Preprints of the 18th IFAC World Congress
Milano (Italy) August 28 - September 2, 2011

Mobile Object Detection from an Embedded camera: how to minimize the latency time?

Michel Devy [*] Dora Luz Almanza Ojeda [*] Ariane Herbulot [*]

[*] *CNRS; LAAS; 7 avenue du Colonel Roche, F-31077 Toulouse, France Université de Toulouse; UPS, INSA, INP, ISAE; LAAS; Toulouse, France (e-mail: michel, dlalmanz, aherbulo@laas.fr).*

Abstract: This paper presents a global approach devoted to the detection of Mobile Objects from images acquired by a single camera embedded on a vehicle or a robot. The general method is presented involving at first optical flow extraction from tracked interest points, then clustering between static points belonging to the background and mobile ones extracted on different mobile objects, and then the tracking procedure performed for every detected mobile object. First results obtained from a sequential implementation, have shown that the latency time was too important. Here two improvements are proposed: first, a probabilistic map is built in order to detect points in more probable regions where a mobile object could be found. Then sequential and parallel architectures proposed to integrate point detection, point tracking and point clustering, are analyzed with respect to the latency time. Finally experimental results are presented and discussed.

Keywords: intelligent vehicles, obstacle detection, vision.

1. INTRODUCTION

This paper presents a visual-based approach for the classical Mobile Obstacle Detection and Tracking problem. Such a module is required on a vehicle which must navigate in cluttered and dynamical environments, typically on an autonomous service robot, or on a car manually driven. In these two contexts, many research works have been devoted to detect obstacles, to estimate their states (position, speed...), to identify their nature (other robot or vehicle, cycle, pedestrian, pet…) and to prevent from collisions: here only the detection and tracking functions are considered.

This problem has been studied using different sensors: laser, radar, vision…Many works for driver assistance concern laser-based obstacle detection and tracking Vu and Aycard (2009). In order to track obstacles, it is required to estimate from the same sensory data, the egomotion of the vehicle: a global probabilistic framework known as SLAMMOT has been proposed in Wang et al. (2007). SLAMMOT combines SLAM (Simultaneous Localization and Mapping) with MOT (Mobile Object Tracking). Some works have made more robust the Object Detection function, from the fusion with monovision Gate et al. (2009), or with stereovision using the v-disparity concept Labayrade et al. (2005). In Alenya et al. (2009), a biologically-inspired monocular approach is proposed estimating the Time To Contact (TTC) from the rate of change of size of features. Nevertheless, in spite of numerous contributions, Mobile Obstacle Detection and Tracking still remains a challenge when it is based only on one embedded camera. This paper tackle this challenge, trying especially to minimize the latency time, i.e. the time required by the system to detect the obstacle after it enters the view field of the camera. As it has been proved in numerous works Davison (2003); Sola et al. (2005); Civera et al. (2008), 2D information is sufficient in order to estimate the robot motion using a Visual SLAM algorithm, based on static points. This paper proposes a strategy in order to detect these static points, and moreover to detect and cluster the moving ones in order to detect and to track mobile objects: it is the first step towards a Visual SLAMMOT approach.

There are various methods for feature detection and tracking using a single mobile camera. In Davison (2003); Sola et al. (2005), new 2D points are extracted from every image by the Harris detector in selected regions of interest; for every one, an appearance model is memorized, and a new landmark is added in the stochastic map, using a gaussian mixture Sola et al. (2005) or the Inverse Depth Parametrization Civera et al. (2008). Then, from every image, an *Active Search* strategy is applied in order to find 2D points matched with landmarks of the map: a point is searched using its appearance model, only in a predicted area, computed from the back-projection of a landmark.

