DETECTING, TRACKING, AND RECOGNIZING ACTIVITIES IN AERIAL VIDEO

by

VLADIMIR REILLY
B.S. University of Central Florida
M.S. University of Central Florida

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2012

Major Professor: Mubarak Shah

ABSTRACT

In this dissertation, we address the problem of detecting humans and vehicles, tracking them in crowded scenes, and finally determining their activities in aerial video. Even though this is a well explored problem in the field of computer vision, many challenges still remain when one is presented with realistic data. These challenges include large camera motion, strong scene parallax, fast object motion, large object density, strong shadows, and insufficiently large action datasets. Therefore, we propose a number of novel methods based on exploiting scene constraints from the imagery itself to aid in the detection and tracking of objects. We show, via experiments on several datasets, that superior performance is achieved with the use of proposed constraints.

First, we tackle the problem of detecting moving, as well as stationary, objects in scenes that contain parallax and shadows. We do this on both regular aerial video, as well as the new and challenging domain of wide area surveillance. This problem poses several challenges: large camera motion, strong parallax, large number of moving objects, small number of pixels on target, single channel data, and low frame-rate of video. We propose a method for detecting moving and stationary objects that overcomes these challenges, and evaluate it on CLIF and VIVID datasets. In order to find moving objects, we use median background modelling which requires few frames to obtain a workable model, and is very robust when there is a large number of moving objects in the scene while the model is being constructed. We then remove false detections from parallax and registration errors using gradient information from the background image. Relying merely on motion to detect objects in aerial video may not be sufficient to provide complete information about the observed scene. First of all, objects that are permanently stationary may be of interest as well, for example to determine how long a particular vehicle has been parked at a certain location. Secondly, moving vehicles that are being tracked through the scene may sometimes stop and remain stationary at traffic lights and railroad crossings. These prolonged periods of non-motion make it very difficult for the tracker to maintain the identities of the vehicles. Therefore, there is a clear need for a method that can detect stationary pedestrians and vehicles in UAV imagery. This is a challenging problem due to small number of pixels on the target, which makes it difficult to distinguish objects from background clutter, and results in a much larger search space. We propose a method for constraining the search based on a number of geometric constraints obtained from the metadata. Specifically, we obtain the orientation of the ground plane normal, the orientation of the shadows cast by out of plane objects in the scene, and the relationship between object heights and the size of their corresponding shadows. We utilize the above information in a geometry-based shadow and ground plane normal blob detector, which provides an initial estimation for the locations of shadow casting out of plane (SCOOP) objects in the scene. These SCOOP candidate locations are then classified as either human or clutter using a combination of wavelet features, and a Support Vector Machine. Additionally, we combine regular SCOOP and inverted SCOOP candidates to obtain vehicle candidates. We show impressive results on sequences from VIVID and CLIF datasets, and provide comparative quantitative and qualitative analysis. We also show that we can extend the SCOOP detection method to automatically estimate the

orientation of the shadow in the image without relying on metadata. This is useful in cases where metadata is either unavailable or erroneous.

Simply detecting objects in every frame does not provide sufficient understanding of the nature of their existence in the scene. It may be necessary to know how the objects have travelled through the scene over time and which areas they have visited. Hence, there is a need to maintain the identities of the objects across different time instances. The task of object tracking can be very challenging in videos that have low frame rate, high density, and a very large number of objects, as is the case in the WAAS data. Therefore, we propose a novel method for tracking a large number of densely moving objects in an aerial video. In order to keep the complexity of the tracking problem manageable when dealing with a large number of objects, we divide the scene into grid cells, solve the tracking problem optimally within each cell using bipartite graph matching and then link the tracks across the cells. Besides tractability, grid cells also allow us to define a set of local scene constraints, such as road orientation and object context. We use these constraints as part of cost function to solve the tracking problem; This allows us to track fast-moving objects in low frame rate videos.

In addition to moving through the scene, the humans that are present may be performing individual actions that should be detected and recognized by the system. A number of different approaches exist for action recognition in both aerial and ground level video. One of the requirements for the majority of these approaches is the existence of a sizeable dataset of examples of a particular action from which a model of the action can be constructed. Such a luxury is not always possible in aerial scenarios since it may be difficult to fly a large number of missions to observe a particular event multiple times. Therefore, we propose a method for

recognizing human actions in aerial video from as few examples as possible (a single example in the extreme case). We use the bag of words action representation and a 1vsAll multi-class classification framework. We assume that most of the classes have many examples, and construct Support Vector Machine models for each class. Then, we use Support Vector Machines that were trained for classes with many examples to improve the decision function of the Support Vector Machine that was trained using few examples, via late weighted fusion of decision values.

To my parents.

ACKNOWLEDGMENTS

I would like to thank Dr. Shah for his guidance and support. His constant motivation and faith in my ability was a very important factor for the completion of this work. I am grateful to Dr. Saad Ali, for his guidance during my first research endeavor. I would also like to thank my friends and colleagues: Berkan Solmaz, Haroon Idrees, Soumyabrata Dey, Dr. Paul Scovanner, Dr. Mikel Rodriguez, Omar Oreifej, Dr. Ramin Mehran, Enrique Ortiz, Dr. Saad Ali, Dr. Saad Masood, Dr. Imran Saleemi, and others for their support. I am also grateful to Dr. Kenneth Stanley, Dr. Michael Georgiopoulos, and Dr. Aristide Dogariu for serving on my committee.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

One of the most fundamental applications of computer vision, involves making computer systems aware of the content of the observed scene. This content includes moving and stationary entities of interest, as well as the actions performed by them. This awareness can be put into a wider context with the use of video data captured from a moving UAV platform, which can observe a much larger area by either moving to a different area of the scene, and/or utilizing a wide area sensor. Object detection, tracking, and activity recognition are active research areas. However most of the methods have been designed to deal with consumer imagery or ground level surveillance, and may not necessarily be applicable to the aerial surveillance scenario. Even the methods that are specifically designed to deal with aerial imagery have difficulty when presented with real data, which may have the additional challenges of large camera motion, low resolution, strong parallax, fast moving objects and strong shadow.

In this thesis, we propose methods for detecting stationary and moving objects, tracking their identities in a dense environment, and recognizing their actions from few examples. In order to deal with the challenges of low resolution, shadows, fast camera motion, and parallax, we rely on scene specific information that we extract directly from the imagery, both with and without the use of auxiliary metadata. We show that this additional information vastly improves the tasks of detecting and tracking objects.

1.1 Overview and Motivation

Unmanned Aerial Vehicles, or UAVs, are becoming more widespread in both military and civilian applications, including surveillance, rescue, and reconnaissance [75] [76] [55]. In the course of these operations, video data containing useful information is collected. This information may be useful during the mission itself or may become useful at a later date. The ever-increasing number of UAV missions equates to a backlog of data which can become quite large, as it requires many man-hours to analyze the data manually. This calls for automated video analysis tools with capabilities that include registration [80], object detection [34], tracking [76] [83], classification [77], and scene and action analysis [13] [35] [72].

The main purpose of utilizing a UAV for surveillance is the mobility and the increased area of coverage that the UAV can provide. Recently, a new sensor platform has appeared on the scene characterized by an extremely large field of view, allowing for persistent monitoring of very large areas. Data obtained from such a sensor is quite different from the standard aerial and ground surveillance datasets, such as VIVID and NGSIM, which have been used in [31, 51], and [1, 78, 32]. The sensor utilizes multiple high-resolution cameras to capture an area that is much larger than the standard datasets, which allows for thousands of objects to be visible at one time. Specifically, in the case of CLIF dataset, the sensor covers $25km^2$, compared to only $0.24km^2$ in the VIVID dataset. This wide area coverage comes at a price of low frame rate, only 1.2 Hz on average compared to 15Hz in VIVID. The combination of low framerate, moving camera, and a large number of densely moving objects create a difficult mix of challenges that renders previously developed automated surveillance methods inapplicable to this data. The very wide area coverage allows one to place the observed objects into a much broader context, requiring a capability to be able to process this kind of data.

When aerial surveillance is performed in an urban environment, not all objects of interest are moving all of the time. Traffic lights or traffic jams periodically cause objects to stop. Therefore, the system must be able to detect objects when they are stationary. One straightforward approach to this problem is to apply a state-of-the-art static frame detection algorithm, such as [14] [18] [38] [47] [59], over the entire frame of the video. This approach, however, runs into the problem of small object size in aerial video, which may make it impossible to construct a meaningful model. Methods that perform part detection explicitly, such as [18] [47] [12] [70] [67], will not be able to construct meaningful models for individual parts at very low resolutions. Bag-of-feature methods, such as [38], also have difficulty constructing models because only a small number of interest points can be found.

The objects are so small that even holistic methods such as [68] [14] [69] [7] will have difficulty extracting sufficient discriminative information. Another issue introduced by the small object size is the need to process a very large number of windows across an entire image. This obviously increases processing time and generates many false positives, especially if the object model is not sufficiently discriminative. The above problems can be further compounded by motion blur and varied orientation of objects within the scene. Therefore, in order to tackle these issues of size and performance, we propose avoiding full frame search, and instead opt to constrain the search using a set of geometric constraints.

In addition to detecting and tracking objects in the scene, the system must also be able to detect the individual actions that performed by those objects. Like all pattern recognition problems, this task requires the method to be able to account for variation within the particular class that one is trying to recognize or detect. This is especially true for recognizing the actions of humans, where the same action can be performed by different actors of different size, different body proportions, and different kinematics. There are two ways to account for that intraclass variation which are commonly used. The first way is to utilize a representation of the action that is sufficiently semantically meaningful, so that it is robust to those intra-class variations. The second way is to construct the model of the action in such a way that it can account for variation within each class. This is typically accomplished by using machine learning to generate the model automatically from a fairly large collection of examples [72] [62] [49]. The problem with the first approach is that the action class may be too complex to allow for a semantically meaningful representation, or the representation may be too difficult, in practice, to generate for the particular dataset. The problem with the second approach is that it is not always possible to collect an extensive dataset of a particular action due to an insufficient number of available actors or resources. This is particularly true in the case of aerial video, since one needs to perform the actions, and to fly the UAV with the sensor platform as well. Therefore, there is a need for a well-performing action recognition framework, which has the capability to construct the model of the action using as few examples as possible.

1.2 Contributions

In this thesis, we develop several methods for processing aerial video data under a number of challenging conditions. These methods allows us to detect, track, and recognize the activities of objects that are moving in aerial video. Unlike the previous approaches which we discuss in Chapter 2, our suggested framework can deal with the challenges of low frame rate, high object density, strong shadows, strong parallax, and insufficiently large action datasets. We demonstrate how detection and tracking methods can be improved by utilizing additional scene information, which can be extracted with or without the help of metadata. We also show how we can recognize human actions from the air, without relying on a large dataset of examples in order to construct the model of a particular action.

1.2.1 Detecting Moving and Stationary Objects in Aerial Imagery

First, we propose a novel framework for detecting moving and stationary humans and vehicles in aerial imagery in the presence of strong parallax and shadow. In order to detect moving objects, we register the consequent frames using interest points and RANSAC to compensate for global camera motion. Then, we take the registered frames and construct a model for the background using the median operator, and perform background subtraction. Median background subtraction allows us to detect moving objects in just 10 frames, and is capable of constructing a good background model, even for areas in the scene, which have dense traffic, meaning that the background is badly contaminated with outlier moving objects. Next, we suppress false motion detections that have been detected due to parallax and not actual moving objects. Homography based motion compensation assumes that the scene is planar. However, in the case of real aerial video, the scene may not necessarily be planar, and interest points that are used during the registration stage are not guaranteed to lie on one plane. This results in a registration plane which will not correspond to any of the real planes in the scene. Due to this, parts of the scene will appear to move even in motion compensated imagery. We suppress these false detections by computing the gradient of the median background image, and subtracting it from the difference image.

The key idea behind our approach for detecting stationary objects is to constrain the search for the objects in the image by assuming that humans are upright shadow casting objects, and vehicles are box-like shadow casting objects. We utilize oriented low level computer vision techniques based on a set of geometric scene constraints derived from the metadata of the UAV platform. Specifically, we utilize the projection of the ground plane normal to find blobs normal to the ground plane; these blobs give us an initial set of potential out of plane object candidates. Similarly, we utilize the projection of shadow orientation to obtain a set of potential shadow candidates. Then, we obtain a refined set of Shadow Casting Out Of Plane, or SCOOP, candidates, which are pairs of shadow and normal blobs that are of correct geometric configuration, and relative size. This is done based on projected directions, as well as the ratio of assumed projected human height and projected shadow length. Once the refined set of candidates has been obtained, we extract wavelet features from each SCOOP candidate, and classify it as either human or clutter using a Support Vector Machine (SVM).

Since vehicles are also out of plane shadow casting objects. Application of our SCOOP candidate detection method produces consistent candidates belonging to vehicles as well. We show that a vehicle can be represented as a combination of a SCOOP and an inverted SCOOP. Therefore, we can combine the SCOOP and inverted SCOOP into a vehicle candidate, and then classify it as either a vehicle or clutter using the same framework we utilized for for humans.

Note that the main idea behind our geometric constraints is to improve the performance of any detection method by avoiding full frame search. Hence, other models, features, and classification schemes suitable for aerial imagery can be used. Additionally, our method can be used to alleviate object localization problems associated with motion detection in presence of strong shadow.

Our method has the following assumptions and operational requirements:

First, we assume that the objects in the scene are casting a shadow. If the weather prevents the casting of shadow, then human detection can still proceed, using only the blobs related to the ground-plane normal as a constraint at the cost of degraded performance. In the case of vehicles, the detection can still proceed as though the shadow is there, however it will also result in degraded performance.

Second, we assume that humans are sufficiently upright for their shadow to be visible. The human can be standing or sitting down, but can not be lying on the ground.

Third, we use metadata associated with the UAV imagery in order to determine the shadow orientation in the world. We also use it to find the orientation of ground-plane normal and shadow in the imagery, as well as their relative size. Errors in the metadata result in incorrect orientations in the image. However, the SCOOP detection method has approximately $+10^\circ - 10^\circ$ robustness to errors in orientation. If metadata is not available, we have developed a method to determine the shadow orientation automatically assuming that the ground-plane normal is fixed.

Fourth, we make a planar scene assumption when we determine the orientation of the shadow in the world and when we project it into the image. When the scene is not planar, this assumption has an effect equivalent to that of incorrect metadata, where the shadow's orientation and/or length in the image is not correct. However, since our method is resistant to orientation errors, its performance will degrade gradually as the surface roughness increases. Note that if a Digital Elevation Map (DEM) is available, we do not have to make this assumption. Rather, we can explicitly compute different shadow orientations for different surfaces in the scene and regions of the image. The automated orientation detection method requires two shadow casting objects to be present on a plane in order to find the correct shadow orientation for that plane.

1.2.2 Tracking a Large Number of Densely Moving Targets in Low Frame Rate Video

Next, we propose a framework for tracking large numbers of densely moving objects in low frame rate video. First, we break the scene up into a series of overlapping cells defined in image space, and formulate the tracking problem as a series of bipartite graphs confined to the individual grid cells. Then, solve it optimally within each grid cell using the Hungarian algorithm. The cell idea accomplishes two things. First of all, it speeds up the solution of the assignment problem by greatly reducing the number of objects that we have to consider within each cell, since the complexity of the Hungarian algorithm is $O(N^3)$, where N is the number of objets. Secondly, it allows us to locally define scene constraints that we can use in order to generate better assignment weights between objects at different time instances.

We construct the graphs to take into account the possibility of object disappearance and reacquisition by adding occlusion nodes and connecting observations to labels from several frames back. Additionally, within each grid we define a set of constraints to allow the assignment of labels to observations, even when the velocity estimate of an object is not available initially. Specifically, we analyze the possible assignments within each cell, in order to compute the dominant orientation of motion within the grid cell. We also construct a model for the contextual relationship between objects and their neighbors. This information is then utilized for assignment weight generation when we construct the bipartite graph, in order to aid with the initial assignment of objects. The main purpose of this is to overcome the problem of low frame-rate and high object density. When one set of objects covers a large distance between the frames, and the set of objects that is travelling behind them, move into an area that is close to the initial positions of the first set of objects. Making a tracking assignment based purely on the distance between old and new positions will invariably result in an incorrect assignment. We demonstrate that when we utilize our derived local scene constraints in the context of high object density and low frame rate, the tracking performance improves.

1.2.3 Action Recognition In Aerial Video From Few Examples

Finally, we propose a method to recognize human actions in aerial video, even when there are few examples from which we can learn the model of the action (in the extreme case, only one). We accomplish this by utilizing transfer learning to transfer knowledge from other classes in the same dataset. We use the the popular representation of actions known as bag of words. We detect primitive spatio-temporal features in the videos, which indicate change in the gradient over time. We extract gradient features from these cuboids and apply Principle Component Analysis to reduce their dimensionality. Then, we cluster the cuboids into spatio-temporal words, and finally represent the video as a histogram of these words. Since the camera is moving in aerial video, we compensate it's motion prior to detecting the points.

This is a multi-class problem, therefore, we train several OnevsAll Support Vector Machine classifiers in order to represent all the actions. We assume that some of the classes have many examples and are learned normally, while the number of classes that contain few examples is small. Our goal is to utilize the models of the classes that have many examples to augment the models of the classes that have few examples. We do this via a late fusion technique during testing, where we modify the decision function of the OnevsAll classifier for the class which contains few examples to include weighted outputs of the classifiers of similar classes which contain many examples. We show how the weights and the classes can be determined automatically, and that they are related to the probability of the class being misclassified as other classes.

1.3 Organization of the Thesis

The thesis is structured as follows: **Chapter** 2 reviews existing literature on object detection, tracking, shadow detection, and action recognition, and attempts to contrast those works with our contribution. **Chapter** 3 presents a framework for detecting moving and stationary objects in the presence of strong parallax and shadow. **Chapter** 4 proposes a novel method for tracking a large number of dense fast-moving targets. **Chapter** 5 describes a novel framework for detecting human actions in aerial video using few examples. Finally, the thesis is concluded in **Chapter** 6, with a summary of contributions and a description of future work.

CHAPTER 2: LITERATURE REVIEW

In this chapter, we discuss some of the prominent approaches for detecting moving and static objects, tracking their identities across multiple frames, detecting their shadows, and classifying their actions. The previous works include methods that have been developed both for ground and aerial view, since the key ideas can be applied across different domains.

2.1 Moving Object Detection and Tracking

There is extensive literature on background subtraction for moving object detection in stationary cameras [65] [30] [85] [31] [25] [46] [21]. The general framework for these approaches is to model the most common observations for each pixel in order to represent the background. Once this is done, the moving objects are detected on a per-pixel basis as outliers to the model, and the model is updated. The models proposed range in sophistication depending on the problem domain and the data. In [21], Friedman and Russel model the background as a single gaussian distribution in gray-scale imagery. Foreground objects and shadows are also modelled as single Gaussian distributions, and are separated from the background and each other using heuristics. In [65] and [25], the background for each pixel in color video is modelled as a mixture of Gaussians. The idea there is that observations of a background pixel can be generated from multiple processes - the actual static background and moving background of limited dynamics, such as a tree branch. Therefore, the different components of the

Gaussian mixture will correspond to different processes which have produced the observations. Pixel values at each time instance are compared against the mixture model, and if a genuine moving object appears in the scene, its color value will be different from all of the components in the mixture and it will be labelled as a moving object. Mittal and Paragios in [46] allow for greater dynamism of the background by constructing a non-parametric Kernel Density Estimation model of optical flow for the background. This allows them to handle a background that consists of dynamic textures, such as the ocean. A dynamic texture model based on Autoregressive Moving Average is used for a similar purpose in [85]. Complex statistical models generally require a fairly large number of observations, which are easily available in the case of long term static surveillance and high frame rate video; However, in the case of aerial video on the other hand, the luxury of persistent observation without large camera motion is very seldom available.

In [1], the authors detect and track moving objects in high frame rate aerial video. They use gradient based direct registration to compensate for global camera motion, then they utilize cumulative frame difference to detect moving objects. Finally, they track the moving objects by constructing probability distributions of the colors contained within the moving blob of the object. Unfortunately, appearance and color based tracking is not that meaningful in single channel imagery, since most of the objects appear to be either white or different shades of gray. The authors also assume a homographic model of global motion compensation, and do not take into account parallax errors caused by the 3D structure of the scene.

One popular method of handling false detections due to parallax is the plane plus parallax model, which was utilized in [32] [29] and [61]. The key idea behind this particular framework is to assume that the scene is dominated by just one plane, and there are a few isolated out-of-plane structures. When these assumptions hold and a robust registration method is utilized, a valid homography can be estimated for pixels belonging to the ground plane. Since the homography does not hold for the out of plane objects, they are also are detected as ‘movers’ during background subtraction. Finally, detections are filtered as either moving or stationary out of plane by applying two or three view geometric constraints based on the fundamental matrix or the tri-focal tensor. The issue with the above framework is that while it is more general than the initial planar scene assumption, it is not sufficiently general to deal with scenes that do not have a dominant plane, or scenes that are so cluttered with out of plane objects that it prevents the correct estimation of homography for any actual plane in the scene. The former scenario was the one found in the WAAS data that we experimented with. The data was captured in mostly urban and sub-urban environments, which have a large number of buildings and trees of different heights. Therefore, the interest points used for registration belong to different planes, and as a result the registration plane does not correspond to any of the planes that are in the scene. Because of this, areas of misalignment, drift, and therefore false motion detections can happen on the ground-plane as well.

In [78], Xiao et.al focus exclusively on detecting and tracking moving vehicles in aerial video. In order to suppress false detections due to parallax, the authors exploit metadata associated with the imagery. They use it to aid in stereo-like 3D reconstruction of buildings present in the scene, as well as color, texture, and depth based classification to detect vegetation. The 3D knowledge is exploited to suppress false detections that appear on out of plane objects. The authors also perform GIS guided road detection, which is used in tracking as an estimate of the direction of the motion of objects. In order to perform 3D reconstruction as suggested by the authors, a complex geo-registration process is required in order to refine the metadata associated with the imagery. However, the geo-registration process itself requires fairly accurate metadata to begin with, which may not always be available or correct. The GIS information for determining road orientation may be unavailable, additionally the use of GIS requires geo-registration. In contrast we show that our constraints can be derived directly from the scene without relying on metadata. When we do use metadata, we do not require it to be as accurate as georegistration does.

In [28], the authors combine the output of supervised class based segmentation, unsupervised gradient and color based segmentation, and moving object segmentation to detect different types of blobs in the scene. Supervised class based segmentation relies on color and gradient information to refine the boundaries of the blobs, which were obtained by the supervised segmentation. The resulting blobs are tracked through the scene, and their labels are refined using adjacency information via a Bayesian graphical model. As pointed out by the authors, semantic blob detection is not particularly reliable and has to be refined using Mean-Shift color segmentation. The use of Mean-Shift color segmentation will not give the best results in single channel data.

The tracking problem can typically be separated into two main stages: the assignment weight generation stage, which computes the likelihood of each track being assigned to a new observation, and the actual assignment stage, where the tracks or labels are assigned to new observations. Occasionally there will be a third correction stage to improve the quality of the tracks. A large variety of weight generation, matching, and correction frameworks exist (see

[83] for a survey of these approaches). In the context of surveillance, a combination of kinematic and appearance features is a popular framework for tracking targets in both aerial [51] and ground views [33]. In both [51] and [33], the authors use a template based approach and greedy assignment strategy for obtaining initial short tracks or tracklets, then in the correction stage they obtain longer tracks by utilizing kinematic information, such as velocity and acceleration of the individual tracklets, and a Gibbsian model of appearance of the tracklets. In our case however, appearance does not provide sufficient discriminative information due to the small number of pixels on the target, single channel gray-scale video, and dense targets. Relying exclusively on kinematics during initialization is unreliable.

Hardware has been one way to tackle the problem of tracking dense, fast moving targets. In [5] [74], the authors track large numbers of flying bats by utilizing several 125Hz infrared cameras to observe the same scene. The authors perform tracking in each view individually, while simultaneously estimating cross-camera correspondences, and fusing the tracks across views in 3D, and correcting incorrect assignments that occurred in individual cameras. The high speed of the cameras improves the effectiveness of matching initial assignments based purely on proximity, while multiple overlapping cameras provide a way of disambiguating assignments due to the density of objects by 3D reconstruction of the trajectories and observations. Unfortunately, in our case of WAAS data, the cameras do not have sufficient overlap or relative translation to allow for 3D reconstruction of trajectories.

2.2 Static Object Detection in Ground Level Imagery

Static object detection can generally be separated into and holistic methods [68] [69] [7] [14],parts based methods [18] [47] [12] [38] [40] [70] [67]. Holistic methods construct a model for the entire human, and then search over the image utilizing sub-window search, commonly following the framework of feature extraction and classification. In a seminal work by Dalal and Triggs [14], objects (in their case humans) are represented by a histogram of oriented gradients. First gradient is computed over an image, then the image is separated into overlapping cells. Then, a histogram of orientations of the gradient are computed within each cell and concatenated into one vector. Finally, a Support Vector Machine is trained on these vectors, in order to separate images containing humans from images containing background. Covariance features are used in [68], where covariance matrices of various features are computed in overlapping windows of the image of a human. Covariance matrices lie on a manifold and cannot be handled by standard vector based machine learning methods. Hence, the authors project the covariance features onto a subspace locally tangential to the manifold, and use LOGITBOOST to classify them.

The disadvantage of holistic approaches is that they are sensitive to the object of interest being occluded. One solution to this is parts based approaches. In [18], the authors train a holistic support vector machine using histograms of oriented gradients as features, as described in [14]. In addition to this main holistic template, which the authors call the ”root” filter, they train a collection of models for parts of the human. During testing, the confidence of a detection is given by the confidence of the holistic root filter, and the sum of the confidences of individual parts filters minus some deformation cost, which are defined within the root filter via a star graph. The models and locations for the parts are learned automatically via latent SVM.

In [47], Mikolajczyk et. al. use a collection of supervised parts detectors to detect humans. They use 7 manually specified parts for the front/profile of the head, front/profile of the face, front/profile of the upper body, and the legs. A set of candidate parts is detected using co-occurrence features and ADABOOST classifier. The final location of the human is determined by a Bayesian decision function as an optimal configuration of the parts and their confidences. Wu and Nevatia in [70] use a combination of edgelet features and ADABOOST to learn holistic model for the human as well as a collection of parts such, as the head, torso, and legs. However, when combining parts into humans they propose utilizing a joint image likelihood, to take care of multiple inter-occluded humans.

In cases of aerial video, as the size of objects becomes smaller and smaller, the feasibility of using static frame detection algorithms decreases. Static object detection methods are generally developed for datasets that have objects which are fairly large, such as INRIA, however they run into limitations when the size of the objects decreases. We can see this fact by using [14] as an example. The authors report that on human images of 128x64, the optimal cell size was 6x6, which corresponds to 4.69% of the height and 9.38% percent of the width. In the case of VIVID (an aerial video dataset), the humans are only 24x14, the smallest cell size possible is 2x2, which corresponds to 8.3% of the height and 14.28% of the width. Therefore, it is no longer possible to generate a descriptor using optimal settings, which in turn hurts the crucial requirement of a full frame classifier of achieving a low rate of false positives per window.

