Vision Based Motion Estimation of Obstacles in Dynamic Unstructured Environments

Andrei Vatavu and Sergiu Nedevschi,

Computer Science Department, Technical University of Cluj-Napoca,
26-28 G. Baritiu Street, Cluj-Napoca, Romania
{Andrei.Vatavu, Sergiu.Nedevschi}@cs.utcluj.ro

Abstract. Modeling static and dynamic traffic participants is an important requirement for driving assistance. Reliable speed estimation of obstacles is an essential goal especially when the surrounding environment is crowded and unstructured. In this paper we propose a solution for real-time motion estimation of obstacles by using the pairwise alignment of object delimiters. Instead of involving the whole 3D point cloud, more compact polygonal models are extracted from a classified digital elevation map and are used as input data for the alignment process.

Keywords: Motion Estimation, Polygonal Map, Object Contour, Iterative Closest Point, Driving Assistance, Stereo-Vision, Object Delimiters.

1 Introduction

In the context of Advanced Driver Assistance Systems, modeling static and dynamic entities of the environment is a key problem. The detection of moving traffic participants is an essential intermediate step for higher level driving technology tasks such as collision warning and avoidance, path planning or parking assistance. The problem of dynamic environment representation becomes even more difficult when the surrounding world is unstructured and heterogeneous, including the cases of crowded urban centers, traffic intersections or off-road scenarios. The representation component may be influenced by several factors: noisy measurements, occlusions, wrong data association or unpredictable nature of the traffic participants. In such complex environments, a driver assistance system should be able to detect other moving traffic entities in real-time and at a high accuracy.

Usually, the classic approaches of dynamic obstacles detection and tracking consist in extracting a set of features from the scene and estimating the motion from their displacement. Current solutions can directly use 3D points [1], or they can track high level attributes such as 2D boxes or 3D cuboids [2], stixels [3], free-form polygonal models [4], object contours [5][8] etc.

The dynamic obstacle modeling solutions can be classified by the nature of used sensors. The most common used sensors are vision based [2], laser [6][7], sonar [9] or radar. The motion estimation techniques are also distinguished by the level at which the dynamic features detection is applied. Some of the existing methods rely on

computing motion before generating a model [10][12], while other methods are based on extracting some attributes and subsequently estimating their dynamic parameters [2][4][5].
Many of dynamic object detection solutions use intermediate representations as primary information. A common practice is mapping 3D information into occupancy grids [10], digital elevation maps [11] or octrees [9].

The data association and identifying correct correspondences steps play an important role in estimating the motion of the traffic entities. One of the widely used methods for model fitting in the presence of many data outliers is the RANSAC algorithm [13]. However, its accuracy depends directly on the number of used samples. This may lead to a high computational cost.

Direct matching solutions such as Iterative Closest Point (ICP) [14] algorithm are most common for vehicle localization and mapping [4]. In [15] the convergence performance for several ICP variants is compared. An optimized ICP method that uses a constant time variant for finding the correspondences is presented. In [4] a moving objects map is segmented by assuming that dynamic parts do not fulfill the constraints of the SLAM. However, the most of scan matching methods do not take into consideration the ego-motion parameters. The data association of objects in subsequent scans is hard to be achieved when the traffic participants or the ego vehicle moves at high speeds or when the measurement uncertainties are not taken into account.

We propose a solution of representing the dynamic environment in real-time by using the pairwise alignment of free-form delimiters and considering the advantages provided by a stereovision system, by inheriting the object information from the intermediate representation. Instead of registering the whole 3D point cloud, our method is based on extracting the most visible object cells from the ego car and using them as input data for the alignment process. We propose an extension of the classical ICP algorithm by applying a set of improvement heuristics:

- The data association is one of the problems of the classical scan matching techniques. It’s hard to estimate the correspondent models from previous scans only based on the proximity criterion. In our case we introduce a pre-processing step. First, we find the correspondence pairs between the model set (contour extracted in previous frame) and the measurement set (current frame results) by finding similarities between object blobs and passing this information at the contour level. Then, a list of associated contour candidates is generated and is used as the input for the next steps of the alignment;
- For the registration process we use free-form polygonal models that minimize the erroneous results caused by occlusions, or by stereo reconstruction errors. The main idea is that we are taking into account only the most visible points from the ego-vehicle by performing a radial scanning of the environment [16];
- The previously extracted speeds are used as the initial guess for the ICP algorithm;
- In order to filter the alignment outliers, a rejection metric that includes stereo uncertainties is proposed;

Our method is based on information provided by a Digital Elevation-Map, but can be easily adapted for other types of intermediate representations.