This approach could fail if some points added in the map, are mobile, because extracted on dynamic objects. The stochatic map can be extended with a speed for every point added in the map, so that mobile points could be detected and discarded from a probabilistic test. This method works fine, but adds to many computations and does not allow

[⋆] This work is supported by the Agence Nationale de la Recherche (ANR - the French national research agency) (ANR Rinavec PsiRob 2007 – ANR-07-ROBO-0006). D.L.Almanza Ojeda has been supported by the scholarship 183739 of CONACYT, SEP and by the mexican government to characterize an image region as a moving obstacle. Another method widely used for robotics applications is based on the optical flow, extracted from interest points, typically extracted by the Harris detector and tracked by the KLT algorithm Shi and Tomasi (1994); points are tracked in the image space, estimating only their apparent motions in order to make more robust the function.

Clustering techniques allow to partition the set of tracked points in subsets that correspond to the static scene background, and to every detected moving 3D object. In Veit et al. (2007) a clustering algorithm based on the *a contrario* method Desolneux et al. (2008) has been validated. It has been also used in Poon et al. (2009) to detect moving objects in short sequences; additionally, a 3D reconstruction is performed on tracked points to better associate points on moving objects; experimental results are only based on images acquired from fixed cameras.

This paper proposes a visual module integrating several functions (KLT tracker, a contrario clustering and object tracker based on Extended Kalman Filtering or EKF-based), that continuously detects moving objects from a mobile robot navigating in an outdoor open environment. In our initial method presented in Almanza-Ojeda et al. (2010), the latency time was not considered as a fundamental evaluation criterion for this application; it is shown here how to reduce this latency time thanks to an active method to monitor continuously the full view field, and thanks to a parallel implementation of the method.

Outline: section 2 describes the complete system, including SLAM and MOT modules not taken into account in this paper. Then section 3 is focused on the active point detection method. Sequential and parallel architectures are evaluated in section 4 with respect to the latency time. Section 5 presents some experimental results. Finally, section 6 summarizes our contribution and presents on-going works.

2. OVERALL STRATEGY FOR MOBILE OBJECT DETECTION AND TRACKING

Figure 1 shows the relationships between all functions and data structures involved in our visual system, devoted both to visual SLAM and to Mobile Object Detection and Tracking. The grey functions or data will not be considered in this paper. Moreover other functions required in an efficient obstacle detection system are not represented here, like for example Object identification or the construction of a robot-centered occupancy grid.

Many authors Gate et al. (2009); Wang et al. (2007) have identified cross-correlations between these functions. For example, detection depends on the camera localization because the optical flow in an image sequence is created both by the egomotion of the camera (i.e. of the robot on which it is mounted) and by the motion of other objects. But the visual-based robot localization in an unknown or partially known environment depends on the construction of a stochastic map with only static points, so on the clustering process able to discriminate static and dynamic points.

Point Detection

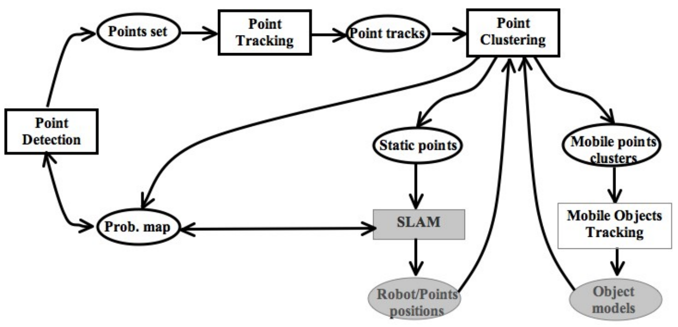


Fig. 1. The Mobile Object detection and Tracking system

The efficiency of our method relies on points detected initially by the first function presented on the left on figure 1. The KLT tracker will be applied on a fixed number of points $Npts$, (typically $Npts = 150$). The classical method extracts points on every image in order to replace the lost ones. Here point detection and point tracking are separated. The *Point Detection* function must profit from all knowledge already found on previous images, or from contextual or application-dependant knowledge. So every time points have to be selected, it exploits a probabilistic map, built from a discretization of the image space; this map gives the probability on every image cell, to find a dynamic point. Initially this probability is uniform; then it is updated according to several criteria discussed in 3.