This fact is very problematic because as the size of humans decreases, the number of windows that one has to search increases. For example, for humans of size 128x64, image size 480x640 and window shift of 8, there are 26624 windows to consider at single scale (ignoring the border). However, for humans of size 24x14 and window shift of 2, the number of windows increases to 143528. In the case of high resolution imagery, the search space was reduced by about a third in [23] through clever re-sampling of the search space. That reduction may still not be sufficient in the case of aerial video. That approach also needs a classifier that has low sensitivity to small translations and scale shifts, hence it is not universal.

2.3 Static Object Detection In Aerial Imagery

Previous works that specifically deal with aerial imagery ([77] and [48]), opt to constrain the search by extracting additional features via preliminary processing. Miller et al use features points to constrain the search for humans [48] by assuming that at least one Harris corner feature point will be detected on the human in each frame. This generates a large number of candidates, which are then suppressed through tracking, the Harris corners in global reference frame. Then, each corner is classified using an Optimum Trade-off Maximum Average Correlation Height (OT-MACH) filter. If a track contains more human classifications than 20% of the total track length, all points within the track are labelled as human. The problem with the above approach is the large number of potential human candidates, they report 200 for a 320x240 image, and the need for a sophisticated tracker to filter them out.

Rudol and Doherty use an infrared camera in [58] to constrain the search for humans that are sitting or prone. First, they threshold the infrared imagery to isolate areas of human body heat, then projected those areas to a color camera. Next, they apply a cascaded ADABOOST classifier to those areas to detect humans. Unfortunately, this process requires mounting two cameras on the UAV. In this case, reliable detection of body heat requires expensive high-quality infra-red cameras.

Gaszczak et. al. also use an infrared camera in [22] for both human detection and vehicle heat-signature confirmation. Initial human detections were found in the infrared camera using cascaded ADABOOST, and then refined using a generative shape model. Initial vehicle detections on the other hand, were obtained in EO imagery using cascaded ADABOOST, and then verified by region-growing of hot spots in the IR camera.

Cascaded ADABOOST is also applied to EO imagery by Breckon et. al. [8] to detect vehicles. The initial detections obtained were then refined by using the height and field of view of the UAV to filter out detections which did not conform to proper vehicle sizes.

Skokalski and Breckon suggest a framework for detecting salient objects in color aerial imagery [64]. Given an input frame, they extracted nine feature images including a contrast map of the meanshift image and various normalized color channels. Next, they applied edge-detection and gradient operators to each, then combined them using AND and OR operators. The contrast map is weighted by the inverse of the probability of belonging to the global color distribution of the image. While this method shows impressive results without relying on meta-data, it requires color information for most of its features. Conversely, in grey-scale imagery with non-uniform background it would lose most of it's discriminative ability.

2.4 Shadow Detection

Since our static object detection method is based around detecting shadow casting objects, we feel that it is necessary to review some of the literature on shadow detection. There are generally two main types of shadow detection methods. One addresses with detecting and removing shadows from moving objects in video, while the other deals with detecting and removing shadows in single images.

2.4.1 Video

When attempting to detect and track moving objects in a video, both detection and tracking can have a number of issues, which we highlight in Figure 3.19. It is difficult to localize the object, since its shadow is part of the moving blob. The fact that the object and the shadow blobs are treated as one by the system makes the motion blobs more similar to each other; This makes it more difficult for the tracker to disambiguate them when their shadows overlap. A number of works have appeared on detecting and removing shadows from moving objects in a surveillance scenario for both people [45] [11] [6] [53] and vehicles [26] [44] [71] [27] [84] [39] (see [54] for a survey of these approaches).

Some works in this area are similar to [39] and [45], they extend the Gaussian Mixture Model based background subtraction to detect shadows in addition to the foreground object. For example, in [45] the key idea is that as objects move over the same surface, the appearance of foreground pixels belonging to objects will be different among the different objects; However, pixels belonging to shadows that are cast by those objects will have pixel values that are similar among the different objects. Therefore, the Gaussian Mixtures, which are constructed for pixels that have observed the shadows of moving objects, will develop a strong component that describes the shadows. Components that are identified as belonging to shadow are used to construct another Gaussian mixture model called the Gaussian Shadow model.

Other works, such as [11] [6] [26], first perform background subtraction to detect moving objects, and then apply a set of heuristics to the blobs in order to determine which parts of the blob contain shadow and which ones contain the object. In [11], Chang et.al. perform background subtraction and obtain blobs of moving people. The moving blobs include both the person and the shadow. In order to separate the person from the shadow, they determine the dominant orientation of the moving blob via the central moments of the blob, and separate the moving blob into person and shadow along the orientation of the blob based on contour heuristics. In order to refine the detected shadow, they construct a Gaussian probability density to represent the shadow. In [6], the heuristics are based on fitting ellipses to the motion blobs and analyzing the orientation of their dominant axes. In [26], vehicles are detected in a traffic surveillance scenario, then those detections are used to estimate the orientation of lanes in the scene. Finally, the parts of the moving blob that belong to shadow are removed based on their position relative to the lane dividers.

Such methods are useful in both ground and aerial surveillance scenarios. However, they require good statistical background subtraction, and the existence of moving objects in order for them to work. We, on the other hand, propose an integrated shadow and static object detection method where the two parts of the method can complement each other.

2.4.2 Single Frame

Detecting shadows in a single image is much more challenging than in video, and there are fewer methods for doing that (such as [20] [79] [50]). In theory, one of these methods can be used in place of the one suggested by us to find the shadows in the image. The methods are based on obtaining illumination invariant (shadow-less) images, and comparing edges between these and original images. The edges that belong to the shadow will appear in the original image, but not in the shadowless image, and therefore can be removed. In [20], the authors obtain the illumination invariant image by projecting pixel values in the 2d log-chromaticity space onto the direction orthogonal to the lighting direction. The authors of [79] also add l_2 normalization of the color channels. However, we found that the above methods perform poorly on our data (see Figure 3.15). We found that the illumination invariant images would remove parts of shadows, humans, and strong background gradients.

2.5 Recognition Using Few Examples

Researchers are aware of the limitations caused by an insufficient number of training examples. Hence, there are a number of works in various computer vision domains that address this problem. The problem has been explored in the domains of object recognition [66], concept detection [81], action recognition [82] [63] [42] [43] [4], object detection [17] [3], and handwritten character recognition [19] [36]. It is important to note that there is no standard definition of the problem, experimental setup, or metrics. Some authors define the problem as having few example for all of the classes of interest, while others allow multiple examples for some of the classes, and others are only interested in learning to separate the class of interest from an amorphous background class, which does not include other classes.

The authors of [66] utilize a nearest neighbor framework for object recognition, where each image is represented by a concatenation of six different features (RGB color histograms, SIFT, rgSIFT, PHOG, SURF, and local self-similarity histograms), which are concatenated and reduced to 500 dimensions using principle component analysis. The key idea behind their method of one-shot recognition is that prior to computing the distance between test examples and training examples, they project the feature vectors into a different space. The projection function is learned automatically through training using other classes, where the classes that are repeatedly separated into a series of one-shot recognition problems and the parameters of the projection are optimized using stochastic gradient descent. One issue with this framework, as reported by the authors, the performance of the method is tied to the number of classes that are available. The more classes that are utilized for learning the projection function, the better the method performs. In action recognition however, the datasets are generally smaller than in object recognition.

In [3], Aytar and Zisserman propose acquiring information about the one-shot class by utilizing a manually specified similar class that contains many examples. Then, the transfer learning is formulated in the the Support Vector Machine framework via Adaptive-SVM. The energy minimization function is almost the same as the one used in the standard SVM formulation, but with an additional term that includes the output of the similar class SVM. The authors show that that the adapted SVM achieves better performance than baseline SVM on PASCAL data. However, one issue is that the similar class is selected manually. While the similar class can be obvious given the problem of object detection and the features used by the authors, it may not be so obvious given some of the more abstract representations that are popular in the field, such as Bag of Words. Additionally, it appears the problem that is solved is one of object detection vs background, the authors did not report what would happen if the test images included examples from the similar class.

In [17], the problem is defined as determining whether a particular object is present in an image or not. Objects are represented by several parametric mixture models which include the appearance and shape information of interest points. The authors train these models for object classes that have many examples multiple times, in order to obtain a set of possible parameters of the probability distributions. Then, they define several probability distribution functions using the parameters of the probability distribution that represent the objects. Then, when learning the parameters of the object containing few examples, they use the hyper-parameters to guide the learning process.

Learning from few examples has also been attempted in template based approaches by building in additional generality, as was done in [82] and [63]. In [82], Yang et. al. do not match the template holistically, instead they break the template up into patches and optimally match each patch within a small neighborhood of the test window, and then sum the match scores across all of the patches. The purpose is to account for small variations in the actor's appearance and kinematics. The authors also attempt to use other datasets to learn weights for the patches, however it is not clear whether or not that actually helped, since the results they obtained are inconsistent. In [63], authors propose a novel technique based on locally steerable kernel, or LSK, for constructing a 3D template, which has some built-in robustness to small temporal and spatial variations in the performance of the action. The main issue with template-based methods is that not all actions are simple enough to be represented by templates.

2.6 Summary

The review of existing literature provided an overview of approaches for detecting, tracking, and recognizing the actions of objects of interest in both ground and aerial imagery. The main purpose was to provide an overview of the different directions in the field, and to focus on the limitations of these approaches when confronted by our problem domain. Also, we wanted to describe how some authors have attempted to address the challenges of small object size, fast object motion, high object density, strong shadows, and incomplete training sets.

In the next chapter, we describe in detail our proposed framework for detecting moving, as well as static objects in aerial imagery in the presence of parallax and strong shadow.

CHAPTER 3: DETECTION OF MOVING AND STATIC OBJECTS IN PRESENCE OF PARALLAX AND SHADOW

In this chapter we present a method for detecting moving and stationary humans and vehicles in imagery taken from a UAV in presence of strong shadow and parallax. First, we introduce a method for detecting moving objects in presence of fast camera motion and parallax. Then we describe our approach for detecting static objects based on constraining full-frame search by detecting shadow-casting out of plane candidates for humans and vehicles based on orientation of shadow and ground-plane normal in the image. We show how we can obtain this information from the me metadata available on the UAV platform. We also show that the candidate detection method can be extended to determine the orientation of the shadow in the image automatically. The chapter is organized as follows. In section 3.1 we describe how we compensate camera motion. Section 3.2 describes our approach for detecting densely moving objects. Section 3.3 describes the removal of false detections due to parallax. Section 3.4 describes how we can use the metadata to derive a set of world constraints for detecting humans and vehicles, and how we can project them into the image space. Section 3.5 focuses on how we use those constraints for human detection and classification, while section 3.6 does the same for vehicles. Section 3.7 describes how we can derive constraints directly from image information, in cases when metadata is unavailable. Section 3.8 presents results.

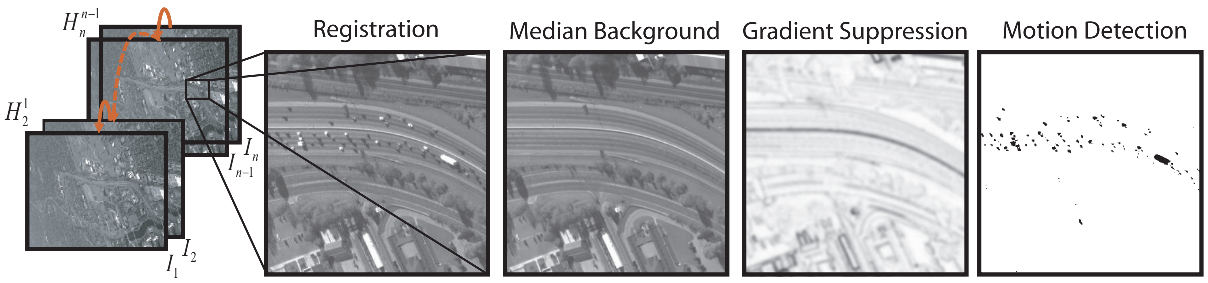


Figure 3.1: Different stages of our moving object detection pipeline. First, we remove global camera motion using point based registration, then we model the background using a 10 frame median image, perform background subtraction, and suppress false positives due to parallax and registration errors.

3.1 Registration

Prior to motion detection in aerial video, we remove global camera motion. The structured man-made environment in these scenes and large amount of detail yields itself nicely to a point-matching based registration algorithm. It is also much faster than direct registration method. We detect Harris corners in frames at time t as well as at time $t+1$. Then we compute SIFT descriptor around each point and match the points in frame t to points in frame $t+1$ using the descriptors. Finally, we robustly fit a homography H_t^{t+1} using RANSAC, that describes the transformation between top 200 matches. Once homographies between individual frames have been computed, we warp all the images to a common reference frame by concatenating the frame to frame homographies.

3.2 Motion Detection

After removing global camera motion, we detect local motion generated by objects moving in the scene.

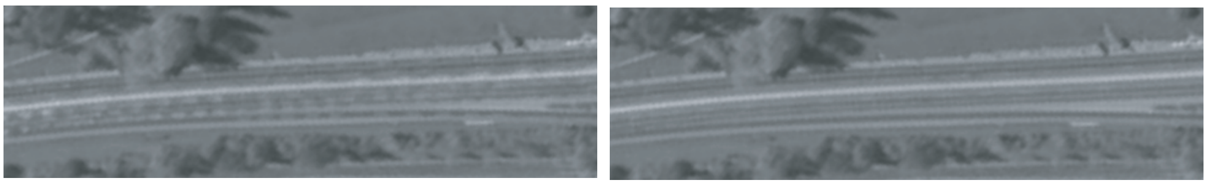


Figure 3.2: On the left, a background model obtained using mean, which has many ghosting artifacts from moving objects. On the right, a background model obtained using median with almost no ghosting artifacts.

To perform motion detection, we first need to model background, then moving objects can be considered as outliers with respect to the background. Probabilistic modeling of the background as in [65] has been popular for surveillance videos. However, we found these methods to be inapplicable to this data. In the parametric family of models, each pixel is modeled as either a single or a mixture of Gaussians. First, there is problem with initialization of background model. Since it is always that objects are moving in the scene, we do not have the luxury of object-free initialization period, not even a single frame. Additionally, since the cameras move, we need to build the background model in as few frames as possible, otherwise our active area becomes severely limited. Furthermore, high density of moving objects in the scene combined with low sampling rate makes the objects appear as outliers. These outliers can be seen as ghosting artifacts as shown in Figure 3.2. In the case of single Gaussian model, besides affecting the mean, the large number of outliers make the standard deviation high, allowing more outliers to become part of the model, which means many moving objects become part of the background model and are not detected.

A mixture of Gaussians makes background modelling even more complex by allowing each pixel to have multiple backgrounds. This is useful when background changes, such as in

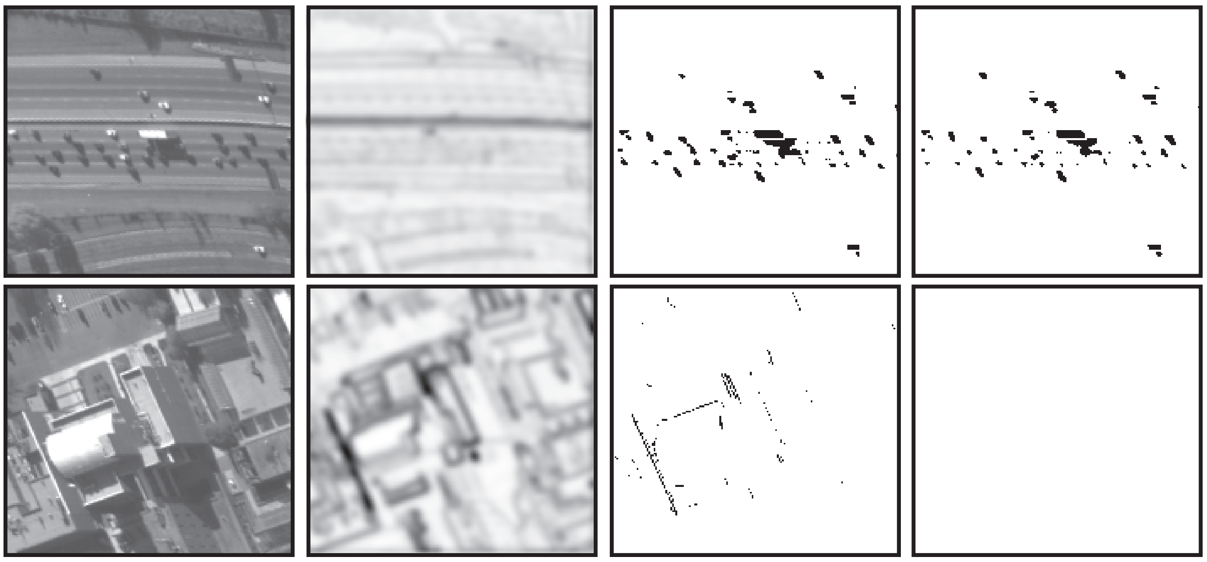


Figure 3.3: Left to right: A section of the original image, gradient of the median image, motion blobs prior to gradient suppression, motion blobs after gradient suppression. The bottom row shows an area of the image that has false motion detections due to parallax and registration errors. The top row shows a planar area of the image.

the case of a moving tree branch in surveillance video. This feature, however, does not alleviate any of the problems we highlighted above.

Therefore, we avoid probabilistic models in favor of simple median image filtering, which learns a background model with less artifacts using fewer frames (Figure 3.2). We found that 10 frame median image has fewer ghosting artifacts than mean image. To obtain a comparable mean image, it has to be computed over at least four times the number of frames which results in smaller field of view and makes false motion detections due to parallax and registration errors more prominent.

We perform motion detection in the following manner. For every 10 frames we compute a median background image B, next we obtain difference image i.e. $I_d = |I - B|$. Prior to thresholding the difference image, we perform gradient suppression. This is necessary to remove false motion detections due to parallax and registration errors.

3.3 Parallax Suppression

Since we fit a homography to describe the transformation between each pair of frames, we are essentially assuming a planar scene. This assumption does not hold for portions of the image that contain out of plane objects such as tall buildings. Pixels belonging to these objects are not aligned correctly between frames and hence appear to move even in aligned frames. Additionally due to large camera motion, there may be occasional errors in the alignment between the frames. An example of this is bottom row of Figure 3.3 where we show a small portion of an image containing a tall building (left). Due to parallax error, the building produces false motion detections along its edges (third image from the left). We suppress these by subtracting gradient of the median image ∇B (second column) from the difference image i.e. $I_d^r = I_d - \nabla B$. The top row shows a planar section of the scene and contains moving objects. As evident from Figure 3.3, this procedure successfully suppresses false motion detections due to parallax error without removing genuine moving objects. Also, the method has the advantage of suppressing false motion detections due to registration errors, since they too manifest along gradients. Note that above method works under an assumption that areas containing moving objects will not have parallax error which is valid for roads and highways.

3.4 Ground-Plane Normal and Shadow Constraints

Our detection method relies on a series of geometric constraints. One way of obtaining these constraints is to use the metadata provided by the UAV platform. In this section we

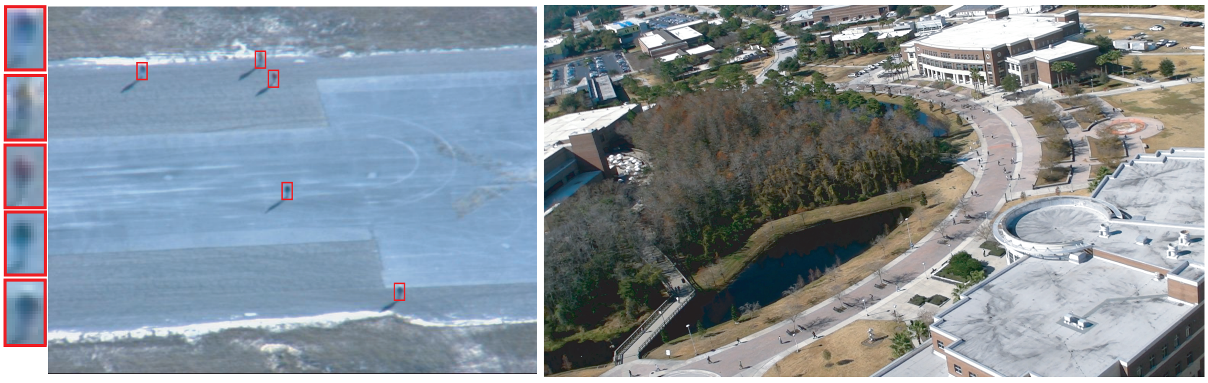


Figure 3.4: On the left, frames from some of the VIVID 3 sequences, also examples of humans, which are only around 24x14 pixels in size, making them difficult to distinguish from the background. On the right, a frame from one of our sequences we have taken which was shot from a balloon. still image shadow detection using techniques from [79]. Pixels belonging to humans, and large parts of background were incorrectly labelled as gradient which belongs to shadow.

describe how metadata of the UAV can be used to define a set of constraints in the world coordinate system, as well as how we can project those constraints into the image space.

3.4.1 Metadata

The imagery obtained from the UAV has the following metadata associated with most of the frames. It has a set of aircraft parameters *latitude*, *longitude*, *altitude*, which define the position of the aircraft in the world, as well as *pitch*, *yaw*, *roll* which define the orientation of the aircraft within the world. Metadata also contains a set of camera parameters *scan*, *elevation*, and *twist* which define the rotation of the camera with respect to the aircraft, as well as *focal length*, and *time*. We use this information to derive a set of world constraints and then project them into the original image.

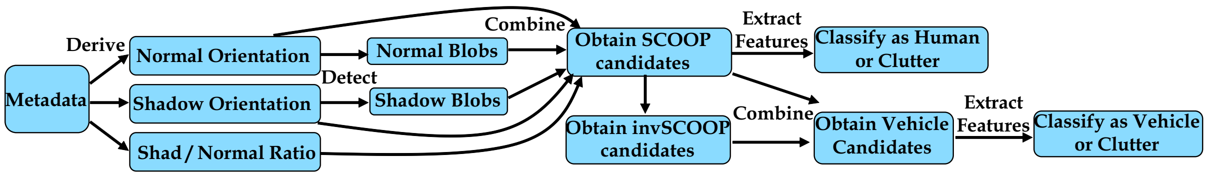


Figure 3.5: The overall pipeline of our system. First, we use metadata to derive geometric constraints. Second, we find normal and shadow blobs in the image. Third, using the geometric constraints, we combine blobs into SCOOP candidates. Fourth, using geometric constraints we combine blobs into inverted SCOOP candidates. Fifth, we combine SCOOP and inverted SCOOP into vehicle candidates. Finally, from each SCOOP candidate we extract wavelet features and classify it as either human or clutter. From every vehicle candidate we extract wavelet features and classify it as either human or clutter.

3.4.2 World Constraints

The shadow is generally considered to be a nuisance in object detection and surveillance scenarios. However, in the case of aerial human and vehicle detection, the shadow information augments the lack of visual information from the object, especially in the cases when the aerial camera is almost directly overhead. For human detection we define three world constraints:

- The person is standing upright.
- The person is casting a shadow.
- There is a geometric relationship between person's height and the length of their shadow (see Figure 3.6).

For vehicle detection, the constraints are:

- The vehicle is a box-like object.
- There is a geometric relationship between the boundaries of the vehicle and the boundaries of its shadow.

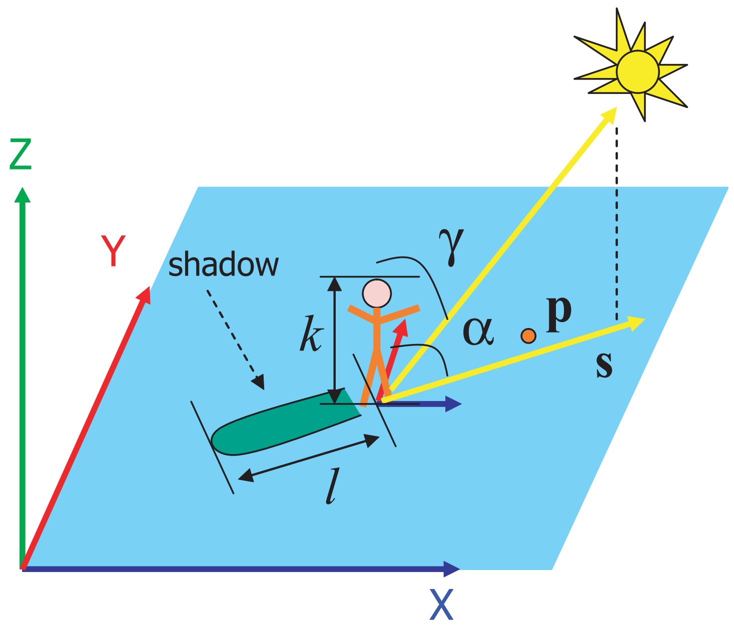


Figure 3.6: Our world. **X** corresponds to the East direction, **Y** to the North, **Z** to the vertical direction. Vector $\vec{\mathbf{S}}$ is pointing from an observer towards the sun along the ground. It is defined in terms of α - azimuth angle between the northern direction and the sun. The zenith angle γ is between the vertical direction and the sun. The height of a human is k and the length of the shadow is l.

Given the *latitude*, *longitude*, and *time* of day, we use the algorithm described in [56] to obtain the position of the sun relative to the observer on the ground. It is defined by the azimuth angle α (from the north direction) and the zenith angle γ (from the vertical direction). Assuming that the height of the person in the world is k we find the length of the shadow as

$$l = \frac{k}{\tan(\gamma - \pi/2)}, \tag{3.1}$$

where γ is the zenith angle of the sun. Using the azimuth angle α we find the ground plane projection of the vector pointing to the sun and scale it with the length of the shadow $\vec{\mathbf{S}} = \langle l\cos(\alpha), l\sin(\alpha), 0\rangle$.

3.4.3 Image Constraints

Before we can use our world constraints for human detection, we have to transform them from the world coordinates to the image coordinates. To do this we use the metadata to obtain the projective homography transformation that relates image coordinates to the ground plane coordinates. For an excellent review of the concepts used in this section see [24].

We start by converting the spherical *latitude* and *longitude* coordinates of the aircraft to the planar Universal Transverse Mercator coordinates of our world $X_w = \textit{east}$ and and $Y_w = \textit{north}$. Next, we construct a sensor model that transforms any image point $\mathbf{p}' = (x_i, y_i)$ to the corresponding world point $\mathbf{p} = (X_w, Y_w, Z_w)$. We do this by constructing the following sensor transform

$$\Pi_1 = T^a_{Zw} T^e_{Xw} T^n_{Yw} R^y_{Zw} R^p_{Xw} R^r_{Yw} R^s_{Za} R^e_{Xa} R^t_{Ya}. \tag{3.2}$$

Matrices T^a_{Zw}, T^e_{Xw}, and T^n_{Yw} are translations for aircraft position in the world: *altitude*, *east*, and *north* respectively. Matrices R^y_{Zw}, R^p_{Xw}, and R^r_{Yw} are rotations for the aircraft: yaw, pitch and roll respectively. Matrices R^s_{Za}, R^e_{Xa} and R^t_{Ya} are rotation transforms for camera: scan, elevation, and tilt, respectively.