The remaining of the paper is structured as follows: Section 2 introduces the architecture of the proposed dynamic environment representation. Section 3 presents the pre-processing module with a group of necessary tasks for extracting object dynamic properties. In section 4, the main steps of the motion estimation component are detailed. The last two sections show the experimental results and conclusion about this contribution.

2 System Architecture

The dynamic environment representation method has been developed and adapted for crowded environments such as urban city traffic scenes. In this paper we extend our previous Dense Stereo-Based Object Recognition System (DESBOR) [22]. The system architecture (see Fig. 1) could be divided in four main blocks: data acquisition and 3D reconstruction, intermediate representation, pre-processing, and motion estimation.

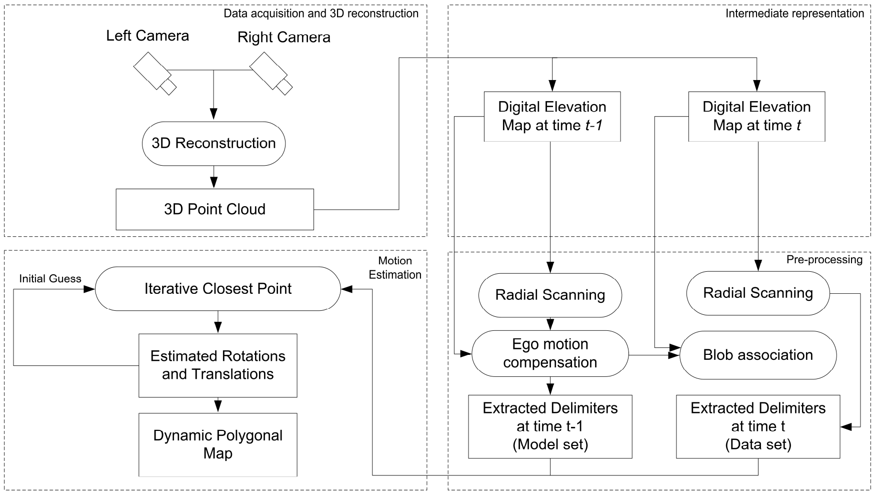
Data acquisition and 3D reconstruction
Left Camera
Right Camera
3D Reconstruction
3D Point Cloud
Intermediate representation
Digital Elevation Map at time t-1
Digital Elevation Map at time t
Motion Estimation
Initial Guess
Iterative Closest Point
Estimated Rotations and Translations
Dynamic Polygonal Map
Pre-processing
Radial Scanning
Radial Scanning
Ego motion compensation
Blob association
Extracted Delimiters at time t-1 (Model set)
Extracted Delimiters at time t (Data set)

Fig. 1. System Architecture.

Data acquisition and 3D reconstruction is the first level of the processing flow. At this stage the images are acquired from the two cameras, then the 3D reconstruction is performed using a specialized TYZX [17] board. The resulted point cloud is used as the input information for computing the Digital Elevation Map.

Intermediate representation: the raw dense stereo information is mapped into a Digital Elevation Map (see Fig. 2). The resulted intermediate representation contains three types of cells: road, traffic isle and object. The cells are labeled based on their height information. More details about the Elevation Map are presented in [18].

Pre-processing: The pre-processing level groups a set of basic tasks that are performed prior the ICP algorithm. At this phase, the object contours are extracted by radial scanning of the Elevation Map. For the delimiters extraction we use the Border

Scanner algorithm previously developed by us [16]. We apply the ego-motion compensation for the Elevation Map and contours that are extracted in previous frame, assuming that we know the odometry information. The ego-vehicle motion is compensated in order to separate its speed from the independent motion of the objects in the traffic scene. Another pre-processing task is to associate the polygonal models. The data association is achieved by using the maximum overlapping score of the Elevation Map blobs. Considering that each polygonal model inherits the blob type, it also inherits the blob association information.