New points are detected in cells which have the higher probability; initially these points are uniformly distributed in the image. Then as one goes along the method, points are replaced once they have been either lost by the point tracker, or identified as static, while dynamic points are kept in the points set and tracked continuously.

Point Tracking

The points set is used by the *Point Tracking* function, which applies the classical Shi-Tomasi algorithm Shi and Tomasi (1994) in order to build point trails during Nim successive images. Some points are lost, or are not tracked during Nim iterations; only N complete tracks will be considered in the next step.

Point Clustering

This function must identify which points belong to the background, and which ones belong to dynamic objects. It is the critical step: the *a contrario* clustering technique identifies one group as meaningful if all their elements show a different distribution than an established background random model. Contrary to most of clustering techniques, neither initial number of clusters is required nor parameters have to be tuned. In unknown environment context these characteristics results are very favorable. Our implementation of this method is described in Almanza-Ojeda et al. (2010).

This function generates a list of static points and a list $\mathcal{O}$ of M clusters or potential objects, each one defined by a list of points and by characteristics: the cluster barycenter and the mean points velocity. It takes as inputs, information coming from the *SLAM* function, in order to compensate the camera egomotion when possible (i.e. on the ground plane). It also takes as inputs, information coming from the *MOT* function, required for two actions:

- decide when a potential object must be fused with objects already detected and tracked by the *MOT* function. The apparent motion estimated for each tracked objet is compared with the characteristics of each potential object for establishing a decision rule for merging.
- update the probability map, taking into account the objects characteristics when evaluating the probability to find mobile points on every cell.

Visual SLAM

Static points are sent to the SLAM function Davison (2003); Civera et al. (2008). At every iteration, this function adds new visual landmarks (here only 3D points) in the map. It exploits the same probability map so that observations of landmarks of the map are uniformly distributed in the current image, are not close to dynamic objects and are selected in cells that will remain visible for a long time, knowing the robot dynamics. Only short memory SLAM is applied: landmarks are removed from the map as soon as they are behind the robot.

This function has its own way to track points, based on the *Active Search* strategy and on the landmark 3D positions estimated in the stochastic map. So static points are no more tracked by the *Point Tracking* function.

Mobile Object Tracking

Finally, clusters of mobile points are processed by this MOT function: at first, image-based tracking is done using KLT, but including a Kalman Filtering step applied only to the barycenter of the points cluster. It allows to make more consistent the points tracking, since it is assumed that these points belong to a same rigid object. Later the region of every mobile object will be segmented and tracked using the snake tracker presented in A.Marin-Hernandez et al. (2004); the snake will be initialized with the hull of the points cluster, and guided using color information extracted inside this hull.

3. ACTIVE POINT DETECTION.

Due to the fact that our detection function is based on a sparse optical flow, it is required to actively select points to be tracked, in regions where the occurrence of a mobile object is the more probable. To describe the behavior of this occurrence, a probability map acts as an occupancy grid, in the sense that higher probabilities represent occupied (mobile) zones and lower probabilities free (static) ones, where to monitor the arrival of new mobile points. These probabilities are given to the *Point Detection* function which seeks for locations of potential new points. As these points are tracked during *Nim* successive images, new mobile cells appear and the probability map must be updated.

As an example, Figure 2 shows positions of tracked points for an image of the sequence presented in section 5. Points labelled O have been classified as static; points labelled $+$ have been identified as dynamic. This grid is

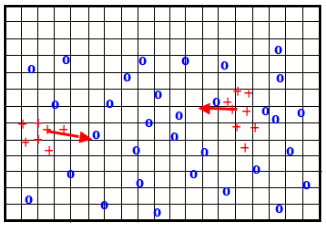


Fig. 2. Results of the Point Clustering function.