We transform 2D image coordinates $\mathbf{p}' = (x_i, y_i)$ into 3D camera coordinates $\hat{\mathbf{p}}' = (x_i, y_i, -f)$, where f is the *focal length* of the camera. Next, we apply the sensor transform from equation 3.2 and ray trace to the ground plane (see Figure 3.7 **(a)**)

$$\mathbf{p} = RayTrace(\Pi_1 \hat{\mathbf{p}}'). \tag{3.3}$$

Ray tracing requires geometric information about the environment, such as the world height at each point. This information can be obtained from the digital elevation map of the

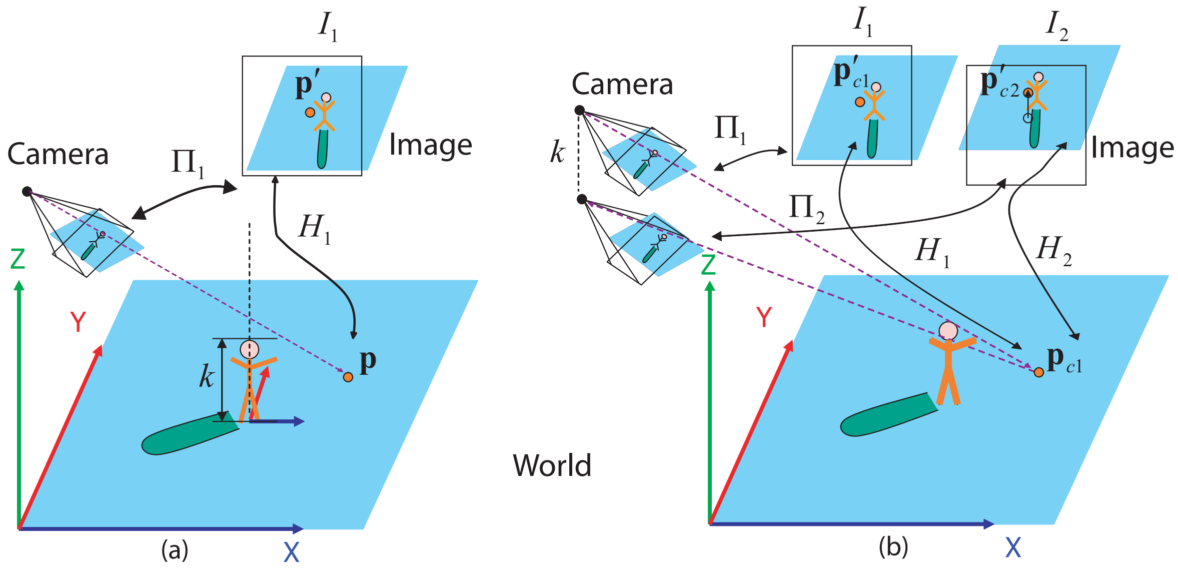


Figure 3.7: On the left, the sensor model Π_1 maps points in the camera coordinates into world coordinates. We place the image plane into the world, and ray trace through it to find the world coordinates of the image points (we project from the image plane to the ground plane). We compute a homography H_1 between the image points and their corresponding world coordinates on the groundplane. The panel on the right, illustrates how we obtain the projection of the ground plane normal in the original image. Using a lowered sensor model Π_2, we obtain another homography H_2, which maps the points in camera coordinates to a plane above the ground plane. Mapping a world point $\mathbf{p}_{c1}$ using H_1 and H_2 gives two image points $\mathbf{p}'_{c1}$ and $\mathbf{p}'_{c2}$, respectively. Vector from $\mathbf{p}'_{c1}$ to $\mathbf{p}'_{c2}$ is the projection of the normal vector.

area - DEM. In our case, we assume the scene to be planar and project the points to the ground plane at zero altitude $Z_w = 0$.

For any set of image points $\mathbf{p}' = (x_i, y_i)$, ray tracing gives a corresponding set of ground plane points $\mathbf{p} = (X_w, Y_w, 0)$. Since we are assuming that only one plane exists in the scene, we only need correspondences of four image corners. We then compute a homography, H_1, between the two sets of points, such that $\mathbf{p} = H_1\mathbf{p}'$. Homography, H_1, will orthorectify the original frame and align it with the North Direction (see Figure 3.8 **(a)**). Orthorectification removes perspective distortion from the image and allows for the measurement of world angles

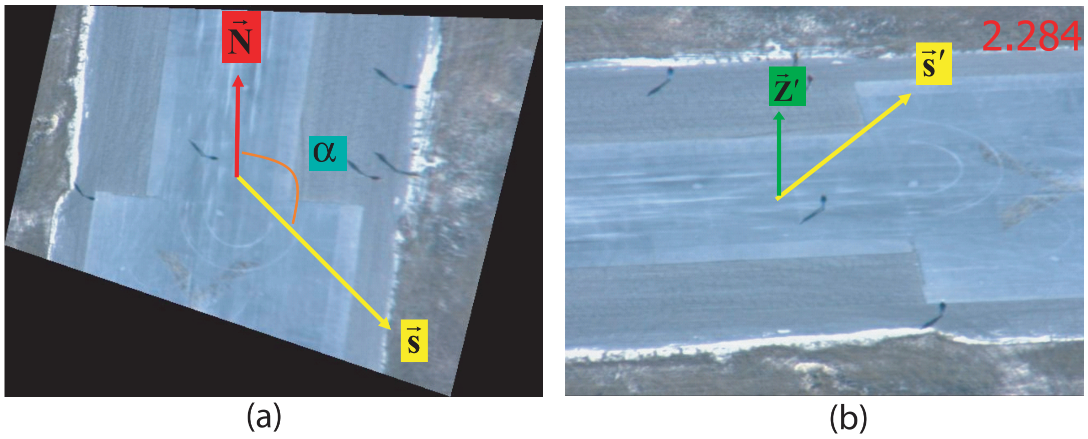


Figure 3.8: **(a)** shows an orthorectified frame from one of the sequences. The vertical direction is aligned with the world north direction $\vec{N}$. The sun vector $\vec{\mathbf{S}}$ is defined by the azimuth angle α between the north vector and the vector pointing to the sun. **(b)** is the original frame showing the projected sun vector $\vec{\mathbf{S}'}$, the projected normal vector $\vec{\mathbf{z}'}$, and the ratio between the projected normal and shadow lengths, 2.284.

in the image. We use the inverse of the homography, H_1^{-1}, to project the shadow vector defined in world coordinates into the image coordinates (see Figure 3.8 **(b)**).

$$\vec{\mathbf{S}'} = \vec{\mathbf{S}} H_1^{-1}. \tag{3.4}$$

Next, we obtain the projected ground plane normal (refer to Figure 3.7 **(b)**). We generate a second sensor model,

$$\Pi_2 = (T^a_{Zw} - [I|k])T^e_{Xw}T^n_{Yw}R^y_{Zw}R^p_{Xw}R^r_{Yw}R^s_{Za}R^e_{Xa}R^t_{Ya}, \tag{3.5}$$

where we lower the camera along the normal direction Z_w, by k, which is the assumed height of the person.

Using the above sensor model Π_2, we obtain a second homography H_2 using the same process that was used for obtaining H_1. We now have two homographies: H_1 maps the points from the image to the ground plane, and H_2 maps the points from the image to a virtual plane parallel to the ground plane that is exactly k units above the ground plane. We select the center point of the image $\mathbf{p}'_{c1} = (x_c, y_c)$, and obtain its ground plane coordinates $\mathbf{p}_{c1} = H_1\mathbf{p}'_c$. Then we map it back to the original image using H_2, $\mathbf{p}'_{c2} = H_2^{-1}\mathbf{p}_c$. The projected normal is then given by

$$\vec{\mathbf{Z}}' = p'_{c2} - p'_{c1}. \tag{3.6}$$

We compute the ratio between the projected shadow length and the projected height of the person as

$$\eta = \frac{|\vec{\mathbf{S}}'|}{|\vec{\mathbf{Z}}'|}. \tag{3.7}$$

3.5 Human Detection

Now that the world constraints have been projected into the image, we can avoid searching over the entire frame and instead search the space of potential object candidates. We define the search space as a set of pairs of blobs oriented in the direction of shadow and direction of normal. These combinations of normal and shadow blobs represent a set of shadow casting out of plane (SCOOP) candidates which can belong to humans, vehicles, or background. In the case of human detection, a single SCOOP candidate makes for a sufficient human candidate. In this section we describe how we use the constraints that we have projected into the image space to find blobs belonging to the ground plane normal and shadow, how to combine them into SCOOP candidates, and finally how to classify those candidates as human or clutter.

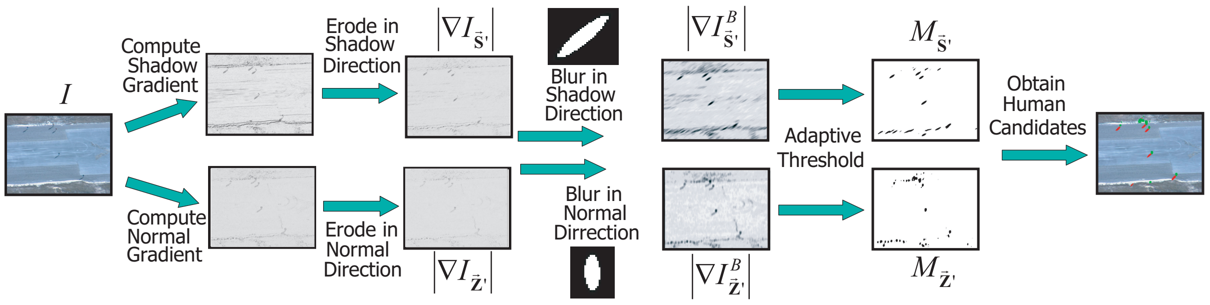


Figure 3.9: This figure illustrates the pipeline which utilizes image constraints to obtain an initial set of human and normal blobs, by applying a series of oriented filters to the original image.

3.5.1 Detecting Shadow and Normal Blobs

The first step of human detection is to detect blobs that potentially belong to out of plane objects, and blobs belonging to shadows. To do so, we use the image projection of the world constraints derived in the previous section: the projected orientation of the normal to the ground plane $\vec{\mathbf{Z}}'$, the projected orientation of the shadow $\vec{\mathbf{S}}'$, and the ratio between the projected height of the person, and projected shadow length η (see Figure 3.9). This image contains gradients oriented in many different directions, therefore we employ directed filters to enhance gradients oriented in the directions of interest while suppressing gradients oriented in other directions.

Given a frame I, we compute gradients oriented in the direction of the shadow by applying a 2D Gaussian derivative filter,

$$G(x,y) = \cos(\theta) 2xe^{-\frac{x^2+y^2}{\sigma^2}} + \sin(\theta) 2ye^{-\frac{x^2+y^2}{\sigma^2}}, \tag{3.8}$$

and take the absolute values of its responses. In the above equation θ is the angle between the vector of interest and the x axis. To further suppress gradients not oriented in the direction of the shadow vector we perform structural erosion along a line in the direction of the shadow orientation

$$|\nabla I_{\vec{\mathbf{S}}'}| = erode(\nabla I, \vec{\mathbf{S}'}). \tag{3.9}$$

We obtain $|\nabla I_{\vec{\mathbf{Z}}'}|$ using the same process. Next, we smooth the resulting gradient images with an elliptical averaging filter whose major axis is oriented along the direction of interest:

$$I^B_{\vec{\mathbf{S}}'} = |\nabla I_{\vec{\mathbf{S}}'}| * G_{\vec{\mathbf{S}}'}, \tag{3.10}$$

where $B_{\vec{\mathbf{S}}'}$ is an elliptical averaging filter, whose major axis is oriented along the shadow vector direction. This process fills in the blobs. We obtain $I^B_{\vec{\mathbf{Z}}'}$ using $G_{\vec{\mathbf{Z}}'}$. Next, we apply an adaptive threshold to each pixel to obtain shadow and normal blob maps

$$M_{\vec{\mathbf{S}}'} = \begin{cases} 1 & \text{if } I^B_{\vec{\mathbf{S}}'} > t \cdot mean(I^G_{\vec{\mathbf{S}}'}) \\ 0 & \text{otherwise.} \end{cases} \tag{3.11}$$

See Figure 3.10 for resulting blob maps overlaid on the original image. We obtain $M_{\vec{\mathbf{Z}}'}$ using the same method. From the binary blob maps we obtain a set of shadow and object blobs using connected components. Notice from Figure 3.10 that a number of false shadow and object blobs were initially detected. In the next section, we describe how to remove those false positives by combining normal and shadow blobs into SCOOP candidates.

3.5.2 *Exploiting Object Shadow Relationship to generate SCOOP candidates*

The initial application of the constraints does not take into account the relationship between the normal blobs and their shadows, hence generating many false positives. Our next step is to relate the shadow and normal blob maps. We obtain a set of SCOOP candidates and remove shadow-normal configurations that do not satisfy the image geometry which we

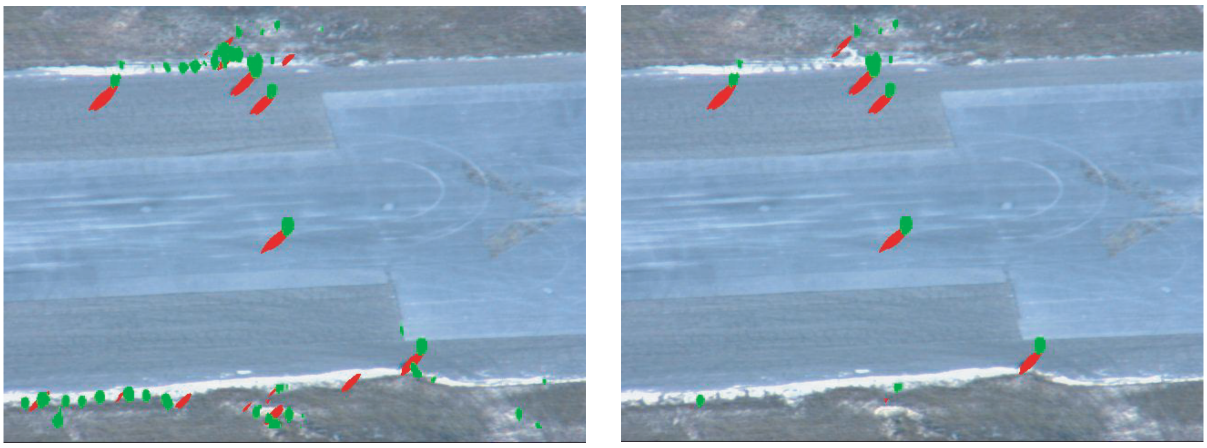


Figure 3.10: The left shows shadow blob map $M_{\vec{\mathbf{S}}'}$ (shown in red) and normal blob map $M_{\vec{\mathbf{Z}}'}$ (shown in green) overlayed on the original image. There are false detections at the bottom of the image. The right shows refined blob maps after each normal blob was related to its corresponding shadow blob. The false detections at the bottom are now gone.

derived from the metadata. We search every shadow blob, trying to pair it up with a potential object blob. If the shadow blob fails to match any object blobs, it is removed. If an object blob never gets assigned to a shadow blob it is also removed.

Given a shadow blob, $M^i_{\vec{\mathbf{S}}'}$, we search in an area around the blob for a potential object blob $M^j_{\vec{\mathbf{Z}}'}$. We allow for a single shadow blob to be assigned to multiple normal blobs, but not vice versa since the second case is rarely observed. The search area is determined by major axis lengths of $M^i_{\vec{\mathbf{S}}'}$ and $M^j_{\vec{\mathbf{Z}}'}$. For any object candidate blob, $M^j_{\vec{\mathbf{Z}}'}$ that falls within the search area, we ensure that it is in the proper geometric configuration relative to the shadow blob (see Figure 3.11) as follows. We make two line segments, l^i, and l^j, each defined by two points as follows $l^i = \{c_i, c_i + Q\vec{\mathbf{S}}'\}$ and $l^j = \{c_j, c_j - Q\vec{\mathbf{Z}}'\}$. Where c_i and c_j are centroids of shadow and object candidate blobs, respectively, and Q is a large number. If the two line segments intersect, then the two blobs exhibit correct object shadow configuration.

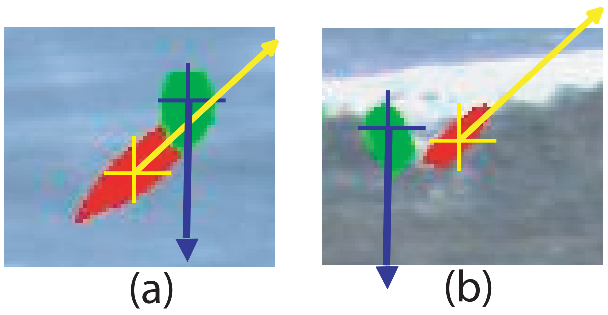


Figure 3.11: **(a)** A valid configuration of normal and shadow blobs results in an intersection of the rays, and is kept as a SCOOP candidate. **(b)** An invalid configuration of blobs results in the divergence of the rays, and is removed from the set of SCOOP candidates.

We also check to see if the lengths of the major axes of $M^i_{\vec{\mathbf{S}}'}$ and $M^j_{\vec{\mathbf{Z}}'}$ conform to the projected ratio constraint η. If they do then we accept the configuration.

Depending on the orientation of the camera in the scene, it is possible for the person and shadow gradients to have the same orientation. In that case the shadow and normal blobs will merge. The amount of merging depends on the similarity of orientations $\vec{\mathbf{S}}'$ and $\vec{\mathbf{Z}}'$. Hence, we accept the shadow object pair if

$$\frac{M^i_{\vec{\mathbf{S}}'} \cap M^j_{\vec{\mathbf{Z}}'}}{M^i_{\vec{\mathbf{S}}'} \cup M^j_{\vec{\mathbf{Z}}'}} > q(1 - abs(\vec{\mathbf{S}}' \cdot \vec{\mathbf{Z}}')), \tag{3.12}$$

where q was determined empirically. For these cases, the centroid of the object candidate blob is not on the person. Therefore, for these cases we perform localization where we obtain a new centroid by moving along the shadow vector $\vec{\mathbf{S}}'$ as follows

$$\tilde{c} = c + \frac{m}{2}(1 - \frac{1}{\eta})\frac{\vec{\mathbf{S}}'}{\|\vec{\mathbf{S}}'\|}, \tag{3.13}$$

where m is the length of the major axis of shadow blob $M^i_{\vec{\mathbf{S}}'}$.

3.5.3 Obtaining Human Candidates

The outlined procedures generate the final set of SCOOP candidates $\mathbf{K}_{\vec{\mathbf{S}}'}^{\vec{\mathbf{Z}}'} = \{\mathbf{k}_{\vec{\mathbf{S}}'1}^{\vec{\mathbf{Z}}'}, ..., \mathbf{k}_{\vec{\mathbf{S}}'n}^{\vec{\mathbf{Z}}'}\}$, where each candidate $\mathbf{k}_{\vec{\mathbf{S}}'}^{\vec{\mathbf{Z}}'} = \{M_{\vec{\mathbf{S}}'}^{i}, M_{\vec{\mathbf{Z}}'}^{j}\}$ is a pair of normal and shadow blobs that were detected at normal orientation $\vec{\mathbf{Z}}'$ and shadow orientation $\vec{\mathbf{S}}'$. Since a single SCOOP candidate is sufficient to capture a human in low resolution aerial video, $\mathbf{K}_{\vec{\mathbf{S}}'}^{\vec{\mathbf{Z}}'}$ is the set of human candidates where we classify each normal blob $M_{\vec{\mathbf{Z}}'}$ as human or clutter.

3.5.4 Classifying Human Candidates

The final step of human detection is to classify each SCOOP candidate $\mathbf{k}_{\vec{\mathbf{S}}'}^{\vec{\mathbf{Z}}'}$ as either human or clutter. For this purpose, we compute the centroid of the normal blob $M_{\vec{\mathbf{Z}}'}^{i}$ of each remaining SCOOP candidate, and extract a $w \times h$ chip around that centroid. We then extract wavelet features from each chip and apply a Support Vector Machine (SVM) classifier (see Figure 3.12). We use the Daubechies 2 wavelet filter, where the low-pass (L) and high-pass (H) filters for a 1-D signal are defined as

$$\phi_1(x) = \sqrt{2}\sum_{k=0}^{3} c_k \phi_0(2x - k), \tag{3.14}$$

$$\psi_1(x) = \sqrt{2}\sum_{k=0}^{3} (-1)^{k+1} c_{3-k} \phi_0(2x - k), \tag{3.15}$$

where ϕ_0 is either row or column of the original image and $c = (\frac{(1+\sqrt(3))}{4\sqrt(2)}, \frac{(3+\sqrt(3))}{4\sqrt(2)}, \frac{(3-\sqrt(3))}{4\sqrt(2)}, \frac{(1-\sqrt(3))}{4\sqrt(2)})$, are the Daubechies 2 wavelet coefficients. In the case 2D signals, such as images, the 1D filters are first applied along x, and then y directions. This produces four outputs: LL, LH, HL, HH. Where LL is a scaled version of the original image and LH, HL, and HH, correspond to gradient like features along horizontal, vertical and diagonal directions. We used only one level, since adding more did not improve the performance. We vectorize the resulting outputs, normalize their values to be in the $[0, 1]$ range, and concatenate them into a single feature-vector. We train a Support Vector Machine [10] on the resulting feature set using the RBF kernel. We use 2099 positive and 2217 negative examples $w \times h$ pixels in size.

Note that if focal length data is available, then the chip size could be selected automatically based on the magnitude and orientation of the projected normal $|\vec{\mathbf{Z}}'|$. Additionally, if perspective distortion in the imagery is fairly strong, we would have to assume different sizes of humans and shadows for different regions of the image, which would require a minor change in the geometric part of the method. The change would include computing multiple shadow and normal vector *magnitudes* for different regions of the image. Since there is little perspective distortion in VIVID, no drastic zoom changes within a video, and the focal length information provided is not correct, we kept $w \times h$ constant over the entire video. We selected $w \times h$ to be 24×14 equal to the size of images in the training set.

3.6 Vehicle Detection

Since vehicles are larger, and more complex objects than humans (at typical aerial surveillance resolutions), a single SCOOP candidate is too simple to serve as a vehicle candidate. While the vehicle does generate a SCOOP (see Figure 3.13 **(b)**), the normal blob of the SCOOP (shown in green) captures only a small part of the vehicle without giving a clear idea of the size of the vehicle or the location of its centroid. This makes it difficult to localize the vehicle for actual classification. Additionally, if our goal is to minimize the number of classifications that we want to perform, then treating every SCOOP candidate as a potential vehicle

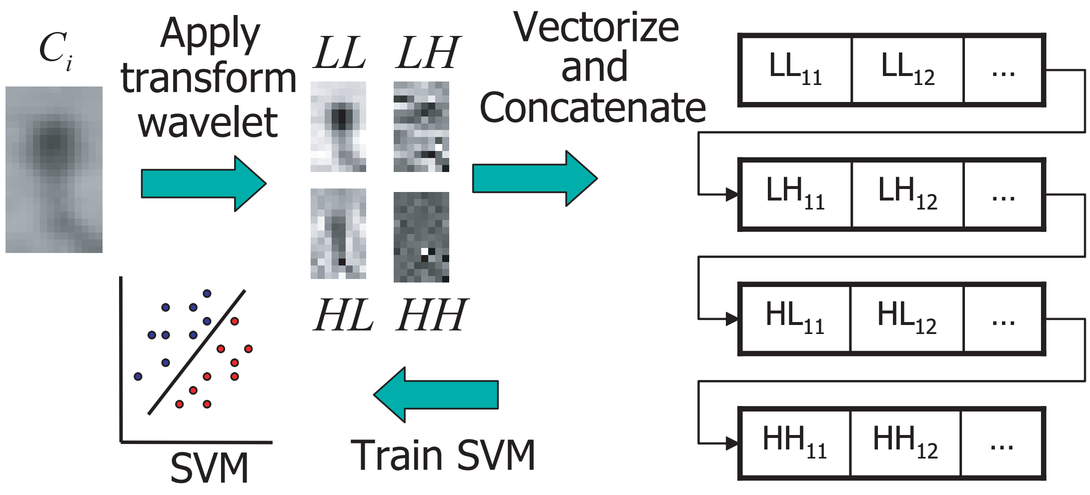


Figure 3.12: Object candidate classification pipeline. Four wavelet filters (LL, LH, HL, HH) produce a scaled version of original image, as well as gradient like features in horizontal, vertical, and diagonal directions. The resulting outputs are vectorized, normalized, and concatenated to form a feature vector. These feature vectors are classified using SVM.

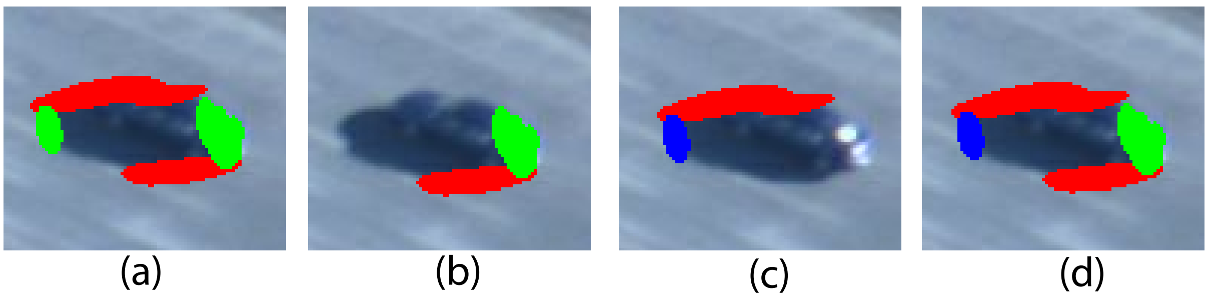


Figure 3.13: **(a)** All shadow and normal blobs obtained using method described in section 3.5.1. **(b)** A SCOOP candidate detected using the method described in section 3.5.2 at normal orientation $\vec{\mathbf{Z}}'$ and shadow orientation $\vec{\mathbf{S}}'$. **(c)** An inverted SCOOP candidate that was detected at normal orientation $-\vec{\mathbf{Z}}'$ and shadow orientation $-\vec{\mathbf{S}}'$. **(d)** A car candidate assembled from a SCOOP and an inverted-SCOOP candidates.

is less than optimal since SCOOPs can be generated by humans, poles, or simply background gradients. As can be seen in Figure 3.13 **(a)**, a vehicle will have at least two shadow and two normal blobs associated with it. In this section we describe how we can use multiple SCOOP candidates to obtain vehicle candidates.

3.6.1 Obtaining Vehicle Candidates

In addition to obtaining a set of SCOOP candidates $\mathbf{K}_{\vec{\mathbf{S}}'}^{\vec{\mathbf{Z}}'}$ as described in section 3.5.2, we use the exact same process in obtaining a set of inverse-SCOOP candidates $\mathbf{K}_{-\vec{\mathbf{S}}'}^{-\vec{\mathbf{Z}}'}$, where we negate the direction of the normal $\vec{\mathbf{Z}}'$ and the direction of the shadow $\vec{\mathbf{S}}'$. Next, we combine the regular and inverted-SCOOP candidates.