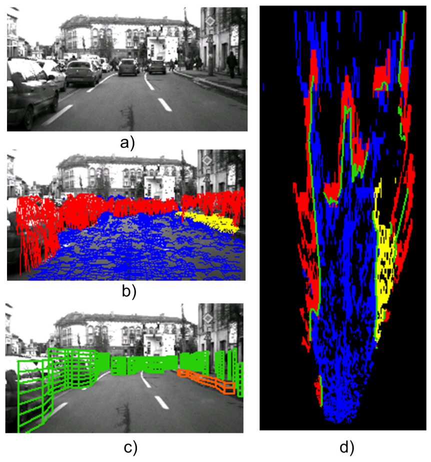

Fig. 2. a) An urban traffic scene. b) The Elevation Map projected on the left camera image. c) A compact representation of the environment. d) The top view of the Elevation Map. The Elevation Map cells are classified (blue – road, yellow – traffic isle, red – obstacles).

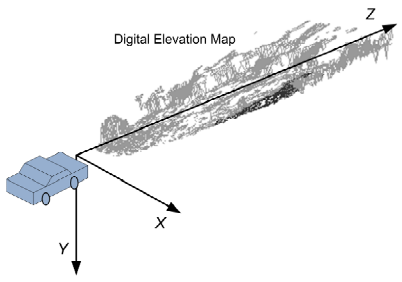

Fig. 3. Coordinate System.

Motion Estimation: As the result of the pre-processing level, a list of candidates is provided for the ICP module. Each candidate represents a pair of associated contours in the previous stage. For each candidate, a rotation and a translation is estimated by the ICP algorithm. Then the computed motion information is associated to the static polygonal models. A dynamic polyline map is generated as the result. Each polyline element is characterized by a set of vertices describing the polygon, position, height, type (traffic isle, obstacle), orientation and magnitude.

In our case the two cameras are placed on a moving vehicle. We use a coordinate system where the z axis points toward the direction of the ego-vehicle, and the x axis is oriented to the right. The origin of the coordinate system is situated in front of the car (see Fig. 3).

3 Pre-Processing Level

The pre-processing stage consists in performing necessary tasks prior the motion estimation. First, extracting a sufficiently generic model is needed. The extracted model should allow us the creation of fast subsequent algorithms and as well it should minimize the representation errors caused by noisy 3D reconstruction or by occlusions.

A second task is to separate the ego-vehicle speed from the independent motion of the other objects in the traffic scene. This is achieved by compensating the ego motion.

And finally, elevation map blob is labeled and is used in data association. As the result a list of pairs of contours is extracted and is provided subsequently to the ICP step. Thus, unlike the other classical methods that involve aligning the whole local maps at once, and then segmenting the dynamic obstacles from the static ones, we first associate the obstacles at the blob level and then apply the ICP for each associated candidate.

3.1 Polyline-Based Environment Perception

For the polyline based object representation we use the Border Scanner algorithm described by us in [16]. The main idea is that we are taking into account only the most visible points from the ego car and extract object delimiters by radial scanning of the Elevation Map. Our method is similar to a Ray-Casting approach. The proposed method consists in determining the first occupied point intersected by a virtual ray which extends from the ego-car position. The scanning axis moves in the radial direction, having a fixed center at the ego-vehicle position (the coordinate system origins). At each step we try to find the nearest visible point from the Ego Car situated on the scanning axis. In this way, all subsequent cells P_i are accumulated into a Contour List C, by moving the scanning axis in the radial direction:

$$C = \{P_1, P_2, \ldots P_n\} \tag{1}$$

For each object O_i described by a contour C_i we apply a polygonal approximation of C_i by using a split-and-merge technique described in [19]. The extracted polygon is used to build a compact 3D model based on the polyline set of vertices as well as on the object height. A polyline based representation is described in Fig. 2.d.