only used to illustrate the idea that the image could be totally divided in cells, but normally, cells are formed only around detected points, as it is explained hereafter. Cells on the image without points stay with a fair probability. Therefore, in this map three kind of cells are highlighted : fair, occupied and empty. If *a priori* context-based information are not taken into account, all locations have the same probability to contain dynamic points; all cells (u, v) in the probability map are initialized by $p(u, v, t = 0) = p_0$ where p_0 is a symbolic value that represents a fair cell. When a mobile point is detected in (u, v), it becomes the center of an occupied area of size rxr; reversely, empty cells are formed around static points. Probabilities are updated at every iteration in function of point motions: if a mobile point is tracked from (u, v) at time $t-1$ to (u', v') at time t, $p(u, v, t)$ will be lower than $p(u, v, t-1)$, and this cell (u, v) could be labeled as empty; $p(u', v', t)$ will be higher than $p(u', v', t-1)$, and the cell (u', v') becomes now an occupied cell.

The probability values in the map are modified by a bidimensional Gaussian function centered on (u, v), with σ_u and σ_v function of r. The eq. 1 gives the spatio-temporal evolution of probability into occupied, empty and fair cells at each current image. The probability that a cell belongs to a dynamic object, is inversely proportional to its distance to the closer detected mobile point; it decreases by the bell shape of the Gaussian function, and then temporally, by the inverse of the value established by the first case in the eq. 2.

$$p(u, v, t) = \sum_{u,v \in Cell} (\alpha(u, v)_{t-1} + \mathcal{G}(\mu_u, \mu_v, \sigma_u, \sigma_v)) \qquad (1)$$

where $\alpha(u, v)$ function describes the previous probability in the cell in function of the "label" of the cell, that is:

$$\alpha(u, v)_t = \begin{cases} p_{max} - p(u, v)_{t-1} & \text{if } Cell(u, v)_{t-1} \text{ occupied} \\ p_0 & \text{if } Cell(u, v)_{t-1} \text{ empty} \\ 0 & \text{if } Cell(u, v)_{t-1} \text{ fair} \end{cases} \qquad (2)$$

with $p_{max} = 1$.

By applying eq. 1 at each iteration, certain zones among cells of previous positions of the same point will be superposed. However, this still keeps the priority in the empty cells, normally behind the point in current image. On the left of Figure 3, this idea is depicted as a darkness zone which not only involves dynamical set of points (given by the Point Clustering) but also all the previous locations of these dynamical points. This results in an enlarged zone restricted by the occupied cells, but the rest of the zone is highly interesting to find new points in. Cells in which points have been detected as static at the end of *Nim* images, are set to the value of an occupied cell because we are mainly interested in mobile points. Both enlarged

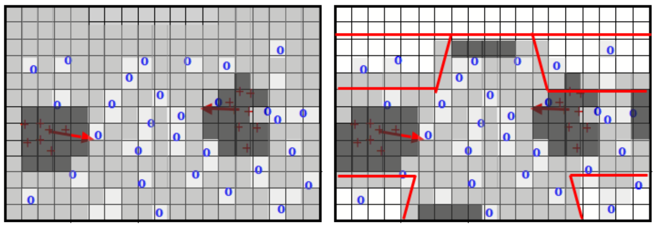


Fig. 3. (left) Updated probability map from the clustering result; (right) idem using simulated contextual knowledge.

zones and static point cell values are changed each $2Nim$ images, when the probability is reset to p_0 fair value and only current dynamic points are set as occupied cells. Figure 3 depicts on the right the application of a simulated context-dependant criterion, e.g. in a road context: red lines represent the road boundaries and the horizon line. It is not useful in such a situation to monitor zones above the horizon line: obstacles will arrive either from the horizon line for cars moving on the opposite way, or from the image bottom for an overtaking car.

Some other criteria, linked either to the robot dynamics or to contextual knowledge, will be integrated soon. Typically, if the robot turns on the left, it is useful to look for monitoring the left side of the image, so to increase the probabilities on the corresponding cells.

4. SEQUENTIAL VS PARALLEL ARCHITECTURE

An important property for a method in charge of detecting obstacles, is the latency time. Up-to-now only a software implementation has been evaluated: so all functions are executed in sequence. Only the *Point Detection*, *Point Tracking* and *Point Clustering* functions are considered in this analysis: they are noted respectively D, T and C in the following. The execution times of these functions are supposed bounded, and noted T_D, T_T and T_C.