For every scoop candidate $\mathbf{k}_{\vec{\mathbf{S}}'}^{\vec{\mathbf{Z}}'} = \{M_{\vec{\mathbf{S}}'}^{i}, M_{\vec{\mathbf{Z}}'}^{j}\}$, we search for the closest inverted SCOOP candidate $\mathbf{k}_{-\vec{\mathbf{S}}'}^{-\vec{\mathbf{Z}}'} = \{M^{k}_{-\vec{\mathbf{S}}'}, M^{k}_{-\vec{\mathbf{Z}}'}\}$. The distance between the candidates is defined as the Euclidian distance between the points e_1 and e_2 (see Figure 3.14 **(a)**). Where e_1 is a point on shadow blob $M_{\vec{\mathbf{S}}'}^{i}$, of SCOOP candidate $\mathbf{k}_{\vec{\mathbf{S}}'}^{\vec{\mathbf{Z}}'}$ that is closest to the corresponding normal blob $M_{\vec{\mathbf{Z}}'}^{j}$, and e_2 is a point on shadow blob $M^{k}_{-\vec{\mathbf{S}}'}$ of inverse-SCOOP candidate $\mathbf{k}_{-\vec{\mathbf{S}}'}^{-\vec{\mathbf{Z}}'}$, that is furthest from its corresponding normal blob $M^{l}_{-\vec{\mathbf{Z}}'}$. Since we want the SCOOP candidates to enclose the car, we impose an additional constraint that the orientation of the vector between the centroids of the shadow blobs $M_{\vec{\mathbf{Z}}'}^{j}$ and $M^{k}_{-\vec{\mathbf{S}}'}$ must be at least 30° different from the orientation of the shadow blobs. Since this gives better localization and excludes a large part of the shadow, a vehicle candidate is represented by a bounding box that encompasses both the SCOOP and the centroid of the shadow blob of the inverse SCOOP.

A single vehicle may have multiple SCOOP pairs detected on it. Distractions like specular reflections or decals can create multiple SCOOP and inverse-SCOOP candidates for a single vehicle. Therefore, as a final processing step we merge bounding boxes that have more that 50% overlap given by the following formula

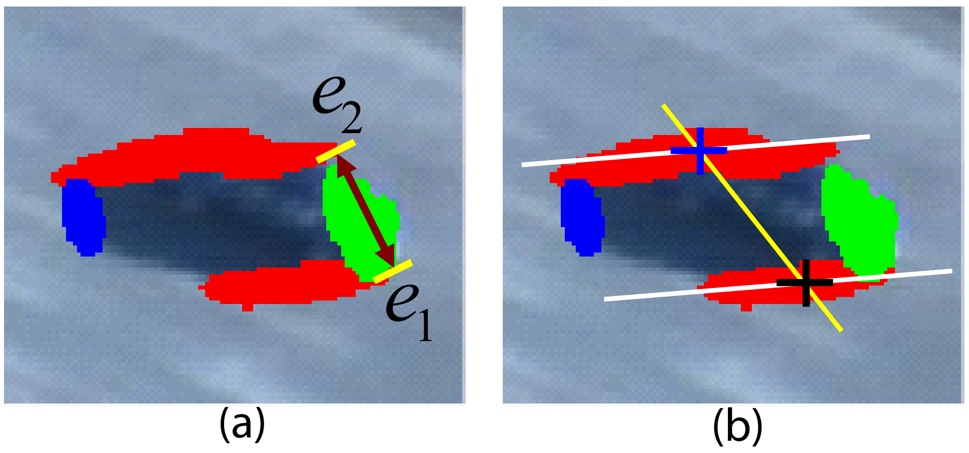


Figure 3.14: **(a)** Shows the distance computation between the SCOOP and inverted SCOOP candidates. The distance is computed between points e_1 and e_2. **(b)** Shows the orientation constraint between the SCOOP and inverted SCOOP candidates which must be satisfied for their configuration to be considered a valid car candidate.

$$\frac{A \cap B}{min(A, B)}, \tag{3.16}$$

where A and B are areas of two vehicle candidate bounding boxes. Dividing by the minimum of the two areas makes it easier to merge boxes. In cases where a small bounding box is enclosed within a larger one, it will be merged. We perform the merge procedure recursively until no additional bounding boxes can be merged.

3.6.2 Classifying Vehicle Candidates

As with humans, the final stage of vehicle detection is classifying each vehicle candidate as either vehicle or clutter. Since vehicle candidates encompass the entire object, we simply extract the region within the candidate's bounding box and resize it to 40×40. Next we extract Daubechies 2 wavelet coefficients and classify them using SVM. The vehicle training set consisted of 1900 positive and 1889 negative examples. We included vehicles at different orientations in the positive set.

3.7 Constraints without Metadata

Having all of the metadata provides a set of strict constraints for a variety of camera angles and times of day. However, there may be cases when the metadata is either unavailable or is incorrect. In such cases it is acceptable to sacrifice generality and computation time to obtain a looser set of constraints that still perform well. If we assume that humans are vertical in the image and ignore the ratio between the size of humans and their shadows, we can determine the orientation of the shadow in the image by exploiting the relationship between out of plane objects and their their shadows.

When SCOOP objects are present in the scene the shadows they cast will be at similar orientations. Therefore, SCOOP objects will be consistently detected in the shadow orientation range of about -10° to $+10^\circ$ of the actual orientation. If we assume that the scene is mostly planar, then the orientation range will be consistent across all SCOOP objects in the scene. We apply the SCOOP detector at all orientations of the shadow, then search for the longest range of orientations consistent across multiple SCOOP objects. This corresponds to the longest common consecutive subsequence (LCCS) among orientation ranges of all potential SCOOP objects detected in the scene. Hence we need at least two SCOOP objects on the plane. If the scene is not planar, shadow orientation ranges for some SCOOP objects will exhibit a shift. This shift will simply shorten the length of the LCCS, though a reasonable orientation allowing for SCOOP detection will still be found. Multiple planes, each with drastically different

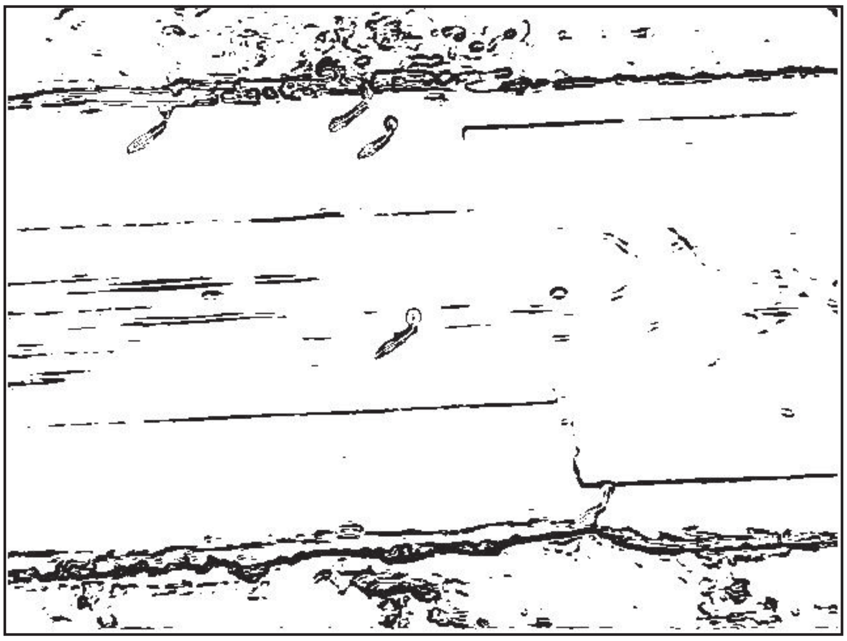


Figure 3.15: Still image shadow detection using techniques from [79]. The dark edges are the gradients labelled by the method as belonging to shadow. The gradients belonging to humans and large parts of the background were incorrectly labelled as gradients which belong to shadow.

orientations, will correspond to multiple LCCSs which can be found if two or more SCOOP objects are present on each plane. The details of this method are as follows.

We quantize the search space of shadow angle θ between 0 and 2π in increments of d (we used $\pi/36$ in our experiments). Keeping the normal orientation fixed, and ignoring shadow-to-normal ratio, we find all human candidates in image I for every orientation θ using techniques described in sections 3.5 and 3.5.2 (see Figure 3.16). We track the candidates across different θ. Similar angles θ will detect the same human candidates. Therefore, each human candidate C_i has a set Θ_i for which it was detected, and a set O_i which is a binary vector, where each element corresponds to whether the shadow and human blobs overlapped. The set of orientations for which it was detected due to overlap is Θ_i^o, and the set of orientations for which it was detected without overlap is $\Theta_i^{\bar{o}}$ (see Figure 3.16). We remove any candidate which

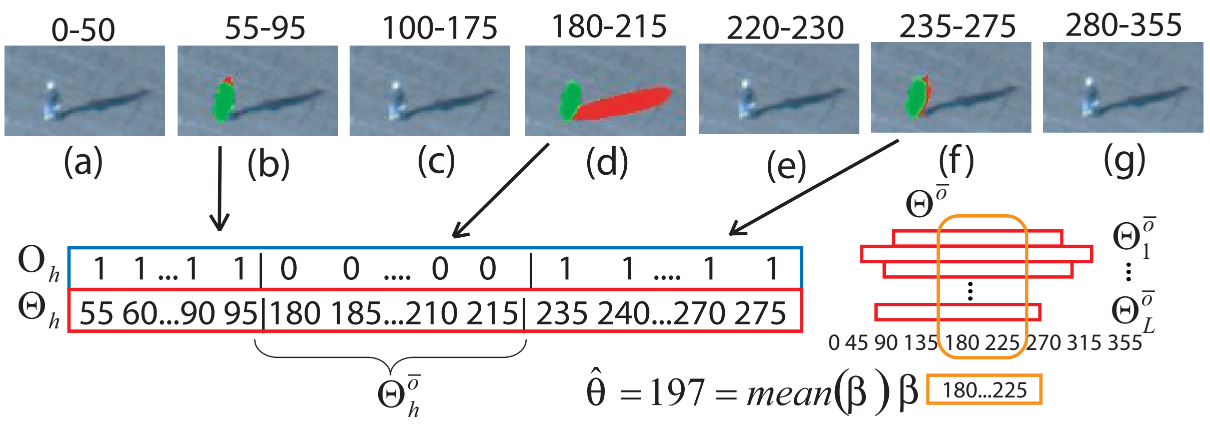


Figure 3.16: Shows our method for finding optimal shadow orientation for a given image in the absence of metadata. The top row shows human candidate responses obtained for different shadow orientations. A human candidate is then described by a vector of orientations for which it was detected and a binary overlap vector. Optimal orientation $\hat{\theta}$ is the average of the longest common consecutive non-overlapping subsequence of orientations among all human candidates.

has been detected over less than p orientations, since a human is always detected as a candidate if shadow and normal orientations are similar and the resulting blobs overlap according to equation 3.12 (as in Figure 3.16 (b) & (f)). Here p depends on quantization; we found that it should encompass at least $70°$.

We now find the optimal shadow orientation $\hat{\theta}$ by treating each $\Theta_i^{\bar{o}}$ as a sequence and then finding the longest common consecutive subsequence β among all $\Theta^{\bar{o}}$. Subsequence β must span at least $20°$ but no more than $40°$. Finally, the optimal orientation $\hat{\theta} = mean(\beta)$. If we cannot find such a subsequence then there are either no shadows, or the orientation of the shadow is the same as the orientation of the normal, so we set $\hat{\theta}$ to our assumed normal. Figure 3.17 shows an example frame for which human candidates were detected using the automatically estimated shadow orientation. There is a $10°$ difference between the estimated orientation and the orientation derived from the metadata. This is the same frame as in Figure

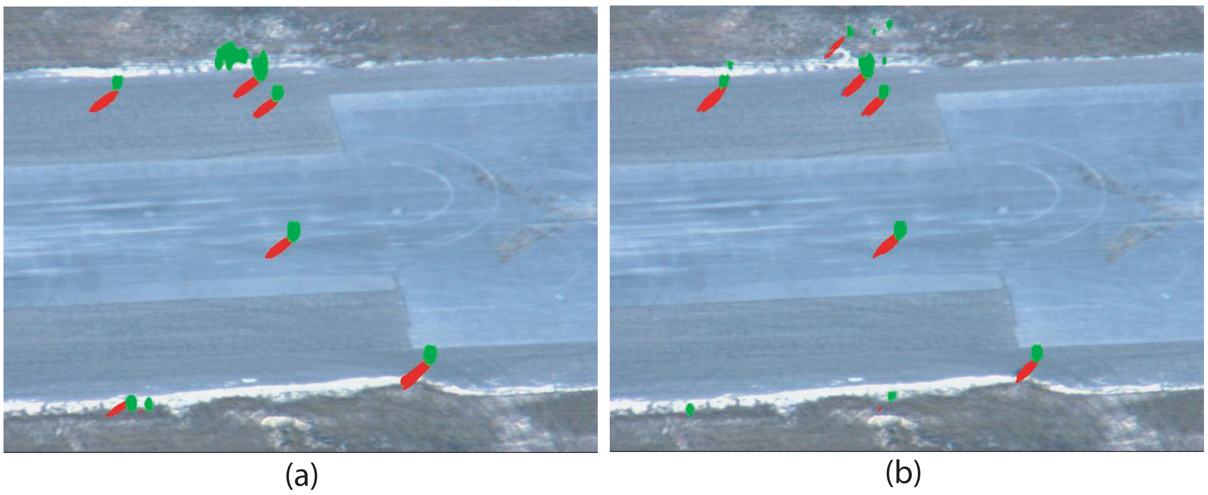


Figure 3.17: **(a)** The refined human candidate blobs for an automatically estimated shadow orientation of $35°$ without metadata. **(b)** The refined blobs that were obtained using the metadata-derived orientation value of $46.7°$.

3.10, qualitative examination of the shadow blobs indicates that the estimated orientation is more accurate than the one derived from the metadata, however the computation time of obtaining it is much larger. In practice, the angle can be estimated in the initial frame and then predicted in subsequent frames using a Kalman filter.

3.8 Results

In this section we present both quantitative and qualitative results of human and vehicle detection on the VIVID and CLIF datasets. These results are compared against motion constrained detection, Harris corner constrained detection, and an unconstrained full frame search.

3.8.1 Human Detection

We evaluated our detection methods on three sequences from the DARPA VIVID3 dataset of 640x480 resolution and compared the detection against manually obtained groundtruth.

Table 3.1: This table provides details on the VIVID sequences that were used for quantitative evaluation of the human detection methods.

	Sequence1	**Sequence2**	**Sequence3**
Frames	1191	1006	823
Total People	4892	2000	3098

The Table 3.1 shows the number of frames and the number of people in each sequence. We removed the frames where people congregated into groups. We used the following evaluation criteria $Recall$ (True detection rate) vs. False Positives Per Frame (FPPF). Recall is defined as $\frac{TP}{TP+FN}$, where FN is number of false negatives in the frame and TP is the number of true positives. To evaluate the accuracy of the geometry-based human candidate detector method, we require the centroid of the object candidate blob to be within w pixels of the centroid blob, where w is 15. We did not use the PASCAL measure of 50% bounding box overlap, since in our dataset the humans are much smaller and make up a smaller percentage of the scene. In the INRIA set introduced in [14], an individual human makes up 6% of the image, in our case the human makes up about 0.1%. Under these circumstances, small localization errors result in a large area overlap difference. Hence, the centroid distance measure is more meaningful for aerial data.

Figure 3.18 compares ROC curves for the following methods: our geometry based method with and without the use of object-shadow relationship refinement and centroid localization, our geometry method augmented with temporal information, conventional full frame detection method (we used HOG detection binaries provided by the authors), and standard motion detection pipeline of registration, detection, and tracking.

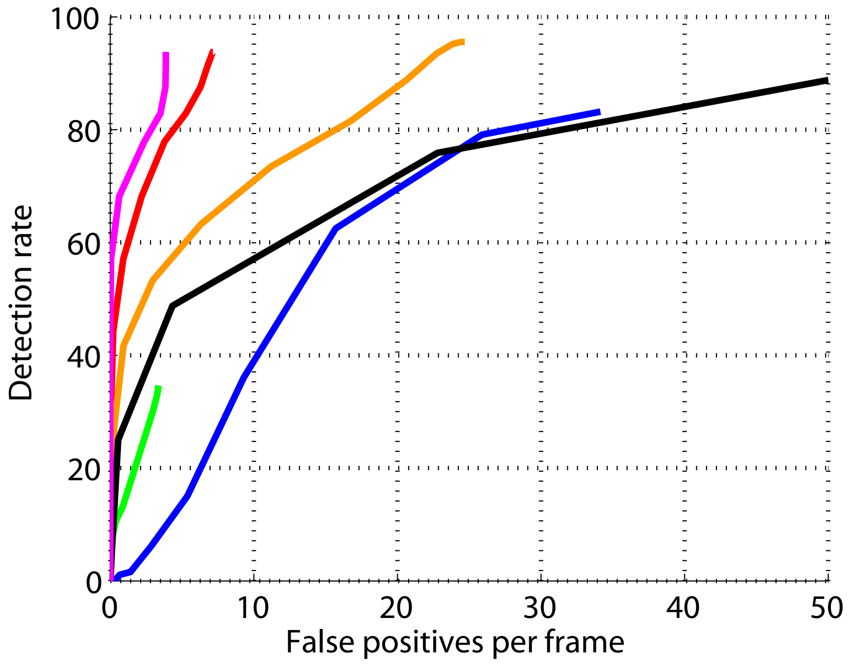


Figure 3.18: SVM confidence ROC curves. Our geometry method based on classifying SCOOP candidates is shown in red. The orange curves reflect our geometry based method without the use of object-shadow relationship refinement or centroid localization. The Magenta curves show our geometry based method augmented with temporal information. A standard full frame detector (HOG) is shown in blue. Green shows results obtained from classifying blobs obtained through registration, motion detection, and tracking, similar to [77]. The **black** curves are for our modified implementation of [48], which uses Harris corner tracks.

Figure 3.21 shows qualitative detection results. Conventional full frame detection is not only time consuming (our MATLAB implementation takes several hours per 640x480 frame), but it also generates many false positives. By contrast, preprocessing the image using the proposed geometric constraints to obtain human candidates is not only much faster (0.72 seconds per frame), but gives far better results. Geometric constraints with the use of shadow based refinement and centroid localization provide the best performance. However, even without these additional steps the geometric constraint based only on the projection of the normal still gives superior results to full frame and motion constrained detection.

Motion based detection suffers from problems discussed in Section 2.4.1 and shown in Figure 3.19. Which is why the green ROC curve in Figure 3.18 is very short. We implemented a part of the method found in [48], where instead of using the OT-Mach filter, we used our wavelet SVM combination for classification. This ROC curve is shown in black. We suspect that the poor performance is due to the large number of initial candidates, since multiple corners are likely to occur on man-made background objects, their shadows, and airfield markings.

By contrast, when proper initialization and localization is provided, temporal information is a convenient way of improving performance. This is illustrated by the magenta curve in Figure 3.18. In this case, detections obtained by refined geometric constraints (red curve) are tracked in a homography-constrained global coordinate system. Objects that persisted for less than 5 frames are discarded. Classification is performed using wavelets and SVM: if 20% of the track is classified as human, then the entire track is labelled as human. The above technique further suppresses false positives, however the maximum detection rate is slightly reduced since we probably removed incidents of short human tracks.

3.8.2 Vehicle Detection

Quantitative evaluation of vehicle detection was performed on the entire frame range of the same VIVID3 sequences used in the quantitative evaluation of human detection. Table 3.2 shows the number of frames and vehicles contained within each sequence. In order to determine true detections, we used the 33% bounding box overlap criteria from [34]. Figure 3.20 shows the Recall vs FPPF ROC curves of the same 6 detection methods described in 3.8.1, with a few slight differences specific to vehicle detection. In the geometry method enhanced

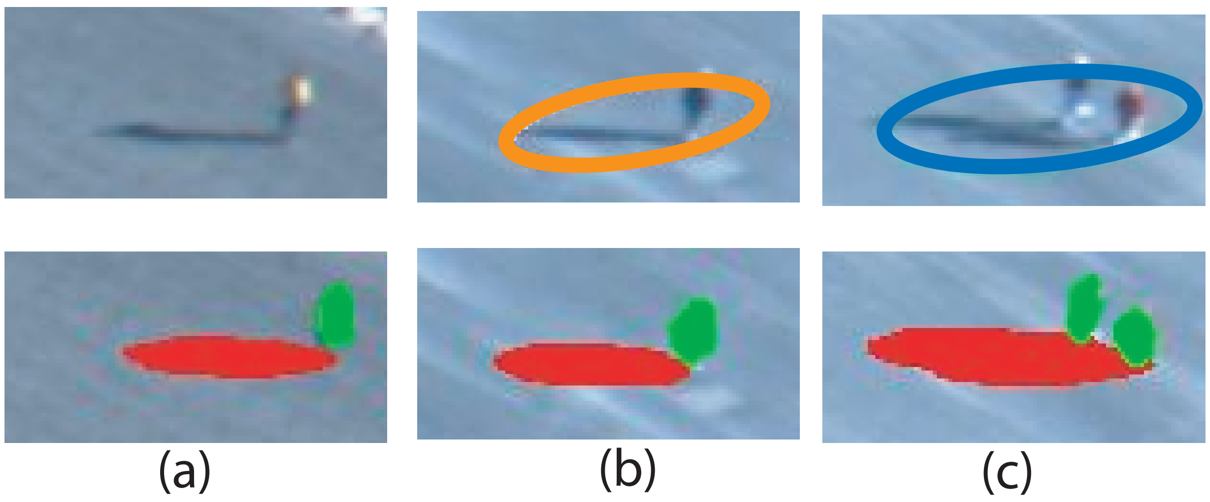


Figure 3.19: Qualitative comparison of motion detection (top row) to our geometry based method (bottom row). (a) The human is stationary and was not detected by the motion detector. (b) The moving blob includes a shadow, the centroid of the blob is not on the person. (c) Two moving blobs were merged by the tracker because of shadow overlap, the centroid is not on either person. In contrast our method correctly detected and localized the human candidate (green).

with temporal information (the magenta curve), we removed tracks of lengths less than ten. In the case of geometry method without using the SCOOP concept (the orange curve), we represented vehicle candidates as pairs of normal blobs instead of classifying each blob individually. In the Harris corner constrained method (the black curve), we did not track and classify individual Harris corners. Because a single vehicle will have multiple corners belonging to it, we clustered the locations of the corners using mean shift then tracked and classified the resulting clusters.

At this resolution, the vehicles are much larger than humans and have a sufficient amount of interesting visible features. Therefore, as can be seen in Figure 3.20, the performance of all detection methods is much better than in the human case. Another thing to note, is that the relative performance of some of the methods is now different. Full frame detection

Table 3.2: This table provides the details for the 3 VIVID sequences which were used for quantitative vehicle detection evaluation

	Sequence1	**Sequence2**	**Sequence3**
Frames	1812	1785	1813
Total Vehicles	1719	1742	2019

receives very large performance gain, since the classifier can construct a good object model at this resolution. The gains from constraining the search using SCOOP candidates is moderate, though the speed advantage is still there.

Another drastic difference is the performance of motion detection based method (green curve). It is actually better than the geometry method without utilizing temporal information (red curve). This is because the camera observes the scene persistently allowing for a good background model to be constructed. Additionally, unlike humans, the vehicles are always moving fairly quickly in these sequences. This allows the background subtraction to detect them with ease. Another issue is that unlike in the case of humans, the shadows cast by the vehicle do not severely distort the motion blob creating localization problems at the classification stage and the ground-truth comparison stage.

Detection based on grouping of normal blobs without utilizing the SCOOP concept (orange curves) is not very meaningful. This is because it essentially represents the vehicle as two neighboring parallel gradients, ignoring the box-like nature of the object.

Finally, representing shadow casting vehicles as clusters of Harris corners is rather challenging, since multiple Harris corners are detected on the vehicle and its shadow. These corners have to be clustered to estimate the location of the vehicle. The relative location and quantity of these corners differs with changes in orientation of the vehicle, and corners from background and neighboring objects can become part of the cluster as the vehicle moves through the scene. This, of course, causes localization problems, not unlike those of motion detection in the human case.

Qualitative evaluation (see Figure 3.26) was performed on segments of the CLIF 2007 dataset. Here the resolution is smaller than in VIVID and there are more confusers. In this case, full frame detection once again starts to perform poorly and there is a clear gain from using SCOOP candidates to constrain the search. In the case of CLIF, the camera is very close to nadir, making the normals very short and in the case of vehicles almost non existent. However, our method can still find the candidates. Additionally, if the vehicle is oriented with the orientation of the shadow, the actual presence of the shadow is not even necessary to be able to detect it as a candidate, since the primitive blobs will be detected on the sides of the vehicle. Whether a vehicle can be correctly detected as a candidate while it is at a $45°$ angle relative to shadow's orientation depends on whether the resolution is high enough to be able to detect the primitive shadow blobs on the corner of the vehicle.