3.2 Ego-Motion Compensation

Before estimating the motion of the traffic entities, the movement of the ego vehicle must also be taken into consideration. In order to compensate the ego motion in the successive frames, for each given point $P_{t-1}(x_{t-1}, y_{t-1}, z_{t-1})$ in the previous frame, the corresponding coordinates $P_t(x_t, y_t, z_t)$ in the current frame are computed by applying the following transformation:

$$\begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} = R_y(\psi) \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ z_{t-1} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ t_z \end{bmatrix} \quad (2)$$

Where $R_y(\psi)$ is the rotation matrix around the Y axis with a given angle ψ, and t_z is the translation on the Z axis. The rotation and the translation parameters are provided by the ego-car odometry. It is considered that the translations on the X and Y axis are zero.

3.3 Data Association

This stage consists in finding the corresponding contours that identify a single object in consecutive frames. As each extracted contour describes an Elevation Map blob, finding the associated contour pairs is reduced to find a similarity between the object blobs.

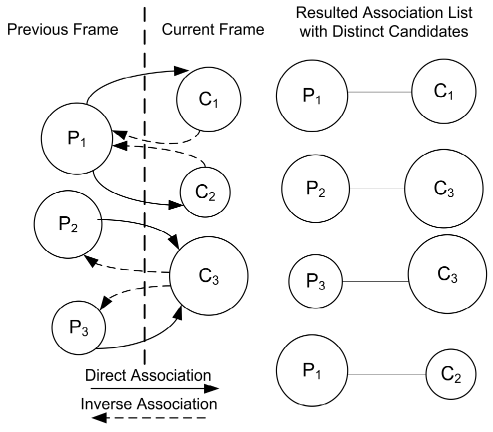

Fig. 4. The association between two set of blobs in the consecutive frames and the resulting set of associated pairs.

For each object P_i from the previous frame and for each object C_j from the current frame we calculate an overlapping score A_{ij}. The results are stored into a score matrix

$A=\{A_{ij}\}$. Candidates with the highest score are taken into account in determining the associations between the two set of objects P and C.

However the association problem may lead only to partial results in the cases when larger objects from the previous frame are split into smaller blobs in the current frame and vice versa. In order to find all possible pairs of candidates we perform two types of associations: a direct association (forward association) finding best overlapping candidates in the current frame for all blobs in the previous frame, and a reverse association (backward association) that finds best overlapped objects in the previous frame for all objects from the current frame (see Fig.4). The final list of candidates includes all distinct pairs associated in the two steps.

4 Motion Estimation

The object motion estimation module receives as input a list of associated contour pairs. For each distinct pair we compute correspondences between the two contours and estimate a rotation and a translation which minimize the alignment error. For the contour pairwise registration we use the Iterative Closest Point (ICP) method. The ICP algorithm was proposed by Besl and McKay [14] and represents a common solution especially for scan-matching techniques, but the idea could be adapted for any kind of models.

For each contour pair that identifies the same object in the consecutive frames we define two set of points: a model set $P=\{p_1,p_2, ..., p_M\}$ that describes the object contour in the previous frame, and a data set $Q=\{q_1,q_2, ..., q_K\}$ that describes the object contour in the current frame. For each point q_j from Q the corresponding closest point p_i from P is found. We want to find an optimal rotation R and translation T that minimize the alignment error. The objective function is defined:

$$E(R,T)=\sum_{i=1}^{N}\left\|Rp_i+T-q_i\right\|^2 \tag{3}$$

where p_i and q_i are the corresponding point pairs of the two sets and N is the total number of correspondences.

The proposed alignment method is described by the following main steps:

1. Matching – for each point from data set, the closest point from the model set is found. A list of correspondent pairs is generated.

2. Outliers Rejection – Rejecting the outliers that could introduce a bias in the estimation of translation and rotation.

3. Error Minimization – estimating new transformation parameters R and T for the next iteration.

4. Updating – having the new R and T, a new target set is computed by applying the new transformation to the model set. A global transformation M_g is updated with the new R and T values.

5. Testing the convergence – compute the average point-to-point distance between the measurement set and transformed model set. Then test if the algorithm has been converged to a desired result. If the error is greater than a given threshold, the process continues with a new iteration. The algorithm stops when the computed error is below the selected error threshold or when a maximum number of iterations have been achieved.