The latency time will be expressed as N time units, selected as T_D the execution duration of the *Point Detection* process, i.e. the classical Shi-Tomasi algorithm, which is executed basically at 15Hz on our implementation. As it was explained in 2, the detection and tracking functions have been decoupled: the point detection is executed only when required in order to replace lost points or points detected as static by the *Point Clustering* function. It is considered here that $T_T = NimT_D$ and $T_C = nT_D$ ($Nim = 6$ and $n = 2$ for the two first timing charts).

Let us first analyze the sequential method; figure 4 presents how and when data structures are exchanged between the three processes D, T and C.

- At first D selects a set with $Npts$ points and sends it to T (dotted lines).
- T tracks these points on Nim successive images, and sends N point trails to C (solid lines): $Npts-N$ points have been lost.
- C analyzes these trails, updates the probability map and sends this map to D (dashed lines).
- A new iteration can begin... D selects $Npts$ minus the number of dynamic points identified by C ...etc ...

Only one process is active at a given time. Let us consider the worse situation: an obstacle appears just after the initial point detection. So it cannot be detected at the first iteration. If the active strategy to detect points is efficient, it will be detected at the second one. So the latency time is $2T_T + 2T_C + T_D$ ... here 17 time units. How to minimize this time? At first let us suppose that

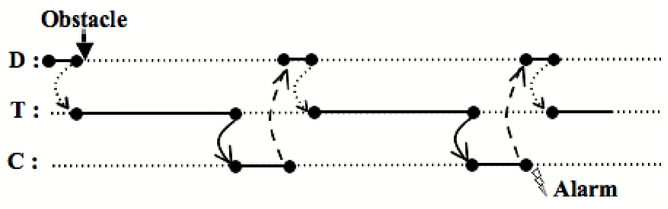


Fig. 4. Timing for the sequential architecture

several T processes can be executed in parallel, for example on a FPGA implementation, or using several processors. Figure 5 presents how it will work, considering that Nim is equal to $kT_{cluster}$ (here k=3).

- D select the first points set, and send it to a first T process. Then it waits during T_C and does again the same operation in order to give a second points set to a second T process. Again it waits during T_C and selects a third points set for a third T process. During these three first D activations, the probability map is not yet updated by C: so D updates itself the map so that the three selected points sets optimize the probability to find mobile objects.
- the three T processes are executed in parallel, with a shift of $T_C + T_D$; thanks to these shifts, they could send the point trails to only one C process, which could analyze these data in sequence.
- at every activation, C updates the probability map, and sends it to D, so that at its fourth activation, D can select new points around detected mobile objects ... and it can iterate.

In the worse situation, if an obstacle appears just after the first D activation, then it will be detected only after the second points selection, so after $T_T + 2T_C + T_D$... here 11 time units. The latency time can be only minimized,

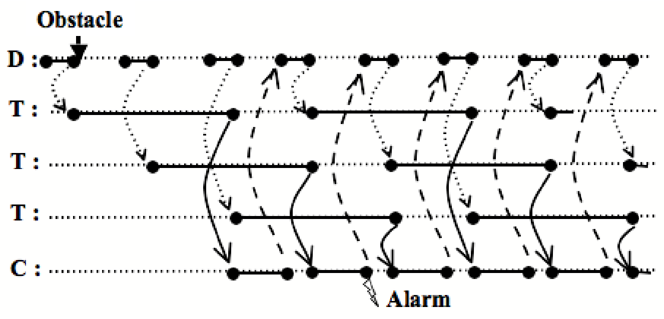


Fig. 5. Timing for a parallel architecture with only one clustering process

considering that the process D is always active; so it selects points sets at every period, and activates a T process. So more T processes will be activated in parallel, and it will be mandatory also to execute several C processes. Figure 6 presents the timing chart of such a parallel architecture:

exchanges of data structures have not been represented, but every points set has un number from 1 to 15 here: this number i is above a D execution, on the left of a T execution, and above a C execution. Vertical dotted lines have been added to make more visible when a process communicates its result to the next one. Here we have selected $T_T = 4T_D$ and $T_C = 2T_D$, only to improve the readability.