3.9 Summary

We proposed a novel method for detecting humans and vehicles in aerial imagery. This method works on a single image and is is based on constraining the search space of the image by detecting Shadow Casting Out Of Plane (SCOOP) object candidates. Our method takes advantage of the metadata information provided by the UAV platform to derive a set of geometric constraints, and to project them into the imagery. In cases where metadata is not available,

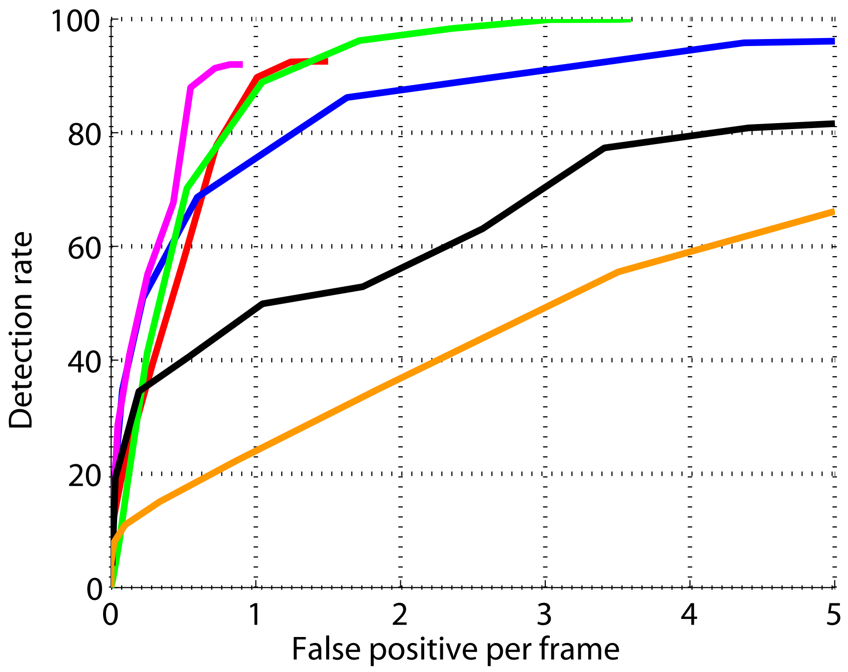


Figure 3.20: SVM confidence ROC curves. Our geometry method based on combining multiple SCOOP candidates is shown in red. The orange curves reflect our geometry based method without the use of object-shadow relationship refinement or centroid localization. The magenta curves are for our geometry based method augmented with temporal information. A standard full frame detector (HOG) is shown in blue. Green shows results obtained from classifying blobs obtained through registration, motion detection, and tracking, similar to [77]. The **Black** curves are for classifying tracks of clusters of Harris corners.

we proposed a method for estimating the constraints directly from image data. The constraints were then used to obtain candidate out-of-plane objects which were then classified as either human or non-human. For vehicles we combined multiple SCOOP candidates to obtain a vehicle candidate. In the case of humans, we evaluated the method on challenging data from the VIVID 3 and 2 datasets, and obtained results superior to both full frame search, motion constrained detection, and Harris track constrained detection. In the case of vehicle detection, we performed evaluation on the VIVID3 and CLIF datasets obtaining superior results. The purpose of the method is to augment the performance of any full frame classifier but could

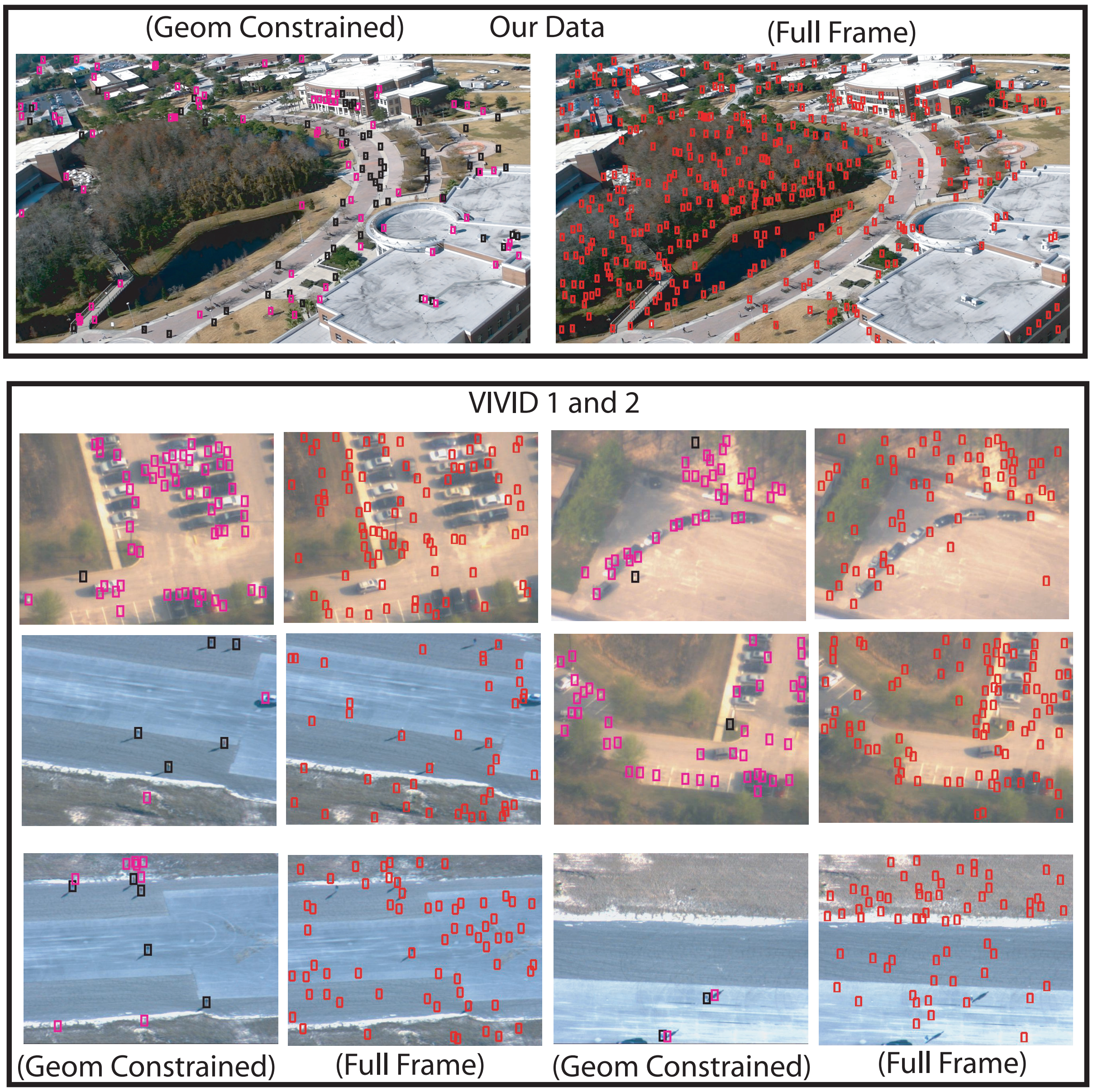


Figure 3.21: Qualitative detection results on VIVID 3, VIVID 2, and some of our own data. Columns labelled (Full Frame) show the result of full frame search (HOG) applied to the entire frame (human detections are shown in red). Columns labelled (Geom Constrained) show the results of our geometry constraint based method. Human candidates that were discarded by the wavelet classifier as clutter are shown in magenta, candidates that were classified as human are shown in **black**.

also be used for shadow detection and removal in the case of background subtraction based surveillance.

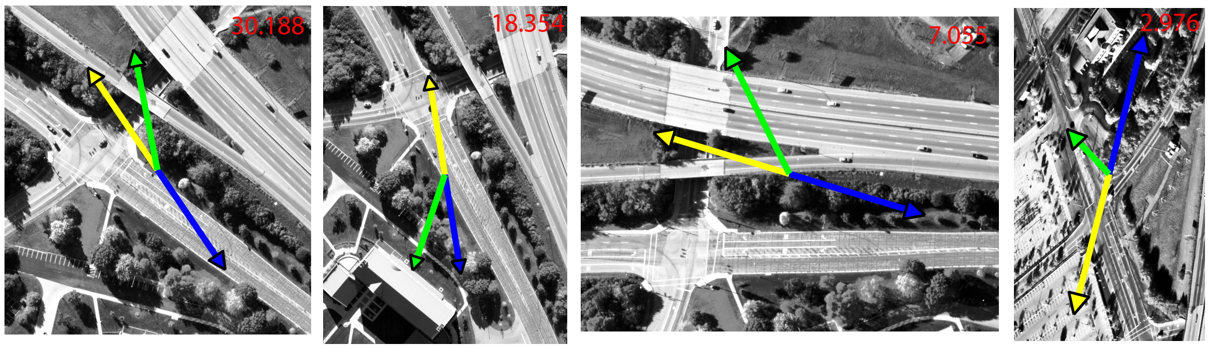


Figure 3.22: This figure shows the metadata derived geometric constraints for images from the CLIF 2007 dataset. The yellow arrow is the vector $\vec{\mathbf{S}}'$ pointing towards the sun. The blue arrow is a vector pointing in the direction of the shadow (opposite of sun direction). The green arrow is the vector $\vec{\mathbf{Z}}'$ pointing in the direction of the normal. The shadow to normal ratio is shown in red text. Note that in the first three images the camera is close to NADIR, hence the shadow to normal ratio is very high.

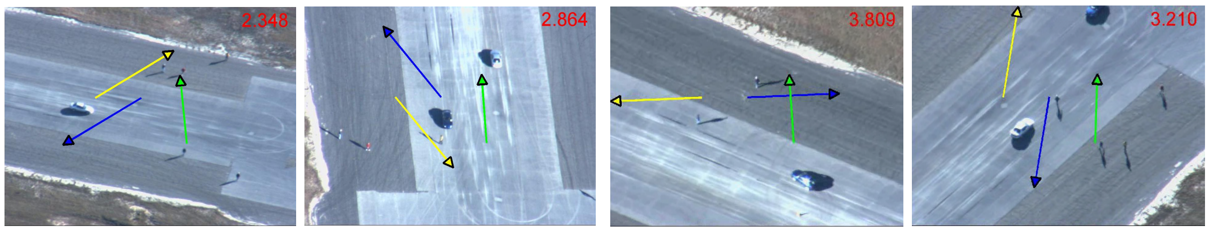


Figure 3.23: This figure shows metadata derived geometric constraints for images from the VIVID 3 dataset. Yellow arrow, is the vector $\vec{\mathbf{S}}'$ pointing towards the sun. Blue arrow is a vector pointing in the direction of the shadow (reverse of sun direction). Green arrow is the vector $\vec{\mathbf{Z}}'$ pointing in the direction of the normal. The shadow to normal ratio is shown as red text.

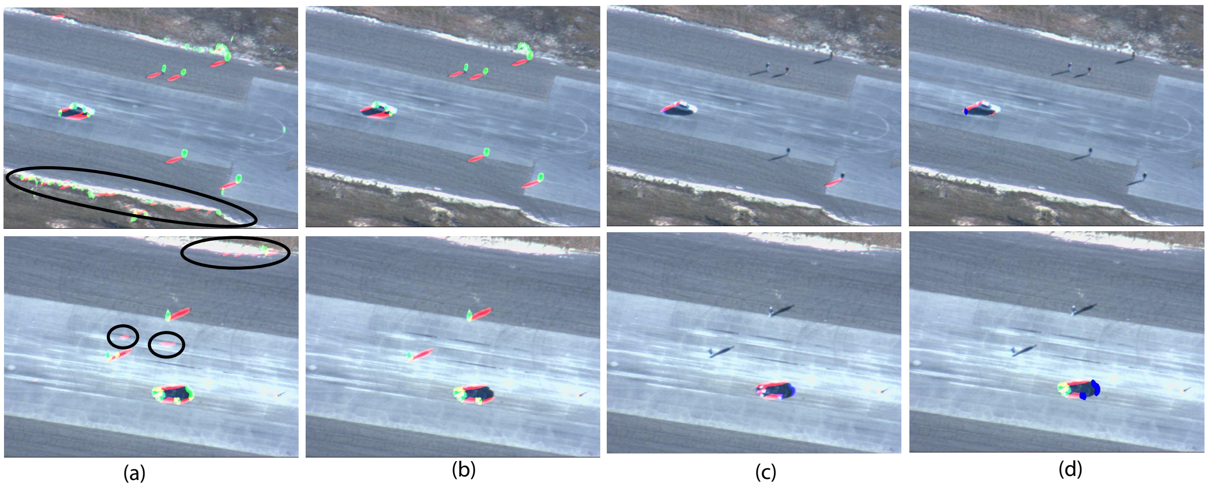


Figure 3.24: This figure shows the candidate refinement process. Figure **(a)** shows all of the shadow and normal blobs that were obtained using the method described in Section 3.5.1. Figure **(b)** shows the set $\mathbf{K}_{\vec{\mathbf{s}}'}^{\vec{\mathbf{z}}'}$ of refined SCOOP candidates after refining the blobs by exploiting object shadow relationship, as described in Section 3.5.2; this set is used as human candidates in the classification stage. Note that a lot nonsense shadow and normal blobs (circled in **black**) were removed from the areas of strong gradient. Figure **(c)** shows the set $\mathbf{K}_{-\vec{\mathbf{s}}'}^{-\vec{\mathbf{z}}'}$ of refined inverse scoop blobs, which were obtained using method described in 3.5.2, but where the sun and normal directions were reversed. Finally, figure **(d)** show vehicle candidates obtained by combining SCOOP and inverse SCOOP candidates using the method described in Section 3.6.1.

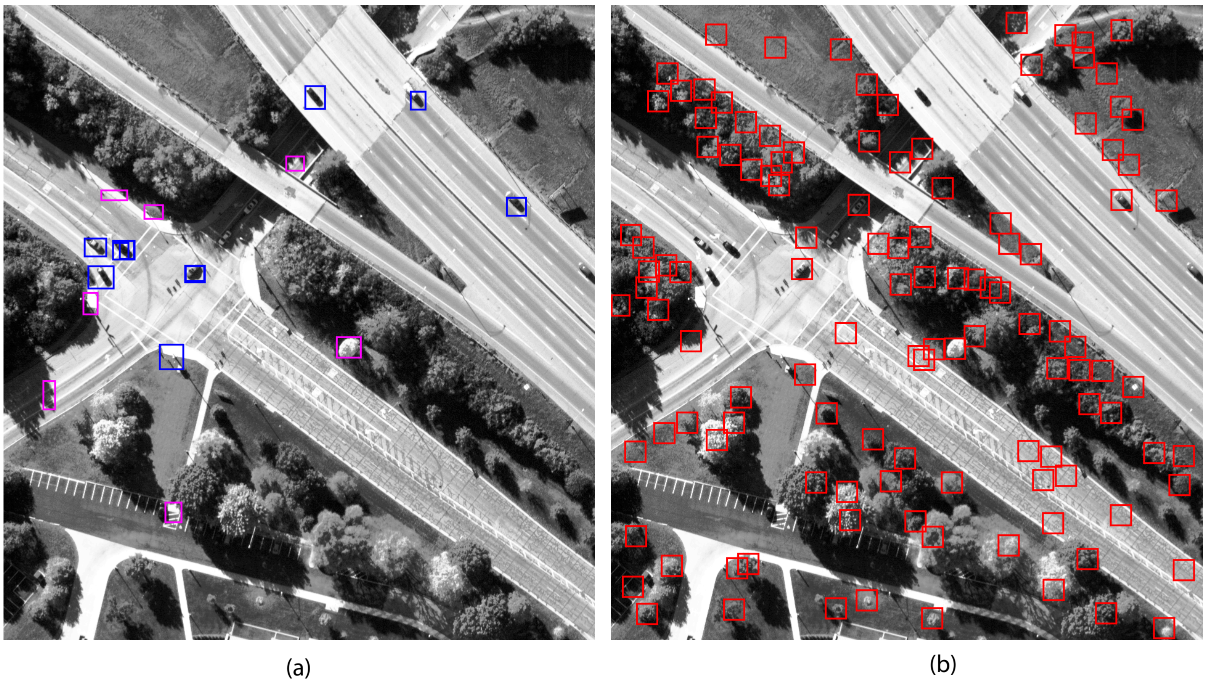


Figure 3.25: Figure **(a)** shows the vehicle detection results for the geometrically constrained method on the CLIF 2007 dataset. The candidates that were classified as vehicles are shown in blue. The candidates that were discarded by the classifier are shown in magenta. Figure **(b)** shows results for full frame detection in red. Full frame detection has generated significantly more false positives. Note that in the top image, one can see two vehicles that our method misses because their shadows are obscured by the shadow cast by the bridge, full frame detector misses only one of those vehicles.

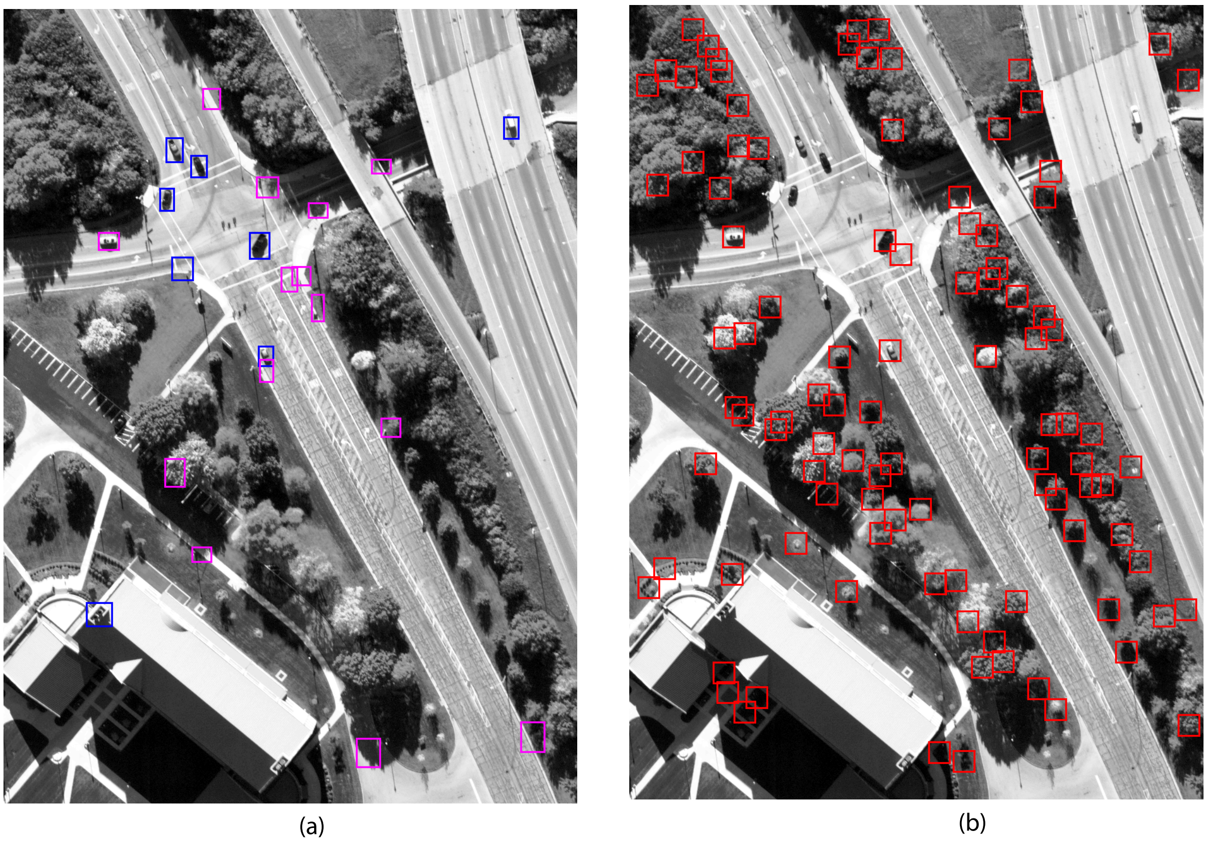


Figure 3.26: Figure **(a)** shows the vehicle detection results for the geometrically constrained method on the CLIF 2007 dataset. Candidates that were classified as vehicles are shown in blue, candidates that were discarded by the classifier are shown in magenta. Figure **(b)** show results for full frame detection in red. Full frame detection has generated significantly more false positives. Note that in the top image, one can see two vehicles that our method misses because their shadows are obscured by the shadow cast by the bridge the full frame detector misses only one of those vehicles.

CHAPTER 4: TRACKING A LARGE NUMBER OF DENSE OBJECTS IN LOW FRAMERATE VIDEO

In this chapter, we present a framework for tracking a large number of dense, fast moving objects in low frame rate video. In order to deal with the sheer number of objects present in the scene, we break the scene up into overlapping grid cells. This splits the tracking problem up into a number of smaller problems which are solved optimally using the Hungarian algorithm. In order to deal with the difficulties of fast object motion, high density, and low frame rate we propose utilizing scene constraint information which we derive directly from the imagery. We use these constraints to generate better assignment weights, which are used to solve the tracking problem. The chapter is organized as follows. Section 4.1 defines the tracking framework within each individual grid cell. In section 4.2, we describe how we generate and use the structured scene constraints to aid with the tracking problem. Section 4.3 describes how we track objects across multiple cameras. Finally, in section 4.4 we present metrics and results.

4.1 Formulating the Tracking Problem Within Cells

After detecting moving objects, we track them across frames using bipartite graph matching between a set of $label$ nodes (circled in blue) and a set of $observation$ nodes (circled in magenta). The assignment is solved optimally using the Hungarian algorithm which has complexity $O(n^3)$ where n is the number of nodes. When we have thousands of objects in the scene, an optimal solution for the entire scene is intractable. To overcome this problem, we

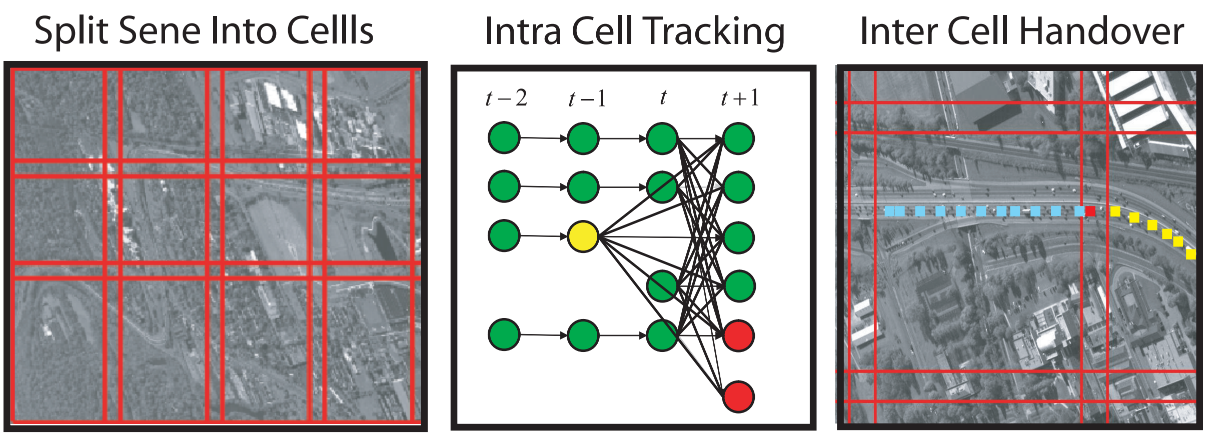


Figure 4.1: The figure shows the major steps of our tracking pipeline.

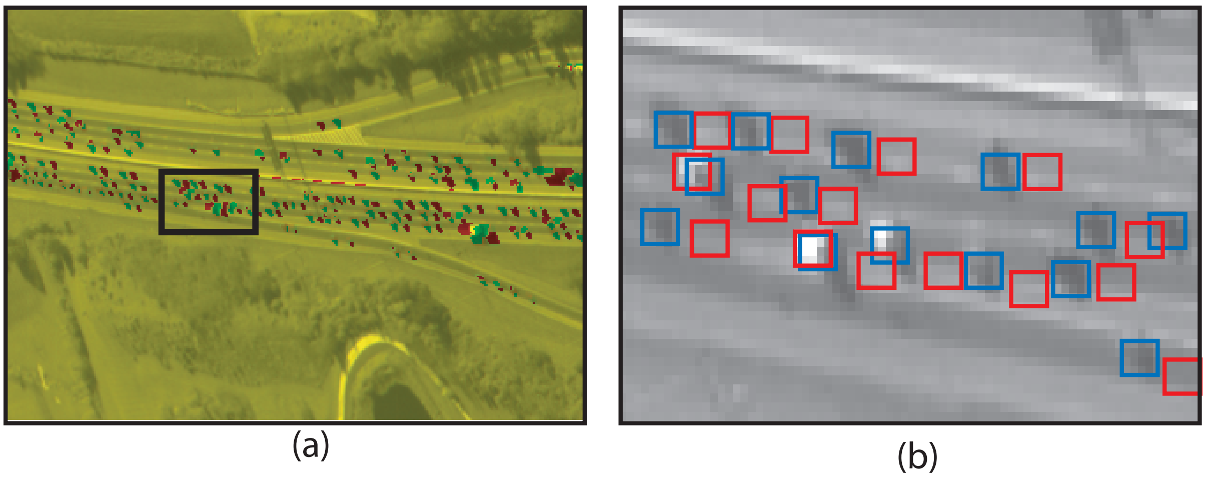


Figure 4.2: (a) shows two consecutive frames overlayed in two different color channels: red is frame t, green is frame $t+1$. (b) shows how far the vehicles move between consecutive frames. Red boxes show vehicle's positions in previous frame and the blue boxes show vehicle's positions in next frame.

break up the scene into a set of overlapping grid cells (see Figure 4.6). We solve the correspondence problem within each grid cell independently and then link tracks across grid cells. The use of grid has an additional advantage of allowing us to exploit local structured-scene constraints for objects within the grid cell, which will be discussed later.

For each grid cell in every pair of frames we construct the following graph. Figure 4.3 shows an example graph constructed for assigning labels between frames t and $t+1$. We add

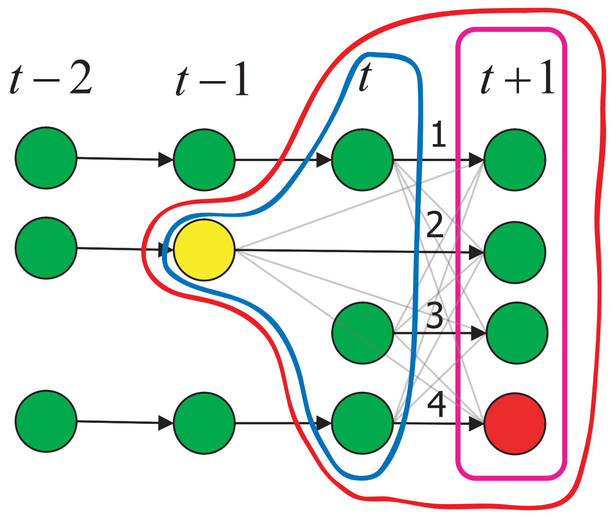


Figure 4.3: The figure shows an example of the bipartite graph that we solve at every frame. Four different types of edges are marked with numbers.

a set of nodes for objects visible at t to the set of $label$ nodes. A set of nodes for objects visible at $t+1$ are added to the set of $observation$ nodes, both types are shown in green. Since objects can exit the scene, or become occluded, we add a set of occlusion nodes to our $observation$ nodes, shown in red. To deal with the case of reappearing objects, we also add $label$ nodes for objects visible in the set of frames between $t-1$ and $t-p$, shown in yellow. We fully connect the $label$ set of nodes to the $observation$ set of nodes, using four types of edges.

1. Edge between label in frame t and an observation in frame $t+1$.
2. Edge between label in frame $t-p$ and an observation in frame $t+1$.
3. Edge between a new track label in frame t and an observation in frame $t+1$.
4. Edge between a label and an occlusion node.

We define edge weights in the following manner. Weight for edge of type 3 is simply a constant δ. Weights for edges of type 1 and 2 contain velocity orientation and spatial proximity components. Spatial proximity component C_p is given by

$$C_p = 1 - \frac{\|x^{t-k} + v^{t-k}(k+1) - x^{t+1}\|}{\sqrt{S_x^2 + S_y^2}}, \tag{4.1}$$

where x is the position of the object, S_x and S_y are the dimensions of the window within which we search for a new object and k is the time past since last observation of the object.

Velocity orientation component C_v is given by

$$C_v = \frac{1}{2} + \frac{\vec{v}^t \cdot \vec{v}^{t+1}}{2\|\vec{v}^t\|\|\vec{v}^{t+1}\|}, \tag{4.2}$$

where $\vec{v}^t$ is the last observed velocity of an object, $\vec{v}^{t+1}$ is the difference between x^{t+1}, the position of observation in current frame, and x^{t-k}, the last observed position of object at frame $t-k$.

We define the weight for edges of type 1 and 2 as follows

$$w = \alpha C_v + (1-\alpha) C_p. \tag{4.3}$$

We found these to be sufficient when object's velocity is available. If on the other hand, velocity of the object is unavailable as in initial two frames or when new objects appear in the scene, we use structured scene constraints to compute weights for edges.

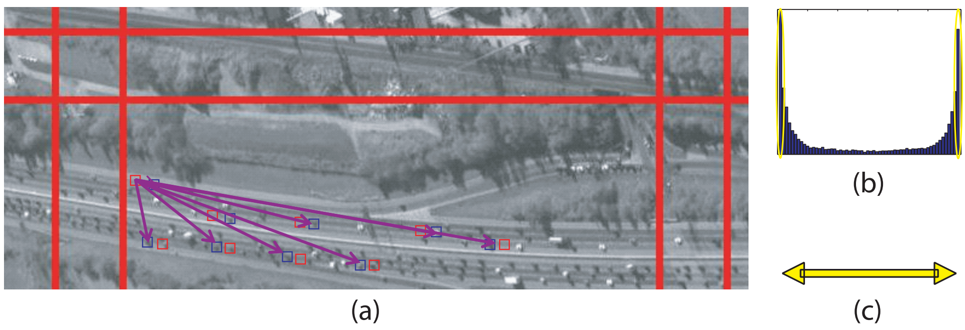


Figure 4.4: This figure shows the process of estimating road orientation within a grid cell. Objects tracked in frame t are shown in red, objects detected in frame $t+1$ are shown in blue. (a) Obtain all possible assignments between the objects in frame t and frame $t+1$. (b) Obtain a histogram of the resulting possible velocities. (c) Take the mean of velocities which contributed to the histogram peak.

