*, 2013. 2

[5] L.-C. Chen, A. G. Schwing, A. L. Yuille, and R. Urtasun. Learning deep structured models. *ICLR*, 2015. 4

[6] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013. 4

[7] C. Forster, M. Pizzoli, and D. Scaramuzza. Air-ground localization and map augmentation using monocular dense reconstruction. In *IROS*, 2013. 2

[8] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *IJRR*, 2013. 1, 7

[9] E. Kalogerakis, O. Vesselova, J. Hays, A. A. Efros, and A. Hertzmann. Image sequence geolocation with human travel priors. In *ICCV*, 2009. 2

[10] T.-Y. Lin, S. Belongie, and J. Hays. Cross-view image geolocalization. In *CVPR*, 2013. 2

[11] T.-Y. Lin, Y. Cui, S. Belongie, and J. Hays. Learning deep representations for ground-to-aerial geolocalization. In *CVPR*, 2015. 2

[12] G. Mattyus, S. Wang, S. Fidler, and R. Urtasun. Enhancing road maps by parsing aerial images around the world. In *ICCV*, 2015. 2, 4, 7, 8

[13] K. Matzen and N. Snavely. Nyc3dcars: A dataset of 3d vehicles in geographic context. In *ICCV*, 2013. 2

[14] H. Mayer, S. Hinz, U. Bacher, and E. Baltsavias. A test of automatic road extraction approaches. In *ISPRS*, 2006. 2

[15] V. Mnih and G. E. Hinton. Learning to detect roads in high-resolution aerial images. In *ECCV*, 2010. 2

[16] V. Mnih and G. E. Hinton. Learning to label aerial images from noisy data. In *ICML*, 2012. 2

[17] J. Porway and Q. W. ands Song Chun Zhu. A hierarchical and contextual model for aerial image parsing. *IJCV*, 88(2):254–283, 2009. 2

[18] A. G. Schwing, S. Fidler, M. Pollefeys, and R. Urtasun. Box In the Box: Joint 3D Layout and Object Reasoning from Single Images. In *ICCV*, 2013. 6

[19] A. G. Schwing and R. Urtasun. Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, 2015. 3, 5

[20] Y.-W. Seo, C. Urmson, and D. Wettergreen. Exploiting publicly available cartographic resources for aerial image analysis. In *SIGSPATIAL*, 2012. 2

[21] Y.-W. Seo, C. Urmson, and D. Wettergreen. Ortho-image analysis for producing lane-level highway maps. Technical Report CMU-RI-TR-12-26, CMU, 9 2012. 2

[22] Q. Shan, C. Wu, B. Curless, Y. Furukawa, C. Hernandez, and S. Seitz. Accurate geo-registration by ground-to-aerial image matching. In *3DV*, 2014. 2

[23] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 3, 5

[24] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 2005. 6

[25] S. Wang, S. Fidler, and R. Urtasun. Holistic 3d scene understanding from a single geo-tagged image. *CVPR*, 2015. 2

[26] J. D. Wegner, J. A. Montoya-Zegarra, and K. Schindler. A higher-order crf model for road network extraction. In *CVPR*, 2013. 2

[27] J. Yuan and A. Cheriyadat. Road segmentation in aerial images by exploiting road vector data. In *COM.geo*, 2013. 2