The number of parallel C processes depends on the ratio T_T/T_C, here equal to 2: the first one receives the point trails for points sets with an odd number, while the second one processes points sets with an even number. During the 8 first iterations of D, the probability map is only updated by D, without considering the C results, i.e. detected mobile objects. In the global system, it will be updated also by the $SLAM$ process, upgrading the probability to find an obstacle where the robot goes.

In this situation which requires 10 parallel executions, the latency time becomes minimal, i.e. $T_D+T_T+T_C$, assuming that points are selected on the obstacle image as soon as it appears.

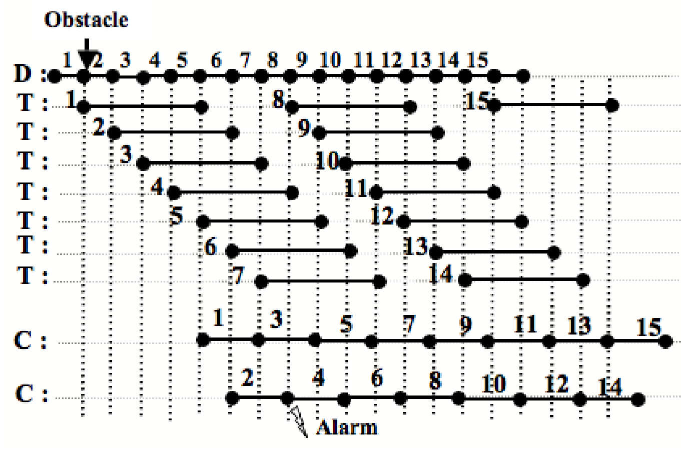


Fig. 6. Timing for a parallel architecture with several clustering processes

5. EXPERIMENTAL RESULTS

Robot navigation was performed on a parking with a camera mounted on the robot (640 × 480 at $10Hz$). The number $Npts$ of points given to the KLT tracker is set to 130, which is the better trade off between expected results and processing time. Figure 7 presents some images in which two moving objects go across the robot view field.

Image 7a shows the bounding box on a moving object (labeled O_1) which enters from the right side of the image; all detected points inside this box are used to initialize the object model. Two times of T_T later a second moving object O_2 is detected, on the left side of the image 7b. Also, this figure shows the current position of the object O_1 detected before, now tracked by the EKF-based object tracker. Object region could grow at each time of T_T when new clusters are detected, thanks to the cluster fusion mechanism. An example of this behavior is shown in figure 7c where we clearly see that the bounding box of O_2 was enlarged. It means that a third object was detected by the clustering process but after a merging test, it has been included in the O_2 model.

This first row of images shows the evolution of the environment during 20 images; along these images, O_1 and O_2 always move in frontal-parallel planes with respect to the image plane. However, middle row of figure 7 shows that the object O_2 performs a diagonal movement across the robot view field. It is an interesting situation because the method finds that some regions in the same object present important differences in their velocities and orientations.

This is illustrated in image 7d, where a new cluster cannot be included in a detected object because the points in O_2 (half lateral side of the car) and those extracted on the back of the car, have different displacements and consequently different velocities. So we initialize and track the third object independently of the others; two times of T_T later, merging is possible as it is illustrated on figure 7e. In the same time, the bounding box of O_1 is reduced because it is partially occluded by O_2. Also both images depict two detected objects in the ground caused by camera movement. These moving "objects" are detected normally in the lower part of the image (their egomotions are more noticeable) however, they are quickly filtered because they move out of image bounds. In figure 7f, O_1 is hidden.

Bottom row of images in figure 7 shows that O_1 is detected again. In other hand the bounding box of O_2 becomes smaller until it disappears and finally O_1 is totally detected and tracked until it disappears from the camera view field.