4.2 Structured Scene Constraints

Assigning labels based simply on proximity between object centroids is not meaningful in wide area scenario. Due to low sampling rate (2 Hz), high scene density and high speed of objects, proximity based assignment is usually incorrect (see Figure 4.5). Therefore we use road orientation estimate and object context as constraints from the structured scene.

Road orientation estimate $\vec{g}$ is computed for each grid cell in the following manner (see Figure 4.4). First, we obtain all possible assignments between objects in frame t and $t+1$. This gives us a set of all possible velocities between objects at frames t and $t+1$. Next, we obtain a histogram of orientations of these velocities and take the mean of orientations that contributed to peak of the histogram. See **Algorithm 1** for a formal description.

Algorithm 1 Algorithm to compute global velocity for each cell in grid of size m x n using detections D_t and D_{t+1}.

```
1: procedure COMPUTEGLOBALVELOCITY
2:     for $i \leftarrow 1, m$ do
3:         for $j \leftarrow 1, n$ do
4:             for all $d \in D_t^{i,j}$ do
5:                 for all $d' \in D_{t+1}^{i,j}$ do
6:                     $\theta = \tan^{-1}(d' - d)$
7:                     Store $\theta$ in $\Theta$
8:                 end for
9:             end for
10:            $h = histogram(\Theta)$
11:            Find bin $\psi$ s.t. $mode(h) \in \psi$
12:            $\theta' = mean(\theta | \theta \in \psi)$
13:            $\vec{g}(i, j)$ = $[\cos(\theta')\ \ \sin(\theta')]$
14:        end for
15:    end for
16: end procedure
```

Algorithm 2 Algorithm to compute context $\Phi(O_t^a)$ for object a at frame t.

```
1: procedure COMPUTECONTEXT
2:     for all $c$ do
3:         if $\|O_t^c - O_t^a\|_2 < r$ then
4:             $\theta = \tan^{-1}(O_t^c - O_t^a)$
5:             $d = \|O_t^c - O_t^a\|_2$
6:             $\Phi = \Phi + \mathcal{N}(\mu, \Sigma)$                ▷ $\mathcal{N}$ centered on $(d, \theta)$
7:         end if
8:     end for
9: end procedure
```

Note that orientation of $\vec{g}$ essentially gives us orientation of the road along which vehicles travel, it does not give us the direction along that road. However, even without the direction, this information is oftentimes sufficient to disambiguate label assignment as shown

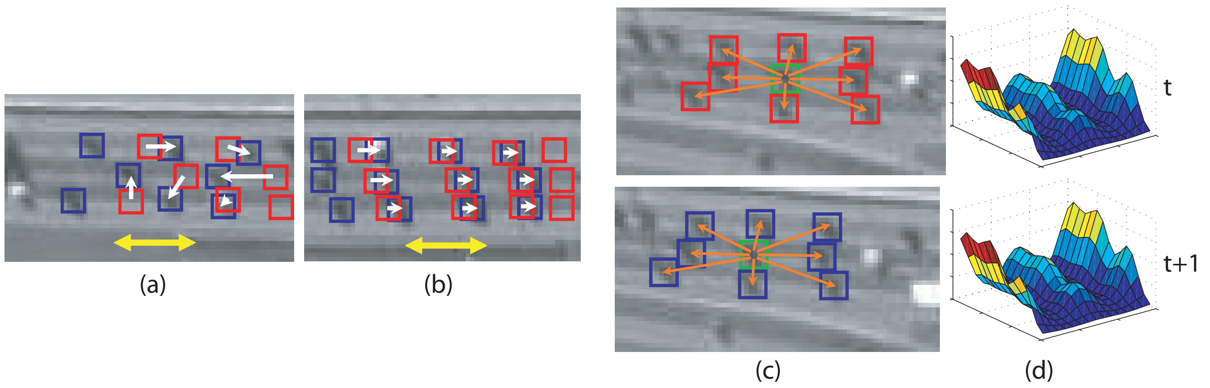


Figure 4.5: Vehicles tracked at time t are shown in red, while vehicles detected in frame $t+1$ are shown in blue. White arrows indicate the assignment of labels to objects based on proximity only and correspond to the resulting velocities of objects. Yellow arrows indicate the road orientation estimate for this particular grid cell. (a) shows a case where road orientation estimate can be used to disambiguate the assignment of labels, and (b) shows where it is not useful. To handle cases such as (b), we introduce a new constraint for the context of each vehicle, as shown in (c). At frames t and $t+1$, we compute vectors between the vehicle of interest (green) and its neighbors (orange). We then compute a 2D histogram of orientations and magnitudes of the vectors as shown in (c).

in Figure 4.5(a). When vehicles travel along the road in a checkerboard pattern, proximity based assignment will result in velocities which are perpendicular to $\vec{g}$. That is not the case when a number of vehicles are travelling in a linear formation as in Figure 4.5(b). Therefore, we introduce an additional formation context constraint (see Figure 4.5(c) and Figure 4.5(d)). If we are trying to match an object O_a in frame t (or $t-k$) to an observation in frame $t+1$, we compute object context as a 2 dimensional histogram of vector orientations and magnitudes between an object and its neighbors.

In order to account for small intra-formation changes, when computing the context histograms Φ_a and Φ_b, we add a 2D Gaussian kernel centered on the bin to which a particular vector belongs. Furthermore, since $0°$ and $360°$ are equivalent, we make the kernel wrap around to other side of orientation portion of the histogram.

The road orientation constraint component is defined as

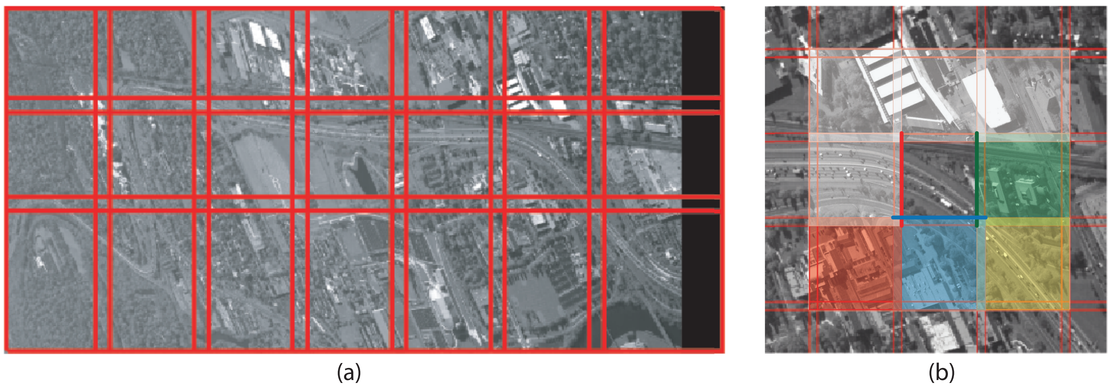


Figure 4.6: (a) This figure shows an example frame with the grid overlayed onto an image. (b) shows the grid cell search procedure for handing over tracks. The bold colored lines correspond to OLLeft, OLBottom, and OLRight, in counterclockwise direction. Only colored grid cells are searched, white cells are ignored.

$$C_g = \frac{1}{2} + \frac{|\vec{g} \cdot \vec{v}^{t+1}|}{2\|\vec{g}\|\|\vec{v}^{t+1}\|} \tag{4.4}$$

The purpose of this constraint is to prevent tracks from travelling across the road. The context constraint is the histogram intersection between histograms Φ_a and Φ_b:

$$C_c = \sum_{p}^{Nbins} \sum_{q}^{Mbins} min(\Phi_a^{p,q}, \Phi_b^{p,q}) \tag{4.5}$$

Finally, weight for edge of type 3 is computed as follows,

$$w = \alpha_1 C_g + \alpha_2 C_p + (1 - \alpha_1 - \alpha_2) C_c \tag{4.6}$$

We solve the resulting bipartite graph using Hungarian algorithm. We track all objects within each grid cell by performing the above procedure for all frames. Next, we find and link tracks that have crossed the cell boundaries, using **Algorithm 3** utilizing the overlapping regions of the neighboring grid cells. (see Figure 4.6 for reference).

Algorithm 3 Algorithm for object handover across grid cells. The size of grid is m x n. $S(i, j)$ represents all tracks for the sequence in the cell at i^{th} row and j^{th} column in grid.

```
1: procedure INTERCELLHANDOVER
2:     for $i \leftarrow 1, m$ do
3:         for $j \leftarrow 1, n$ do
4:             Calculate OLLeft, OLRight and OLBottom                          ▷ See figure Figure 4.6
5:             for all $s^{i,j} \in S(i, j)$ do
6:
7:                 if $\exists\ k \mid s_k^{i,j} > OLRight$ then
8:                     completeTrack($s^{i,j}, S(i + 1, j)$)
9:                 else if $\exists\ k \mid s_k^{i,j} > OLRight\ \wedge \exists\ k \mid s_k^{i,j} > OLBottom$ then
10:                    completeTrack($s^{i,j}, S(i + 1, j + 1)$)
11:                else if $\exists\ k \mid s_k^{i,j} > OLBottom$ then
12:                    completeTrack($s^{i,j}, S(i, j + 1)$)
13:                else if $\exists\ k \mid s_k^{i,j} < OLLeft\ \wedge \exists\ k \mid s_k^{i,j} > OLBottom$ then
14:                    completeTrack($s^{i,j}, S(i - 1, j + 1)$)
15:                end if
16:
17:            end for
18:        end for
19:    end for
20: end procedure
```

```
1: procedure COMPLETETRACK($s, S$)                 ▷ s=track to complete, S=tracks in neighboring cell
2:     for all $s' \in S$ do
3:         if $\exists\ (l, m) \mid s_l.detectionID = s'_m.detectionID \wedge s_l.t = s'_m.t$ then
4:             assign $s$ and $s'$ unique label
5:         end if
6:     end for
7: end procedure
```

4.3 Handling Multiple Cameras

There can be several possible frameworks for tracking objects across overlapping cameras which employ inter-camera transformations. One possible way is to establish correspondences at the track level where objects are detected and tracked in each camera independently, and afterwards, tracks belonging to the same object are linked. But, this approach has a serious issue which arises from the fact that background for a particular frame of a camera can only be modeled on overlapping region of all frames used for background. This reduces the area of region where objects can be detected. When objects are detected in cameras separately, reduction in detection regions results in the loss of overlap between two cameras. While methods for matching objects across non-overlapping cameras exist [31, 52, 33, 2], low resolution and

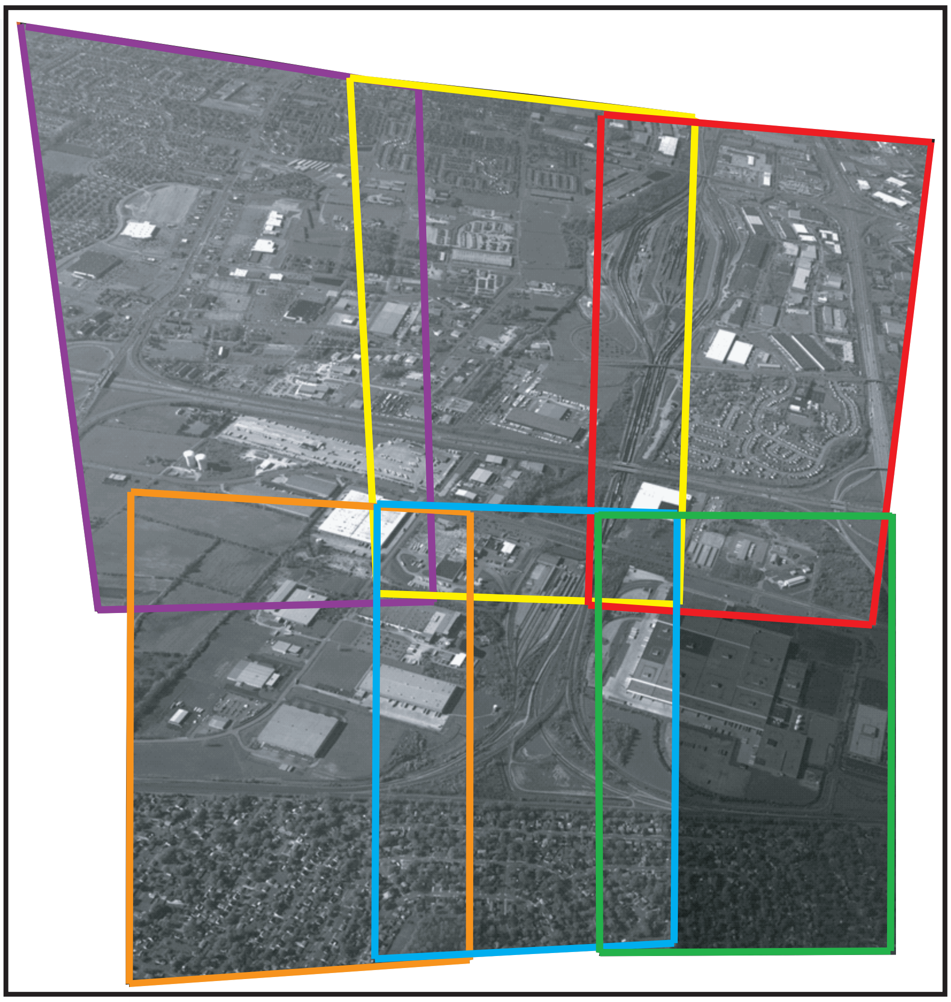


Figure 4.7: CLIF data - all six cameras.

single channel data disallow the use of appearance models for object hand over, and reacquisition based on motion alone is ambiguous. The increased gap between cameras arising from detection adds further challenge to a data already characterized by high density of objects and low sampling rate of video.

In order to avoid above problems, we perform detection and tracking in global coordinates. We first build concurrent mosaics from images of different cameras at a particular time instant using the Registration method in §3.1 and then register the mosaics treating each concurrent mosaic as a single image.

One problem with this approach, however, is that cameras can have different Camera Response Functions or CRFs. This affects the median background, since intensity values for each pixel now come from multiple cameras causing performance of the detection method to deteriorate. To overcome this issue, we adjust the intensity of each camera with respect to a reference camera using the gamma function [16] i.e.

$$I'_C(x,y) = \beta I_C(x,y)^{\gamma}, \tag{4.7}$$

where $I_C(x,y)$ is the intensity of the original image at location (x,y). We find β, γ by minimizing the following cost function:

$$\underset{\beta,\gamma}{\operatorname{argmin}} \sum_{(x,y)\in I_{C1}\cap I_{C2}} (I_{C1}(x,y) - I'_{C2}(x,y))^2, \tag{4.8}$$

where $I_{C1} \cap I_{C2}$ is the overlap between the two cameras. The cost function is minimized using a trust region method for nonlinear minimization. The approximate Jacobian matrix is calculated by using finite difference derivatives of the cost function. Transformation in equation 4.7 is then applied to each frame of the camera before generating concurrent mosaics. Results for this procedure are shown in Figure 4.8.

4.4 Results

We validated our method on four sequences from CLIF 2006 dataset. Sequences 1 to 3 are single camera sequences while sequence 4 has multiple cameras. The average number of objects in these sequences are approximately 2400, 1000, 1200 and 1100 respectively. Objects in sequence 2 and 3 undergo merging more often than objects in the other two sequences. This

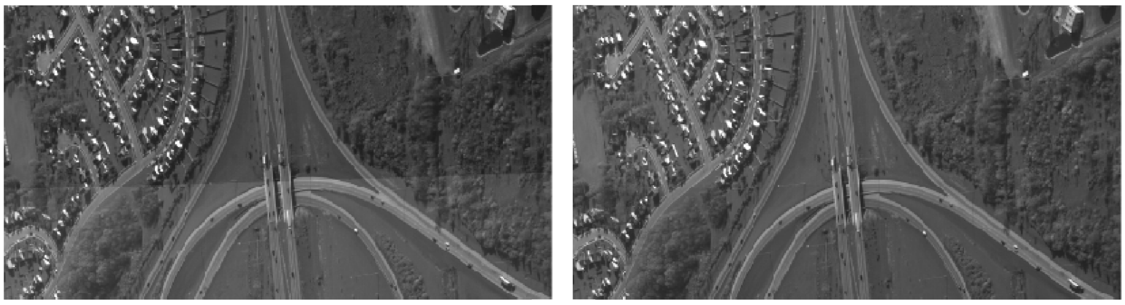


Figure 4.8: This figure shows the results of multi-camera intensity equalization. Notice the seam in the image on the left which in not visible in the equalized image on the right.

is primarily due to oblique angle between highway and camera in these sequences as opposed to top view in sequences 1 and 4. Figure 4.9 shows some of the tracks from these sequences.

For quantitative evaluation, we manually generated ground truth for the four sequences. Due the sheer number of objects, smaller size and similar appearance, generating ground truth for each object is a daunting task. We selected one region from sequence 1,3 and 4 and two regions from sequence 2 for ground truth. Objects were randomly selected and most of them undergo merging and splitting. The number of objects for which ground truth was generated are 34 for sequence 1, 47 and 60 for sequence 2 and 50 each for sequences 3 and 4.

Our method for evaluation is similar to [51] and measures performance of both detection and tracking. We compute the following distance measure between generated tracks and ground truth tracks:

$$D(T_a, G_b) = \frac{1}{|\Omega(T_a, G_b)|^2} \sum_{t \in \Omega(T_a, G_b)} \|\mathbf{x}_t^a - \mathbf{x}_t^b\|^2, \tag{4.9}$$

where $\Omega(T_a, G_b)$ denotes the temporal overlap between T_a and G_b, $|.|$ denotes cardinality while $||.||$ is the Euclidean norm. A set of pairs are associated i.e. $(a, b) \in A$ iff T_a and G_a have an overlap. The optimal association,

$$A^* = \underset{A}{\text{argmin}} \sum_{(a,b)\in A} D(T_a, G_b) \text{ subject to } \Omega(T_a, T_c) = \emptyset \quad \forall (a, b), (c, b) \in A \tag{4.10}$$

is used to calculate the performance metrics. Abusing notation, we define

$$A(G_b) = \{T_a | (a, b) \in A\}. \tag{4.11}$$

The first metric *Object Detection Rate*, measures the quality of detections prior to any association:

$$ODR = \frac{\# \text{ correct detections}}{\# \text{ total detections in all frames}}. \tag{4.12}$$

We cannot compute ODR for each track and then average, because that would bias the metric towards short tracks as they are more likely to have all detections correct. Further notice that, it is not possible to detect false positives as the number of ground truth tracks is less than number of objects. A related metric, *Track Completeness Factor*,

$$TCF = \frac{\sum\limits_a \sum\limits_{T_b \in A(G_a)} |\Omega(T_b, G_a)|}{\sum_a |G_a|}, \tag{4.13}$$

measures how well we detect an object after association. TCF will always be less than or equal to ODR. The difference between ODR and TCF is the percentage of detections that were not included in tracks. Finally, *Track Fragmentation* measures how well we maintain identity of the track,

Table 4.1: Quantitative Comparison

		Our Method			GreedyBIP		
	ODR	**TCF**	**TF**	**NTF**	**TCF**	**TF**	**NTF**
Seq1	0.975	0.716	2.471	2.506	0.361	13.06	13.11
Seq2	0.948	0.714	2.894	2.895	0.489	12.55	12.55
Seq3	0.972	0.727	2.76	2.759	0.583	8.527	8.53
Seq4	0.984	0.824	1.477	1.48	0.638	6.444	6.443

$$TF = \frac{\sum_a |A(G_a)|}{|\{G_a | A(G_a) \neq \emptyset\}|}. \tag{4.14}$$

Weighing the number of fragments in a track with length, we get *Normalized Track Fragmentation*,

$$NTF = \frac{\sum_a |G_a| \cdot |A(G_a)|}{\sum_{a|A(G_a)\neq\emptyset} |G_a|}. \tag{4.15}$$

which gives more weight to longer tracks as it is more difficult to maintain identity for long tracks than short ones.

We compare our method with the standard bipartite matching using greedy nearest-neighbor initialization. Initial assignment is done based on proximity while linear velocity model is used for prediction. Standard gating technique is used to eliminate unlikely candidates outside a certain radius. The same registration and detection methods were used for all experiments. The values of parameters for our tracking method were α = 0.5 (eq. 4.3) and α_1 = α_2 = 0.33 (eq. 4.6). Table 4.1 shows the comparison between both methods:

As can be seen from Table 4.1, our method achieved better TCF and TF because unique characteristics of WAS demand the use of scene-based constraints which were not leveraged by the standard bipartite matching. We derived road orientation estimate and object context using only the image data, which allowed for better initialization and tracking performance.

4.5 Summary

We analyzed the challenges of a new aerial surveillance domain called Wide Area Surveillance, and proposed a method for detecting and tracking objects in this data. Our method specifically deals with difficulties associated with this new type of data: unavailability of object appearance, large number of objects and low frame rate. We evaluated proposed method and provided both quantitative and qualitative results. These preliminary steps pave way for more in-depth exploitation of this data such as scene modelling and abnormal event detection.

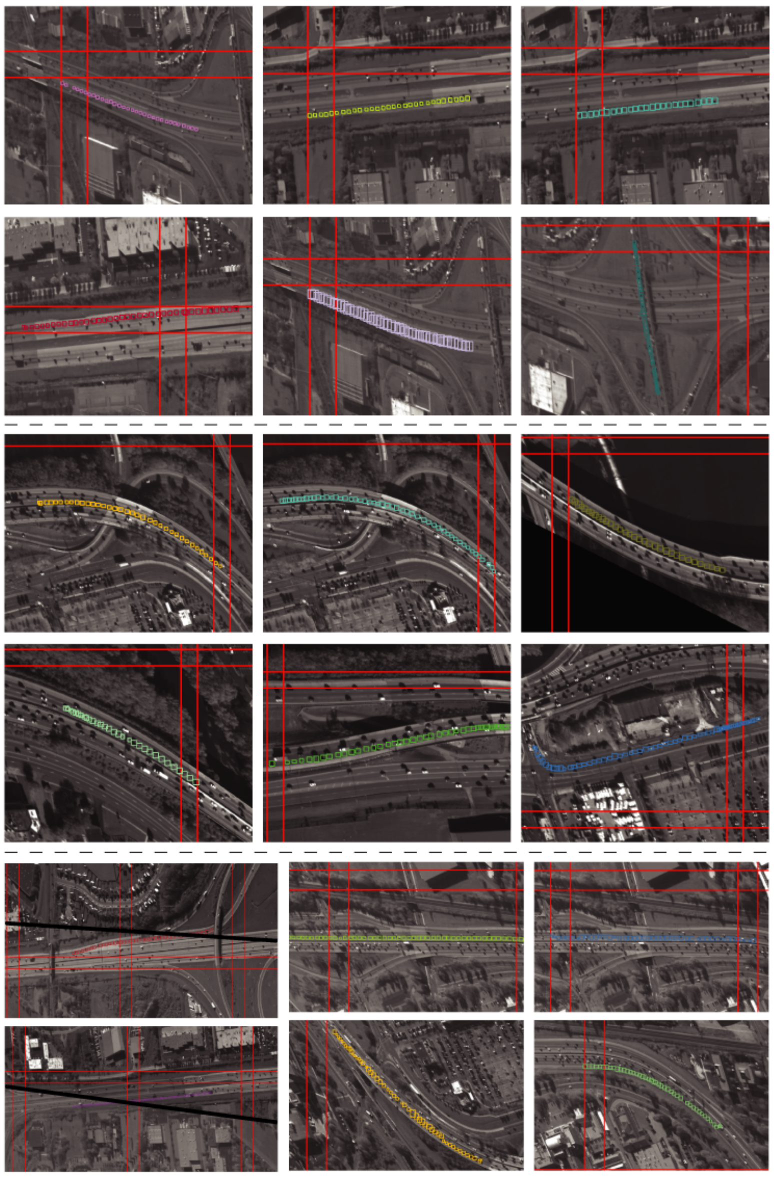


Figure 4.9: This figure shows a number of results for different sequences. The top group is for sequence 1, while the second group is for sequence 2. In the bottom group, first column is from the multiple camera sequence (camera boundary is shown in black), the next two columns are from sequence 4.

CHAPTER 5: ACTION RECOGNITION IN AERIAL VIDEO FROM FEW EXAMPLES

In this chapter we propose a method for recognizing actions in aerial video from few examples using a simple and popular action recognition framework. We use the easy to compute Bag of Words model in order to represent the actions, which usually requires a large number of examples. In order to overcome this limitation, we augment the classifier of the action class, which has only one example, with a a weighted fusion of responses of classes that have many examples.

5.1 Problem Description

One of the requirements of any recognition method is to construct a model of a class with sufficient generality so as to be able to capture variations within the class, class; This assures that these variations do not negatively affect the performance of the model during testing. In the case of computer vision, there are two main paradigms for tackling this problem. The first involves constructing a semantically meaningful representation of the class. For example, in the problem of human action recognition, a classic formulation is to analyze the kinematics of body parts or joints of the humans [9]. In this scenario, the semantic meaning comes from the knowledge about the anatomy of a human. While the methods are generally intuitive and elegant when it is assumed that the joints have already been detected, things become very difficult when the joints and/or limbs have to be detected automatically. Hence, the main problem with the first paradigm is that obtaining a semantically meaningful representation is a difficult problem in itself. Therefore, researchers have generally opted for the second paradigm of building generality into the model by utilizing a data-driven approach of constructing a model by using multiple examples.

The most common way to approach this particular paradigm is to utilize a dual stage framework of extracting features and then using machine learning to automatically learn the model from a set of training examples, as demonstrated [73] [49] [41] [62] [37]; However, some researchers have generalized templates to utilize multiple examples as well [57]. One representation of actions that is commonly used today is the bag of words model, similar to [15]. First, simple spatio-temporal interest points are detected. Then, a cuboid is extracted around each point, and descriptors of gradient that describe the local motion and appearance are obtained around each point. Next, these descriptors are clustered into spatio-temporal words using K-means to obtain a collection of clusters. Each cluster represents a visual word, and it can be determined which cluster or word each cuboid belongs to. Therefore, the videos in the dataset can be represented by a histogram of visual words (See Figure 5.1). The advantage of this approach is that primitive spatio-temporal features are easy to detect, hence this framework can be used in many different cases. The disadvantage is that since the words are not very semantically meaningful they pass the problem of generalization to the next stage of the framework, namely classification, as we illustrate below.