Next we will detail each of these steps.

4.1 Matching

At this stage, for each point q_i from Q we want to find the closest point from the model set P:

$$d(q_i, P) = \min_{j \in \{1..N_p\}} d(q_i, p_j) \tag{4}$$

Usually this task is the most computationally extensive in the ICP algorithm. The classical brute force search approach has a complexity of $O(N_q \cdot N_p)$, with N_p being the number of points in P and N_q – the number of points in Q. In order to reduce the complexity to $O(N_q \cdot \log N_p)$ many solutions employ a KD-Tree [21] data structure. In our case, for finding closest points problem, we use a modified version of Chamfer based Distance Transform [20] (see Fig. 5).

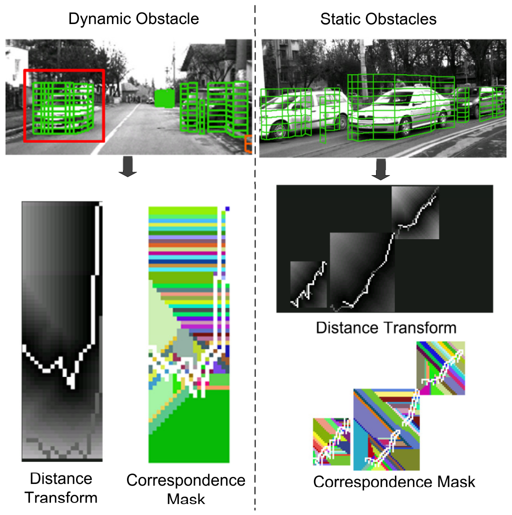

Fig. 5. Distance Transforms and Corresponding Masks are computed for dynamic obstacles (left side), and for static obstacles (right side). Data contours (gray color) and model contours (white color) are superimposed on the Distance Transform image. Each contour point in the correspondence mask is labeled with a unique color. The colors in the corresponding mask identify uniquely the closest contour point (having the same color).

A distance transform represents a map that has the property that each map cell has a value proportional to the nearest obstacle point. In our case, for each separate model contour we define a region of interest and compute the distance transform. The difference of our solution is that we use two maps: a distance map that stores the minimum distances to the closest points, and a correspondence map, storing the positions of the closest points (see Fig. 5). The correspondences from the model set are identified by superimposing the data contour on the two masks.

4.2 Outliers Rejection

The purpose of this stage is to filter erroneous correspondences that could influence the alignment process. We use two types of rejection strategies: rejection of pairs whose point-to-point distance is greater than a given threshold, and eliminating the points where the overlap between the two contours is not complete.

4.2.1 Distance Based Rejection

The classical strategy consists in rejection of pairs whose point-to-point distance is larger than a given threshold D_t:

$$d(q_i, p_j) > D_t \tag{5}$$

Because the stereo reconstruction error generally increases with the square of the z distance, the stereo-system uncertainties must be taken into account. As suggested by [10], if we assume that the stereo-vision system is rectified, then the *z* error is given by the following relation:

$$\sigma_z = \frac{z^2 \cdot \sigma_d}{b \cdot f} \tag{6}$$

Where *z* is the depth distance, *b* is the stereo system baseline; *f* is the focal length and σ_d denotes the disparity error. Thus, for each corresponding pair of points (p_i,q_i) from the two sets, the rejection is made if:

$$d(q_i, p_j) > D_t + \sigma_z \tag{7}$$

This would mean that the rejecting threshold is increased at once with the *z* distance.

4.2.2 Boundary Based Rejection

The second type of rejecting is filtering the point correspondences caused by incomplete overlap between contours. Usually, these situations appear when one of the two contours is incompletely extracted due to occlusions, and may lead to incorrect alignments. A possible solution is to identify the subsets of points from *Q* that have the same correspondent point p_j in *P*, and keeping only the pair with the minimum distance (see Fig. 6).