6. CONCLUSIONS AND FUTURE WORKS

This paper has presented a method for the Detection and the Tracking of Moving Obstacles from images acquired by an embedded camera. Such a method must be executed in real time, so that the latency time in the worst case could be minimized. It is shown that a parallel architecture is required in order to reduce this latency time at its minimum, executing in parallel functions for points detection, tracking and clustering.

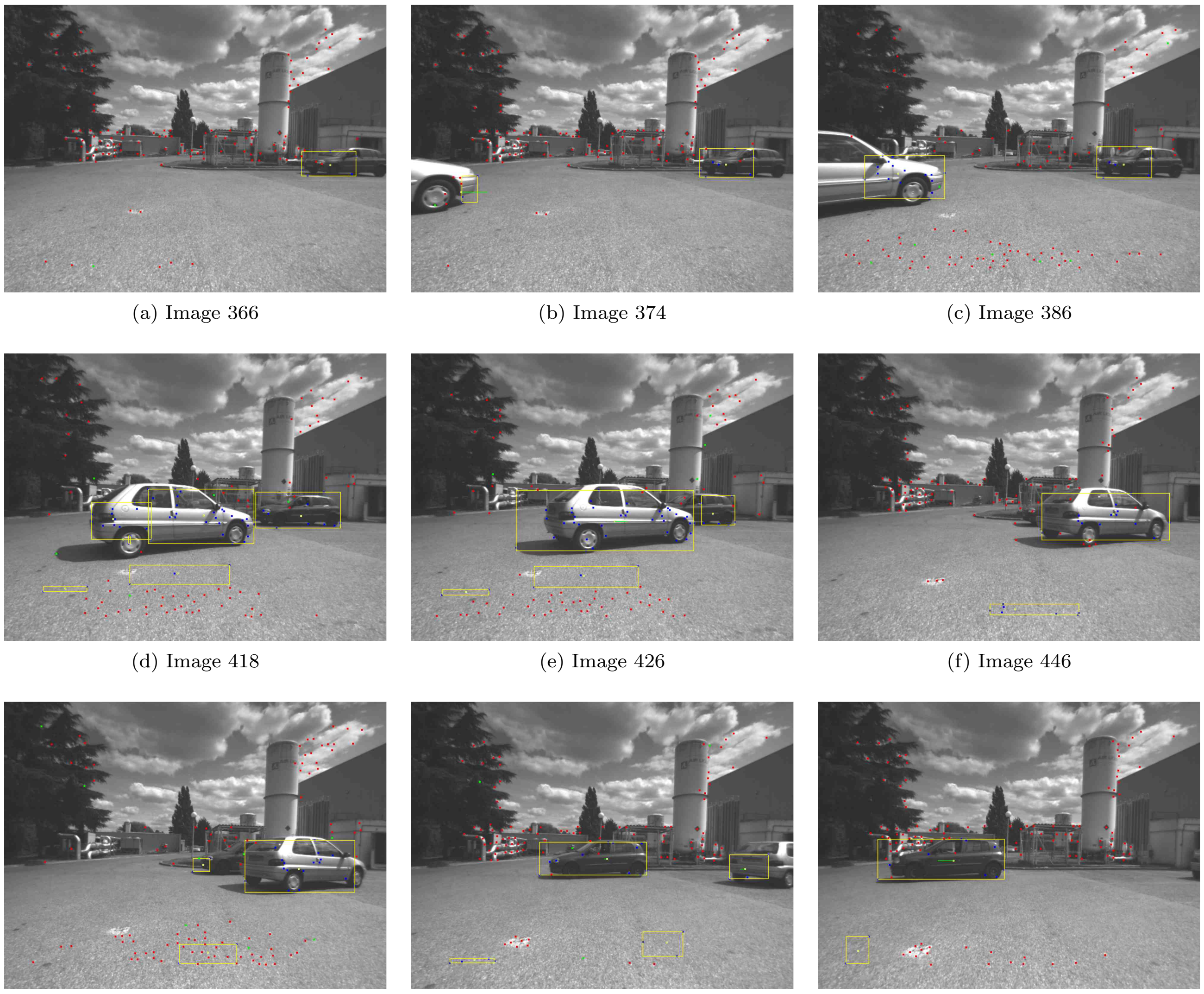
This result is only guaranteed if points are detected on obstacles. It is the reason why a probability map is continuously updated in order to select new points in the region where new objects have the higher probability to appear. By now a parallel architecture is implemented on a multi-processor architecture, made from a large-size Altera FPGA.