Once the histogram descriptors of every video have been obtained, one can take the histograms and automatically construct a model of each class by utilizing a Support Vector Machine, or SVM. In a two class case, given a set of tuples $\langle \mathbf{x}, y \rangle$, where $\mathbf{x}$ is a feature vector

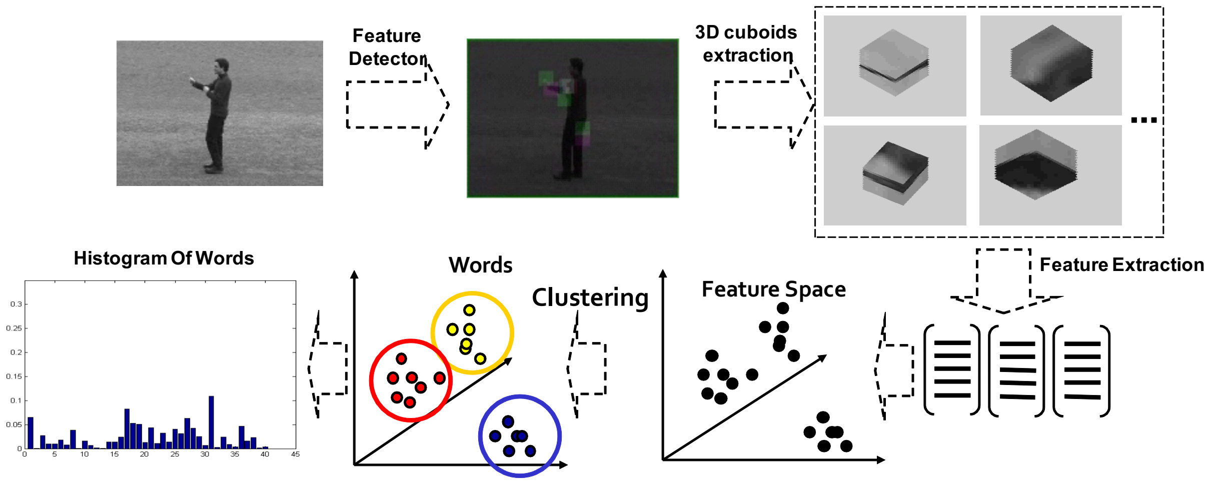


Figure 5.1: This figure illustrates the Bag of Words processing stages.

and $y \in \{-1, 1\}$ is the class label, an SVM will construct a model $\{\mathbf{w}, b\}$, s.t. the label $\hat{y}$ of an unknown example $\hat{\mathbf{x}}$ is given by

$$f(\hat{\mathbf{x}}) = sign(\mathbf{w}^T\hat{\mathbf{x}} + b). \tag{5.1}$$

Where $\mathbf{w}$ is the normal to the hyperplane that best separates the data, and b is the hyperplane bias. Because $\mathbf{w}$ is obtained by solving the dual of the original problem. The actual decision function that is used in practice is

$$f(\hat{\mathbf{x}}) = sign(\sum_{k=1}^{N} \alpha_i \mathbf{x}_i^T \hat{\mathbf{x}} + b) \tag{5.2}$$

where N is the number of examples, and α_i is the lagrange multiplier for that particular example. SVM is a maximum margin classifier that selects only the examples that are closest to the hyperplane and discards the rest. The lagrange multiplier α is zero for examples that have been discarded, and non-zero for the examples that have been retained, which are referred to as the support vectors. If a kernel function K is used for matching the vectors, then the decision function becomes

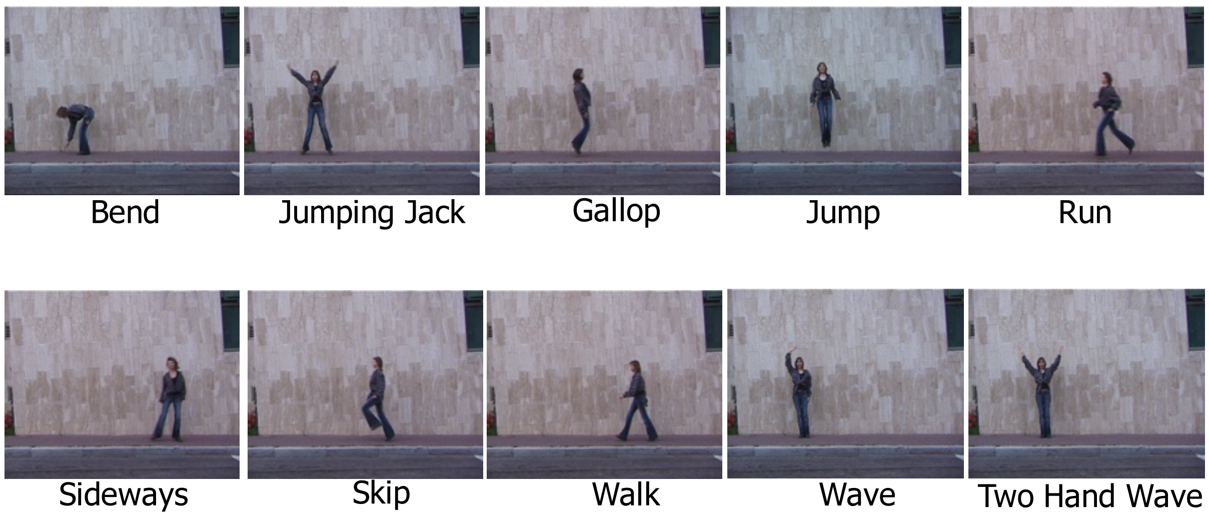


Figure 5.2: Examples from the Weizmann dataset of actions.

$$f(\hat{\mathbf{x}}) = sign(\sum_{k=1}^{N} \alpha_i K(\mathbf{x}_i, \hat{\mathbf{x}}) + b). \tag{5.3}$$

The number of support vectors required to separate the data can be viewed as an indication of the complexity of the model, which is needed in order to recognize the classes. The more support vectors that SVM has selected, the more complicated the model is. This can be viewed as an indication of the separability of the data, and by extension the quality of the features that are used to represent the data. This can be illustrated in the case of action recognition in the following experiments on the Weizmann Action dataset; Examples of which can be seen in Figure 5.2.

The Weizmann action dataset contains 10 classes, with 9 examples per class. When actions from this dataset are represented using the Bag of Words model, where each video is represented by a histogram vector of 1000 dimensions, cross validation performance obtained on the data is 98.8%. However, almost all of the data was selected as support vectors, 82 examples or 91.11%. In contrast, when actions were represented by a 25 dimensional histogram of motion patterns from [60], only 22 examples or 24.44% of the data was selected as support vectors, while the cross-validation performance was identical at 98.8%. The low number of support vectors indicates that we do not need a very large number of examples in order to learn the model. We can confirm this by repeatedly forcing one class in the dataset to have only one training example, in that case the performance drops only slightly to 97.36%. In contrast, in the case of BOW representation, the performance drops significantly to 24.16% when few examples are available. Applying various dimensionality reduction techniques, such as PCA or PLSA, on Bag of Words representation does not improve this result, which indicates that this is not just a matter of dimensionality. Unfortunately, generating motion patterns is a more sensitive and involved process than the Bag of Words and may not necessarily be applicable to data that is blurrier, nosier, and more contaminated with camera motion than Weizmann.

In summary, there is a tradeoff between how semantically meaningful the action representation is, how easy it is to compute, and how many examples are required to construct a meaningful action model. Complex meaningful representations can be difficult to obtain, but in the end require few examples to construct the final model. While simple, easy to compute representations require a large number of examples to construct the final model, and exhibit degradation of performance when those examples are not available.

As we have pointed out before, there are a number of problems that appear in aerial video. The moving camera can create additional motion blur in the imagery, motion compensation can introduce geometric distortions, and there are more possible viewpoints that an action can take. Therefore, in the case of aerial video, simple, easy to compute action represen-

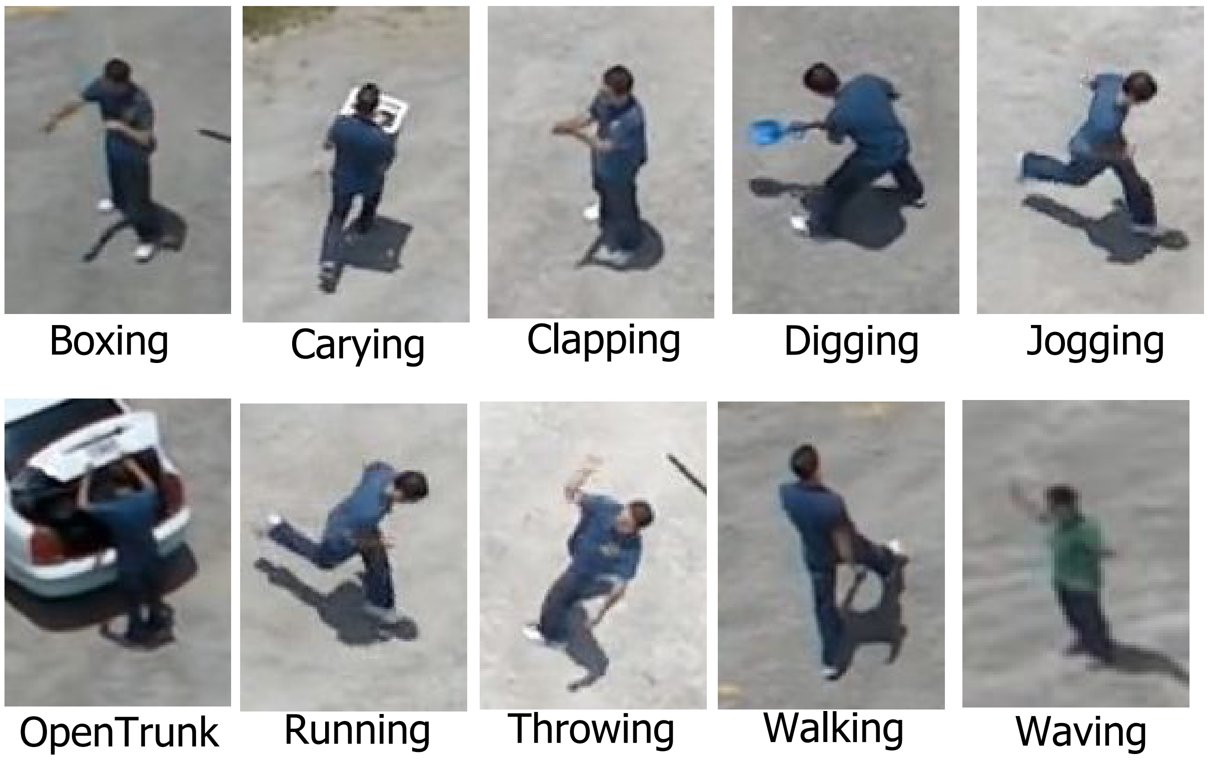


Figure 5.3: Examples from the UCF-ARG Dataset of actions.

tation should be utilized, which unfortunately, requires a larger number of examples in order to construct the model. However, these examples may not always be available due to the added difficulty of having to fly the UAV while collecting the data (See Figure 5.3).

5.2 Method

In this section, we describe our experimental setup, as well as the details of our method. We assume that one of the action classes has only one example, while the rest of the classes have many examples. We want to augment the model of the one-shot class by transferring knowledge from similar classes that have many examples. This is somewhat similar to [3], however our approach is different in the following ways: First, we want to select multiple similar classes, and to do so automatically. In contrast to [3], only one similar class was used and it was selected manually. Second, we want to include the similar classes in testing. In [3], on the other hand, similar classes were not included in testing. Including similar classes in testing makes the problem more challenging.

5.2.1 Experimental Setup and Definitions

Prior to describing our method in detail we feel that it is necessary to describe the setup of our experiments in detail . We have N classes, and M_i examples per class. Each example is a tuple $\langle \mathbf{x}, c \rangle$, where $c \in \{1, ..., N\}$. Adopting a 1vsAll multi-class classification framework, we train a binary Support Vector Machine $f_c(\mathbf{x})$ for each class c. So that $f_c(\mathbf{x}) > 0$ if $\mathbf{x} \in c$ and $f_c(\mathbf{x}) < 0$ if $\mathbf{x} \notin c$. The predicted class $\hat{c}$ of the unknown example $\hat{\mathbf{x}}$ is then given by

$$\hat{c} = \frac{argmax}{i} \{f_1(\hat{\mathbf{x}}), ..., f_i(\hat{\mathbf{x}}), ..., f_N(\hat{\mathbf{x}})\}. \tag{5.4}$$

It is typically assumed that all classes have many examples $M_i >> 1$. The standard procedure is to repeatedly split examples for each run j from each class i into training and testing sets R_i^j and S_i^j, respectively, where the cardinality of the sets is $M_i > |R_i^j| \gg 1$ and $M_i > |S_i^j| \geq 1$. Then, training and testing is performed for each split, and the results are averaged across different runs. The resulting framework performs quite well on a number of datasets. We refer to this approach as *Multi-Shot* classification. In the right-most column of Figure 5.10 we show the result of performing this experiment on the 6 class KTH action dataset, using around 85 training and 15 testing examples per class. The performance is clearly quite good when many examples are available, on average an accuracy rate of 90% was achieved.

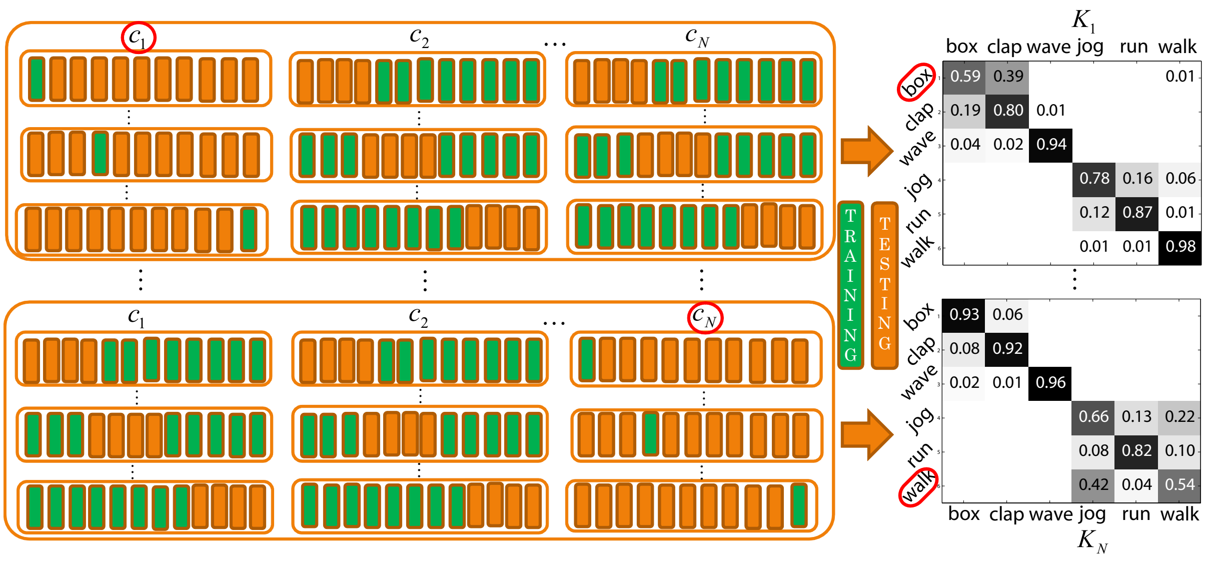


Figure 5.4: This figure shows our experimental setup. The vertical rectangles are individual feature vectors. The confusion matrices displayed here are for KTH data, using our proposed fusion method.

In order to test the performance of the one-shot approach, we perform the following procedure (see Figure 5.4 for reference). We repeatedly select a class to be the one-shot class o. Next we split examples from each of the classes in the following manner: we select only one example for training from class o so that $|R_o^j|$ = 1 $|S_o^j| = M_o - 1$, on the other hand we perform standard selection for other classes so $M_i > |R_i^j| >> 1$ and $M_i > |S_i^j| >> 1$ for $j \neq o$. We perform training and testing for different selections j, combining the results into a confusion matrix K_o, and normalizing each row independently. We refer to this approach as *Naive One-Shot*, the resulting confusion matrices can be seen in the leftmost column of Figure 5.10. As can be seen from the confusion matrices, a problem arises when one of the classes has only one example. If one of the classes has only one example, then the Support Vector Machine corresponding to that class (let's call it $f_o(\mathbf{x})$) will have only one positive example. This also means that it will have only one positive support vector, and a smaller collection of negative support vectors than it would have if its corresponding class had many examples. When this

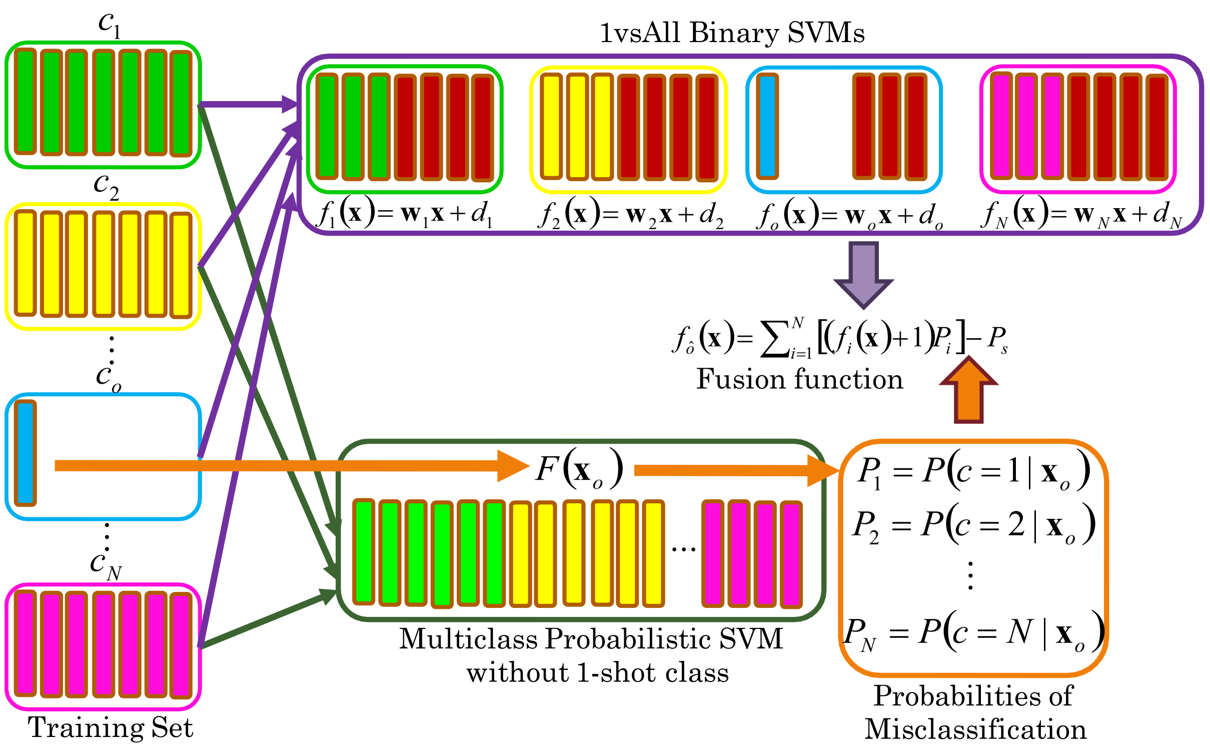
SVM $f_o(\mathbf{x})$ is used in the same pool as other multi-example SVMs, i.e. that the pool of binary SVMs is $\{f_1(\hat{\mathbf{x}}), ..., f_o(\hat{\mathbf{x}}), ..., f_N(\hat{\mathbf{x}})\}$, problems arise in the testing stage. As can be seen in the **left** column of the confusion matrices in Figure 5.10, the testing examples of the class that was selected as the one-shot class becomes greatly confused with other classes. In fact, in the case of the KTH dataset, none of the testing examples belonging to the one-shot class were classified correctly. However, the performance of the other classes is unchanged. In the next section, we will describe how to alleviate some of these problems.

5.2.2 Weighted Fusion

The fundamental idea behind our approach is that we replace the binary SVM for the one-shot class $f_o(\hat{\mathbf{x}})$, using the following fusion of all of the binary SVM responses

$$f_{\hat{o}}(\hat{\mathbf{x}}) = sign(\sum_{i=1}^{N}[f_i(\hat{\mathbf{x}})P_i^o] - P_s^o. \tag{5.5}$$

In the above equation, P_i^o is the probability of an example $\mathbf{x}_o$ from the one-shot class o being confused with class i when using the 1vsAll multi class classification framework with only one example available for class o. Formally, we define $P_i^o = P(\hat{c} = i|\mathbf{x}_o)$. Similarly, P_s^o is the probability of an example $\mathbf{x}_o$ from the one-shot class o to be confused with a similar class s, formally $P_s^o = P(\hat{c} = s|\mathbf{x}_o)$, we define s as the most likely class for the one-shot to be confused with. Given the collection of probabilities that an example of the one-shot class $\mathbf{x}_o$ is confused with other classes $\{P(\hat{c} = 1|\mathbf{x}_o), ..., P(\hat{c} = i|\mathbf{x}_N)\}$. We define P_s as



c_1
c_2
c_o
c_N
Training Set
1vsAll Binary SVMs
$f_1(\mathbf{x}) = \mathbf{w}_1\mathbf{x} + d_1$
$f_2(\mathbf{x}) = \mathbf{w}_2\mathbf{x} + d_2$
$f_o(\mathbf{x}) = \mathbf{w}_o\mathbf{x} + d_o$
$f_N(\mathbf{x}) = \mathbf{w}_N\mathbf{x} + d_N$
$f_{\hat{o}}(\mathbf{x}) = \sum_{i=1}^{N} [(f_i(\mathbf{x})+1)P_i] - P_s$
Fusion function
$F(\mathbf{x}_o)$
Multiclass Probabilistic SVM without 1-shot class
$P_1 = P(c=1 \mid \mathbf{x}_o)$
$P_2 = P(c=2 \mid \mathbf{x}_o)$
$P_N = P(c=N \mid \mathbf{x}_o)$
Probabilities of Misclassification

Figure 5.5: The overall dataflow of our method. The vertical rectangles represent column vectors of individual training examples and support vectors. Different colors correspond to different classes.

$$P_s = max(\{P(\hat{c}=1|\mathbf{x}_o), ..., P(\hat{c}=N|\mathbf{x}_o)\}). \tag{5.6}$$

We fix P_o^o, the probability of an example $\mathbf{x}_o$ belonging to the one-shot class being confused with the one-shot class itself $P(\hat{c}=0|\mathbf{x}_o) = 1$. The final multi-class decision function is given by

$$\hat{c} = \frac{argmax}{i}\{f_1(\hat{\mathbf{x}}), ..., f_{\hat{o}}(\hat{\mathbf{x}}), ..., f_N(\hat{\mathbf{x}})\}. \tag{5.7}$$

One possible way to obtain P_i^o is to use the information directly from the confusion matrix obtained via the *Naive* experiments described in section 5.2.1. To use the specific example of the KTH dataset, if we have currently selected boxing as the one-shot class o, the probability of *boxing* being confused with *clapping* $P^{boxing}_{clapping} = K_1(boxing, clapping) = P^1_2 = K_1(1,2) = 0.96$, while the probability of boxing being confused with *waving* and *walking* is $P^{boxing}_{waving} = K_1(boxing, waving) = P^1_3 = K_1(1,3) = 0.02$ and $P^{boxing}_{walking} = K_1(boxing, walking) = P^1_6 = K_1(1,6) = 0.02$, respectively. The probability of confusing boxing with *jogging* and *running* is 0. The highest probability of misclassification is with *clapping*, at 0.96. Hence, $P^o_s = P^{boxing}_{clapping} = 0.96$.

While the above approach for estimating the confusion probabilities works quite well, it is not very practical for realistic scenarios since it essentially trains on the testing data. At the very least, it assumes the existence of a validation set for the one-shot class on which the experiments can be run and P^o_i can be estimated, which in the case of one-shot learning should not be available. Instead, we can estimate the P^o_i as follows: we take the training examples for the classes that have many examples, excluding the one shot R^j_i $i \neq o$, and train a probabilistic multi-class Support Vector Machine F. Given an unknown testing example $\hat{\mathbf{x}}$, the output of this SVM is a set of probabilities of the unknown example belonging to the $N-1$ classes, excluding the one-shot class.

$$F(\hat{\mathbf{x}}) = \{P(\hat{c} = i|\hat{\mathbf{x}})\}, \tag{5.8}$$

where $i \neq o$. When we input the one example $\mathbf{x}_o$ from the one-shot class into the SVM F, we get the similarity of the one-shot class to the other multi-shot classes, this provides a good estimate of the probability of misclassifying the one-shot class as each one of the multi-shot classes. Thus we set

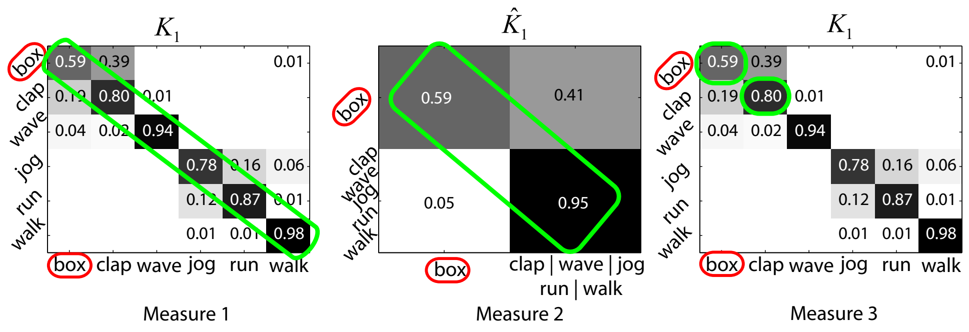


Figure 5.6: This figure demonstrates how the three performance measures are computed from the confusion matrix. The confusion matrices displayed here are for the KTH dataset and were obtained using our proposed fusion method. In the figure, boxing, circled in red, was treated as the one-shot class. The green circles show the values that we average to compute the corresponding measures.

$$P_i^o = F(\mathbf{x}_o). \tag{5.9}$$

5.3 Experiments

We perform experiments on 4 datasets. Three of these datasets are standard action recognition datasets KTH, UCF11, and UCF50. The fourth dataset is the UCFARG dataset which is an aerial action recognition dataset. KTH (see Figure 5.7) contains 6 classes, approximately 100 examples per class, and 1000 dimensions per example. UCF11 (see Figure 5.8) is a much more unconstrained dataset obtained from Youtube consisting of 11 actions approximately 100 to 150 examples per action around 4000 dimensions per example. UCF 50 (see Figure 5.9) is an expanded Youtube dataset consisting of 50 actions with approximately 100 to 150 examples per action 2000 dimensions per example. For our experiments we used a subset of the UCFARG dataset consisting of 6 actions, 48 examples per action, 1000 dimensions per example.

The bag of words framework was designed to work best when the camera is static, however since UCFARG is an aerial dataset there is a lot of camera motion. In order to overcome this, we first perform registration to compensate the camera motion, next we assume that rough motion detection and tracking have been done, and crop a small region around the person performing the action and process that as our clip of interest.

In order to evaluate the performance we use two baselines and three different measures. The baselines have been described in section 5.2.1 they are *Naive One-Shot* which provides the lower bound on performance that can be expected, and *Multi-Shot* which is the standard training and testing frame work which uses a large number of training examples and provides an upper bound on performance. We need to get as close to this level as possible.

The three measures that we use are defined as follows (See Figure 5.6 for reference). As described in section Figure 5.4 we select a class to be the one-shot class o, and then perform many possible splits of the data performing training and testing for each while pooling the results from different splits into the same confusion matrix which is then normalized row by row, giving us the confusion matrix K_o. The confusion matrix K_1 for the boxing action class can be seen on the left in Figure 5.6. Measure $D1_i$ for the experiment when class i was selected as the one-shot class is just the average accuracy of the diagonal of the confusion matrix K_i.

$$D1_i = \frac{1}{N} \sum_{j=1}^{N} K_i(j,j) \tag{5.10}$$

This measure measures the level of confusion between the one-shot and multi-shot classes, as well as the level of confusion of the multi-shot classes among themselves. The shortcoming of this measure is that when as the number of multi-shot classes increases, the

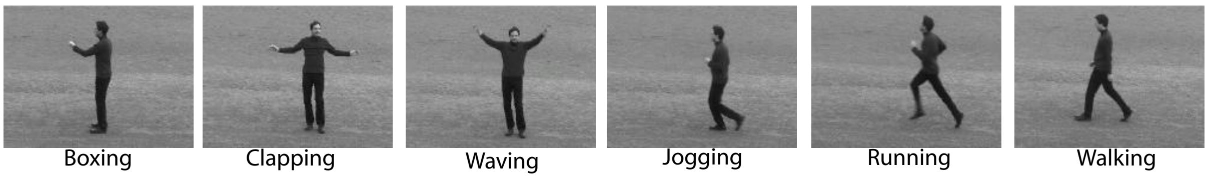


Figure 5.7: This figure shows examples of actions from the KTH datset.