REFERENCES

Alenya, G., Nègre, A., and Crowley, J.L. (2009). Time To Contact for Obstacle Avoidance. In *European Conference on Mobile Robotics*. Dubrovnik Croatie.

Almanza-Ojeda, D., Devy, M., and Herbulot, A. (2010). Incremental detection and tracking of moving objects by optical flow and a contrario method. In *Proc. Int. Conf. on Computer Vision Theory and Applications (VISAPP 2010)*. Angers, France.

A.Marin-Hernandez, M.Devy, and Cervantes, J. (2004). Color active contours for tracking roads in natural environments. In *9th Iberoamerican Congress on Pattern Recognition (CIARP'2004), Puebla (Mexico).*



(a) Image 366 (b) Image 374 (c) Image 386

(d) Image 418 (e) Image 426 (f) Image 446

(g) Image 450 (h) Image 470 (i) Image 488

Fig. 7. Detection clustering and tracking objects results. Top row: Two moving objects (labeled as O_1 and O_2) enter the view field and they are detected and tracked. Second row: O_1 and O_2 go across the view field with frontal-parallel motions. Third row: O_1 and O_2 detected again after the crossing event.

Civera, J., Davison, A., and Montiel, J. (2008). Inverse depth parametrization for monocular slam. *IEEE Trans. on Robotics*, 24(5).

Davison, A. (2003). Real-time simultaneous localisation and mapping with a single camera. In *Int. Conf. on Computer Vision*, 1403–1410.

Desolneux, A., Moisan, L., and Morel, J.M. (2008). *From Gestalt Theory to Image Analysis A Probabilistic Approach*, volume 34. Springer Berlin / Heidelberg.

Gate, G., Breheret, A., and Nashashibi, F. (2009). Centralised fusion for fast people detection in dense environments. In *IEEE Int. Conf. on Robotics Automation (ICRA), Kobe, Japan.*

Labayrade, R., Royere, C., Gruyer, D., and Aubert, D. (2005). Cooperative fusion for multi-obstacles detection with use of stereovision and laser scanner. *Autonomous Robots*, 19(2).

Poon, H.S., Mai, F., Hung, Y.S., and Chesi, G. (2009). Robust detection and tracking of multiple moving objects with 3d featu res by an uncalibrated monocular camera. In *Proc. 4th Int. Conf. on Computer Vision/Computer Graphics Collaboration Techniques (MIRAGE'09)*, 140–149. Springer-Verlag, Berlin, Heidelberg.

Shi, J. and Tomasi, C. (1994). Good features to track. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 1994.*, 593–600.

Sola, J., Monin, A., Devy, M., and Lemaire, T. (2005). Undelayed initialization in bearing only slam. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'2005)*, 2751–2756. Edmonton (Canada).

Veit, T., Cao, F., and Bouthemy, P. (2007). Space-time a contrario clustering for detecting coherent motion. In *IEEE Int. Conf. on Robotics and Automation, ICRA'07*, 33–39. Roma, Italy.

Vu, T.V. and Aycard, O. (2009). Laser-based detection and tracking moving objects using data-driven markov chain monte carlo. In *IEEE Int. Conf. on Robotics Automation (ICRA), Kobe, Japan.*

Wang, C., Thorpe, C., Thrun, S., Hebert, M., and Durrant-Whyte, H. (2007). Simultaneous localization, mapping and moving object tracking. *Int. Journal of Robotics Research.*