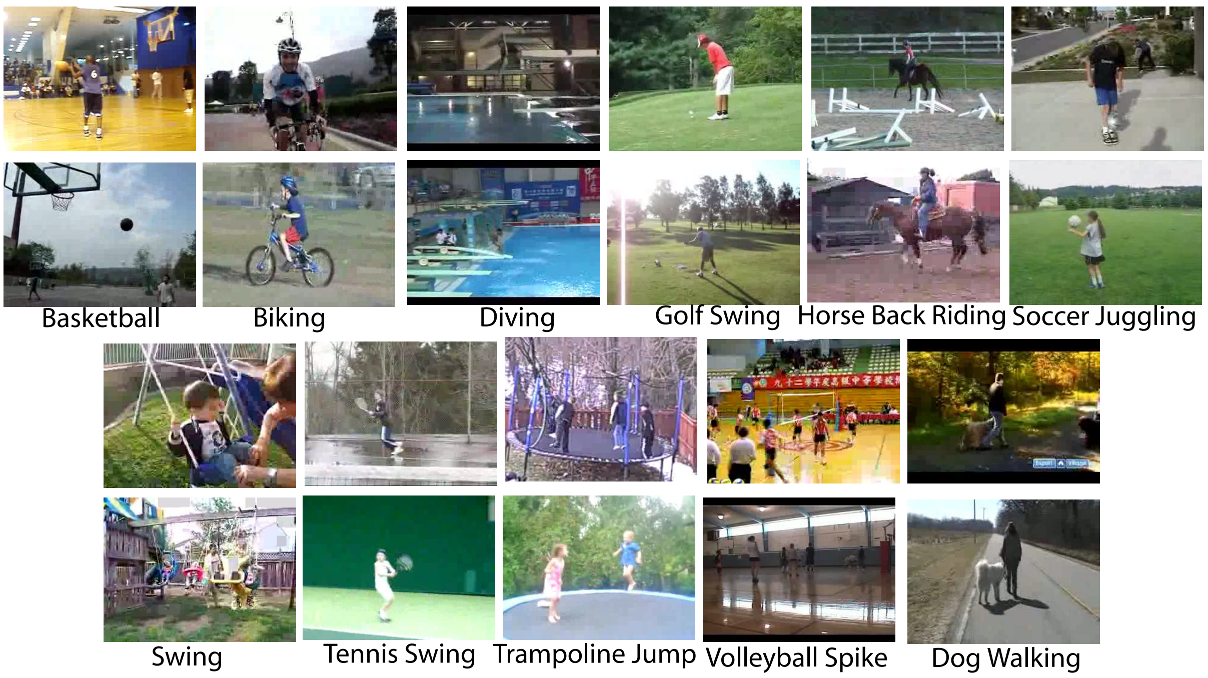


Figure 5.8: This figure shows examples of actions from the UCF11 dataset.

measure becomes dominated by the multi-shot classes and differences in performance between different methods becomes difficult to notice. Because of this we also introduce measures 2 and 3 which are more sensitive to the accuracy of the one-shot class.

To compute Measure2 we first convert the $N \times N$ confusion matrix K_i, into a 2×2 confusion matrix $\hat{K}_i$. The confusion matrix $\hat{K}_i$ only captures the level of confusion of the one-shot class with multi-shot classes and vice versa, while the confusion between the multi-shot classes is ignored. $D2_i$ for the experiment when class i was selected as the one-shot class is just the average accuracy of the diagonal of the confusion matrix $\hat{K}_i$.

$$D2_i = \frac{1}{2}\sum_{j=1}^{2}\hat{K}_i(j,j) \tag{5.11}$$

The danger of using the second measure is that it is possible to achieve an improvement in performance by misclassifying one of the classes as the one-shot class. The class that is most likely to be misclassified as the one-shot class is the similar class s. To make sure that this is not happening, we compute the third measure $D3_i$ that measures the level of confusion between the one-shot class o and the similar class s.

$$D3_i = \frac{1}{2}(K_i(o,o) + K_i(s,s)) \tag{5.12}$$

Since different classes can be selected as the one-shot class, we can compute the average of each measures across all of the classes in the dataset. The average measures are simply computed as

$$\bar{D}1 = \frac{1}{N}\sum_{i=1}^{N} D1_i \quad \bar{D}2 = \frac{1}{N}\sum_{i=1}^{N} D2_i \quad \bar{D}3 = \frac{1}{N}\sum_{i=1}^{N} D3_i. \tag{5.13}$$

The results of our method are summarized in Table 5.1. As can be seen from the table, there is a clear disparity in the performance between the *Naive one-shot* and the *Multi-Shot* methods. While our proposed method improves all three measures for all of the four datasets. Note that the gain in performance for avgMeasure1 may seem small for UCF50, but this as we pointed out previously is due to the fact that Measure1 captures the confusion among the multi-shot examples as well, and since UCF has a lot more examples than KTH, avgMeasure1

Table 5.1: This table shows the average measures for all four datasets that we have tested. Naive is the lower bound for performance, and Multi is the upper bound. The performance of our method falls between the two.

	AvgMeasure1	**AvgMeasure2**	**AvgMeasure3**
KTH Naive	74.79%	50.12%	47.45%
KTH Fusion	82.20%	74.97%	66.61%
KTH Multi	90.13%	91.85%	86.71%
UCF11 Naive	77.19%	50.49%	39.46%
UCF11 Fusion	78.33%	68.78%	55.13%
UCF11 Multi	83.95%	90.63%	77.87%
UCF50 Naive	76.14%	50.30%	39.05%
UCF50 Fusion	76.32%	62.52%	50.43%
UCF50 Multi	77.48%	88.23%	74.85%
UCFARG Naive	55.12%	50.01%	37.79%
UCFARG Fusion	56.77%	55.22%	40.09%
UCFARG Multi	57.52%	73.96%	51.39%

is dominated by the multi-shot classes. Hence for UCF50 it is more meaningful to examine measures avgMeasure2 and avgMeasure3.

5.4 Summary

In summary we have developed a framework for improving classification accuracy for when dealing the the problem of having few examples available for training. The key idea is that we want to utilize information from classifiers that were trained for classes that have many examples, to aid the classifier that was trained using only one positive example. We do this

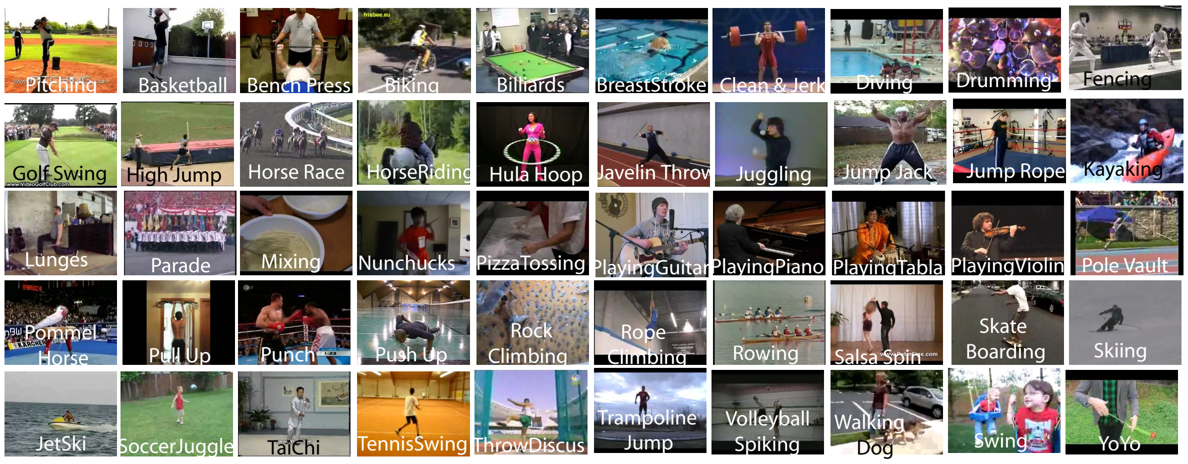


Figure 5.9: This figure shows examples of actions from the UCF50 dataset.

via late fusion of all classifier decision values, where the weights in the fusion are estimated automatically and correspond to the probability of the one-shot class being misclassified.

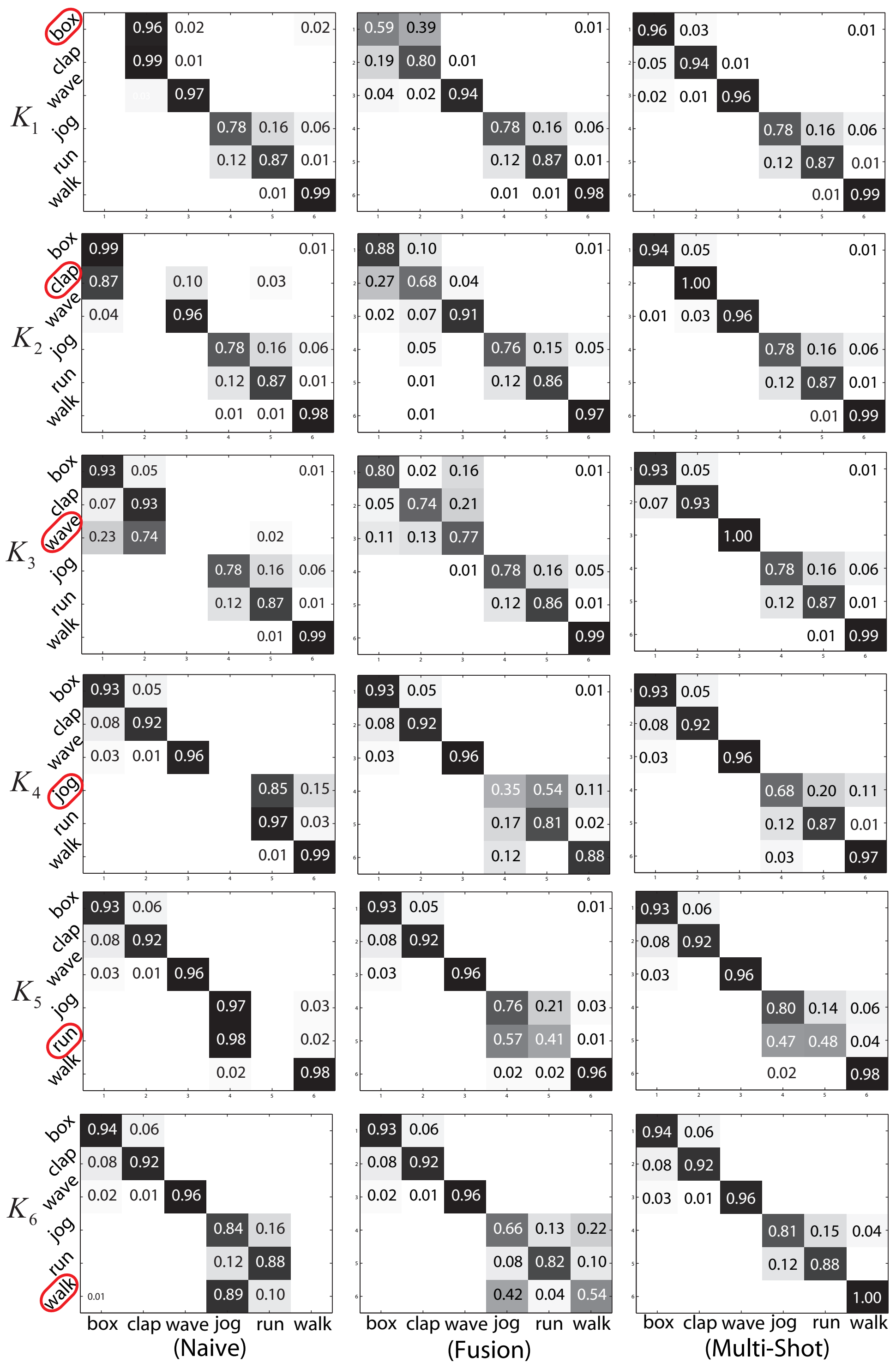


Figure 5.10: This figure shows confusion matrices for the KTH dataset for our proposed fusion method (center column), and the two baseline methods (left and right columns). The action circled in red is the action with only one example (class o). In the case of multi-shot (right column), we select many examples around the one-shot example.

CHAPTER 6: CONCLUSION

In this thesis we have addressed the problems of detecting, tracking, and recognizing activities of targets of interest in aerial imagery under a number of challenging conditions. These conditions include fast camera motion, low frame-rate parallax, strong shadows, fast moving targets, dense targets, and insufficient number of action examples.

The first problem we addressed was the detection of moving and stationary objects in presence of fast camera motion, low frame-rate, parallax, and shadows. We compensated camera motion using interest points and homographic framework, detected moving objects using median background subtraction, and removed false parallax detections using the gradient of the background image. To detect stationary humans and vehicles we utilized a set of geometric constraints in order to detect blobs that are oriented in the directions of ground-plane normal and shadow. We then combined these blobs into a set of shadow casting out of plane object candidates, with which we primed static object detectors in order to constrain their search-space.

The second problem we addressed was the tracking of large numbers of dense and fast moving objects in low frame-rate video. In order to manage the large complexity of the tracking problem, we split the scene into overlapping grid cells, solved the tracking using bipartite graph matching, and linked the tracks across cells. When solving the tracking problem within each cell, we could not rely on appearance due to low resolution and single channel data. Instead we used the cells to derive a set of scene constraints directly from the data, in order to help the tracker with initial assignments.

The third problem we addressed was that of action recognition in Aerial Video in the challenging case of having few examples. Here we utilized transfer learning to transfer information from the classes that had many examples to the classes that had few examples. We did this via late fusion of weighted SVM decision functions of classes that are similar to the class with few examples.

6.1 Summary of Contributions

Our main contributions are summarized below.

1. Object Detection In Aerial Video
   (a) A new approach to detecting moving objects in presence of parallax.
   (b) A new approach to detecting stationary objects in presence of shadow.
   (c) A new approach to detecting shadows of objects in aerial video.
   (d) A new approach to estimating the orientation of the sun in aerial video.
2. Object Tracking in Aerial Video
   (a) Handling large number of targets.
   (b) Handling large target density.
   (c) Deriving scene constraints from the imagery.
3. Action recognition from few Examples in Aerial Video.

(a) A new method of improving classification accuracy in one-shot action recognition framework.

(b) A new way of performing classifier fusion.

(c) A new method for estimating the weights needed for the late fusion of classifier outputs.

6.2 Future Work

In this section we explore some of the possible improvements, extensions, and directions that can be explored.

The obvious direction to take the stationary object detection framework, is to further expand the classes of objects that can be detected with the help of their shadows and the relationship between the shadows and the objects that are casting them. More complex objects can include various types of aircraft on the runways, construction equipment, and buildings. However since these objects, their shadows, and the relationship between them are more complex than humans and vehicles, it would require reworking of the rule-based detection of candidates into a softer probabilistic framework, to allow for misdetections of shadow and object parts. This work can also be extended to aid in a more general semantic analysis of the scene, such as aiding in the detection of large shadow regions, being cast by multiple large overlapping scene structures.

In the case of tracking large number of targets, additional scene constraints need to be explored for better assignment disambiguation. Additional constraints can be derived through better analysis of the environment in the scene. This can include detecting bridges and overpasses to allow for a more explicit target reacquisition handling as opposed to leaving it up to the graph structure, detecting traffic markers such as zebra crossings, stop sign lines, and roundabouts. These additional constraints can either be made part of the kinematic model, or to generate additional weights to be made part of the graphical structure.

The framework for recognizing actions from few examples can be extended into other representations, and other domains such as object recognition or event detection. Additionally improving the representation of actions themselves should also be explored.

LIST OF REFERENCES

[1] Saad Ali and Mubarak Shah. Cocoa: tracking in aerial imagery. volume 6209. SPIE, 2006.

[2] C. Arth, C. Leistner, and H. Bischof. Object reacquisition and tracking in large-scale smart camera networks. In *ICDSC*, 2007.

[3] Y. Aytar and A. Zisserman. Tabula rasa: Model transfer for object category detection. *ICCV*, 2011.

[4] E. Bart and S. Ulman. Cross-generalization: learning novel classes from a single example by feature replacement. *CVPR*, 2005.

[5] M. Betke, D. E. Hirsh, A. Bagchi, N. I. Hristov, N. C. Makris, and T. H. Kunz. Tracking large variable numbers of objects in clutter. In *CVPR*, 2007.

[6] S. Bi, D. Liang, X. Shen, and Q. Wang. Human cast shadow elimination method bad on orientation information measures. *ICAL*, 2007.

[7] A. Bissacco and M-H Yang. Detecting humans via their pose. *NIPS*, 2007.

[8] T. Breckon, S. Barnes, M. Eichner, and K. Wahren. Autonomous real-time vehicle detection from a medium-level uav. *UAVS*, 2009.

[9] C. Cedras and M. Shah. Motion-based recognition: A survey. *Image and Vision Computing*, 1995.

[10] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[11] J.-C. Chang, W.-F. Hu, J.-W. Hsieh, and Y.-S. Chen. Shadow elimination for effective moving object detection with gaussian models. *ICPR*, 2002.

[12] Y. T. Chen, C. S. Chen, Y. P. Hung, and K. Y. Chang. Multi-class multi-instance boosting for part-based human detection. *ICCV*, 2009.

[13] Hui Cheng, D. Butler, and C. Basu. ViTex: Video to tex and its application in aerial video surveillance. *CVPR*, 2006.

[14] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *CVPR*, 1, June 2005.

[15] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. *PETS*, 2005.

[16] H. Farid. Blind inverse gamma correction. *IEEE Trans. Image Processing*, 2001.

[17] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *PAMI*, 2006.

[18] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. *CVPR*, June 2008.

[19] M. Fink. Object classification from a single example utilizing class relevance metrics. *NIPS*, 2004.

[20] G.D. Finlayson, S.D. Hordley, Cheng Lu, and M.S. Drew. On the removal of shadows from images. *IEEE PAMI*, 28(1), Jan. 2006.

[21] N. Friedman and S. Russel. Image segmentation in video sequences: A probabilistic approach. *UAI*, 2000.

[22] A. Gaszczak, T. Breckon, and J. Han. Real-time people and vehicle detection from uav imagery. *SPIE*, 2011.

[23] G. Gualdi, A. Prati, and R. Cucchiara. Multi-stage sampling with boosting cascades for pedestrian detection in images and videos. *ECCV*, 2010.

[24] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[25] E. Hayman and J. Eklundh. Statistical background subtraction for a mobile observer. *ICCV*, 2003.

[26] J.-W. Hsieh, S.-H. Yu, Chen Y.-S., and W.-F. Hu. A shadow elimination vethod for vehicle analysis. *ICPR*, 2004.

[27] H. Hu, Y.-Q. Huang, and L.-M. Li. Moving vehicle shadow elimination approach based on mark growing of multi-feature fusion. *ICACIA*, 2010.

[28] C. Hui, D. Butler, and C. Basu. ViTex: Video to tex and its application in aerial video surveillance. *CVPR*, 2006.

[29] M. Irani and P Anandan. A unified approach to moving object detection in 2d and 3d scenes. In *PAMI*, volume 20, 1998.

[30] R. Jain and H Nagel. On the analysis of accumulative difference pictures from image sequences of real world scenes. *PAMI*, 1979.

[31] Omar Javed, Khurram Shafique, and Mubarak Shah. Appearance modeling for tracking in multiple non-overlapping cameras. In *CVPR*, 2005.

[32] Jinman Kang, Isaac Cohen, and Chang Yuan. Detection and tracking of moving objects from a moving platform in presence of strong parallax. In *ICCV*, 2005.

[33] Robert Kaucic, Amitha Perera, Glen Brooksby, John Kaufhold, and Anthony Hoogs. A unified framework for tracking through occlusions and across sensor gaps. In *CVPR*, 2005.

[34] A. Kembhavi, D. Harwood, and L.S. Davis. Vehicle detection using partial least squares. *IEEE PAMI*, 2011.

[35] S. Kluckner, T. Mauthner, P. M. Roth, and H Bischof. Semantic classification in aerial imagery by integrating appearance and height information. *ACCV*, 2009.

[36] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum. One shot learning of simple visual concepts. *AMCSS*, 2011.

[37] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. *CVPR*, 2008.

[38] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. *CVPR*, 2005.

[39] B. Li, B. Yuan, Y. Sun, and C. Wan. A novel shadow elimination algorithm for moving object segmentation. *ICWMMN*, 2006.

[40] Z. Lin and L. S. Davis. Shape-based human detection and segmentation via hierarchical part-template matching. *IEEE TPAMI*, 32, 2010.

[41] J. Liu, S. Ali, and M. Shah. Recognizing human actions using multiple features. *CVPR*, 2008.

[42] J. Liu, B. Kuipers, and S. Savarese. Recognizing human actions by attributes. *CVPR*, 2011.

[43] J. Liu, M. Shah, B. Kuipers, and S. Savareset. Cross-view action recognition via view knowledge transfer. *CVPR*, 2011.

[44] Z. Liu, F. Zhao, and H. Yang. A new method of moving shadow elimination combining texture and chrominance of moving foreground region bsed on criterion. *WCICA*, 2010.

[45] N. Martel-Brisson and A. Zaccarin. Moving cast shadow detection from a gaussian mixtrue shadow model. *CVPR*, 2005.

[46] A. Middal and N. Paragios. Motion-based background subtraction using adaptive kernel density estimation. *CVPR*, 2004.

[47] K. Mikolajczyk, Schmid C., and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. *ECCV*, 2004.

[48] A. Miller, P. Babenko, M. Hu, and M. Shah. Person tracking in UAV video. *CLEAR*, 2007.

[49] J. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *BMVC*, 2006.

[50] A. Panagopoulos, D. Samaras, and N. Paragios. Robust shadow and illumination estimation using a mixture model. *CVPR*, 2009.

[51] A.G.A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and W.S. Hu. Multi-object tracking through simultaneous long occlusions and split-merge conditions. In *CVPR*, 2006.

[52] R. Pflugfelder and H. Bischof. Tracking across non-overlapping views via geometry. In *ICPR*, 2008.

[53] F. Porikli and J. Thornton. Shadow flow: A recursive method to learn moving cast shadows. *ICCV*, 2005.

[54] A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara. Detecting moving shadows: Algorithms and evaluation. *IEEE TPAMI*, 25, 2003.

[55] M. Quaritsch, K. Kruggl, D. Wischounig-Strucl, S. Bhattacharya, M. Shah, and B. Rinner. Networked uavs as aerial sensor network for disaster management applications. *Elektrotechnik und Informationstechnik*, (127), 2010.

[56] I. Reda and A. Anreas. Solar position algorithm for solar radiation applications. *NREL Report No. TP-560-34302*, 2003.

[57] M. Rodriguez, A. Ahmed, and M. Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. *CVPR*, 2008.

[58] P. Rudol and P Doherty. Human body detection and geolocalization for uav search and rescue missions using color and thermal imagery. *IEEE Aerospace*, 2008.

[59] P. Sabzmeydani and G. Mori. Detecting pedestrians by learning shapelet features. *CVPR*, June 2007.

[60] I. Saleemi, L. Hartung, and M. Shah. Scene understanding by statistical modeling of motion patterns. *Image and Vision Computing*, 1995.

[61] S. H. Sawhney, Y. Guo, and R. Kumar. Independent motion detection in 3d scenes. In *PAMI*, volume 22, 2000.

[62] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. *ICPR*, 2004.

[63] H.J. Seo and P. Milanfar. Action recognition from one example. *PAMI*, 2010.

[64] J. Sokalski and T Breckon. Automatic salient object detection in uav imagery. *UAVS*, 2010.

[65] Chris Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. *CVPR*, 1999.

[66] K. Tang, M. Tappen, R. Sukthankar, and C. Lampert. Optimizing one-shot recognition with micro-set learning. *CVPR*, 2010.

[67] T-P. Tian and S. Sclaroff. Fast multi-aspect 2d human detection. *ECCV*, 2010.

[68] O. Tuzel, F. Porikli, and P. Meer. Pedestrian detection via classification on riemannian manifolds. *PAMI*, 30, 2008.

[69] X. Wang, T. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. *ICCV*, 2009.

[70] B. Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. *ICCV*, 2005.

[71] Qi. Wu, X. Luo, H. Li, and P. Liu. An improved multi-scale retinex algorithm for vehicle shadow elimination based on variational kimmel. *SWUATC*, 2010.

[72] S. Wu, O. Oreifej, and M Shah. Action recognition in videos acquired by a moving camera using motion decomposition of lagrangian particle trajectories. *ICCV*, 2011.

[73] S. Wu, O. Oreifej, and M. Shah. Action recognition in videos acquired by a moving camera using motion decomposition of lagrangian particle trajectories. *ICCV*, 2011.

[74] Zheng Wu, Nickolay I. Hristov, Tyson L. Hedrick, Thomas H. Kunz, and Margrit Betke. Tracking a large number of objects from multiple views. In *ICCV*, 2009.

[75] Jiangjian Xiao, Hui Cheng, Feng Han, and H. Sawhney. Geo-spatial aerial video processing for scene understanding and object tracking. *CVPR*, June 2008.

[76] Jiangjian Xiao, Hui Cheng, Harpreet Sawhney, and Feng Han. Vehicle detection and tracking in wide field-of-view aerial video. *CVPR*, 2010.

[77] Jiangjian Xiao, C. Yang, F. Han, and H. Cheng. Vehicle and person tracking in aerial videos. *Multimodal Technologies for Perception of Humans*, 2008.

[78] J.J. Xiao, H. Cheng, F. Han, and H.S. Sawhney. Geo-spatial aerial video processing for scene understanding and object tracking. In *CVPR*, 2008.

[79] Li Xu, Feihu Qi, and Renjie Jiang. Shadow removal from a single image. *Intelligent Systems Design and Applications*, 2, Oct. 2006.

[80] S. Yahyanejad, D. Wischounig-Strucl, M. Quaritsch, and Rinner B. Incremental mosaicking of images from autonomous, small-scale uavs. *AVSS*, 2010.

[81] J. Yang, R. Yan, and A. Hauptmann. Cross-domain video concept detection using adaptive svms. *ACM MM*, 2007.

[82] W. Yang, Y. Wang, and G. More. Human action recognition from a single clip per action. *ICCV*, 2009.

[83] A. Yilmaz, O. Javed, and M. Shah. Object tracking a survey. *ACM Comput. Surv*, 38, 2006.

[84] A. Yoneyama, C. H. Yeh, and C.-C. Jay Kuo. Moving cast shadow elimination for robust vehicle extraction based on 2d joint vehicle/shadow models. *AVSS*, 2003.

[85] J. Zhong and Sclaroff S. Segmenting foreground objects from a dynamic textured background via a robust kalman filter. *ICCV*, 2003.