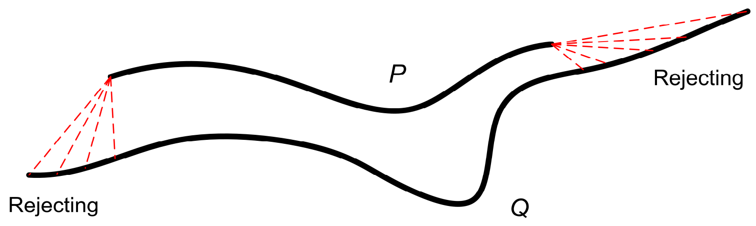

Fig. 6. Rejecting the contour boundary.

4.3 Error minimization

In this step we determine the optimal rotation R and translation T by minimizing the objective function defined by Equation (3).

The rotation matrix around the Y axis is linearized, approximating $\cos\alpha$ by 1 and $\sin\alpha \approx \alpha$ by α:

$$R_y(\alpha) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & 0 \\ -\alpha & 0 & 1 \end{bmatrix} \tag{8}$$

The translation vector is defined as:

$$T = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T \tag{9}$$

We can rewrite the Equation (3) as:

$$E(R,T) = \sum_{i=1}^{N} \left\| \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & 0 \\ -\alpha & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{i,x} \\ p_{i,y} \\ p_{i,z} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} - \begin{bmatrix} q_{i,x} \\ q_{i,y} \\ q_{i,z} \end{bmatrix} \right\|^2 \tag{10}$$

The $E(R,T)$ is minimized with respect to α, t_x, t_y, and t_z by setting the partial derivatives to zero:

$$\begin{cases} \dfrac{\partial E(R,T)}{\partial \alpha} = 2\sum_{i=1}^{N} \begin{bmatrix} \alpha(p_{i,x}^2 + p_{i,z}^2) + t_{i,x} p_{i,z} \\ -t_{i,z} p_{i,x} - q_{i,x} p_{i,z} + q_{i,z} p_{i,x} \end{bmatrix} = 0 \\ \dfrac{\partial E(R,T)}{\partial t_x} = 2\sum_{i=1}^{N} \left(t_{i,x} + p_{i,x} + \alpha p_{i,z} - q_{i,x} \right) = 0 \\ \dfrac{\partial E(R,T)}{\partial t_y} = 2\sum_{i=1}^{N} \left(t_{i,y} + p_{i,y} - q_{i,y} \right) = 0 \\ \dfrac{\partial E(R,T)}{\partial t_z} = 2\sum_{i=1}^{N} \left(t_{i,z} + p_{i,z} - \alpha p_{i,x} - q_{i,z} \right) = 0 \end{cases} \tag{11}$$

Therefore we can obtain the unknown coefficients:

$$\begin{cases} \alpha = \dfrac{1}{N\sum_{i=1}^{N}(p_{i,x}^2 + p_{i,z}^2) - \left(\sum_{i=1}^{N} p_{i,x}\right)^2 - \left(\sum_{i=1}^{N} p_{i,z}\right)^2} \cdot \\ \cdot\left(\sum_{i=1}^{N} p_{i,x}\sum_{i=1}^{N} q_{i,z} - \sum_{i=1}^{N} p_{i,z}\sum_{i=1}^{N} q_{i,x} + \right. \\ \left. + N\left[\sum_{i=1}^{N}(q_{i,x}p_{i,z}) - \sum_{i=1}^{N}(q_{i,z}p_{i,x})\right]\right) \\ t_x = \dfrac{1}{N}\left(\sum_{i=1}^{N} q_{i,x} - \sum_{i=1}^{N} p_{i,x} - \alpha\sum_{i=1}^{N} p_{i,z}\right) \\ t_y = \dfrac{1}{N}\left(\sum_{i=1}^{N} q_{i,y} - \sum_{i=1}^{N} p_{i,y}\right) \\ t_z = \dfrac{1}{N}\left(\sum_{i=1}^{N} q_{i,z} - \sum_{i=1}^{N} p_{i,z} + \alpha\sum_{i=1}^{N} p_{i,x}\right) \end{cases} \tag{12}$$

4.4 Updating

Assuming that we have estimated new R and T parameters in the previous step, a new target set is computed by applying the new transformation to the model set.

Having the rigid body transformation matrix *M*:

$$M = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \alpha & t_x \\ 0 & 1 & 0 & t_y \\ -\alpha & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

Each point p_i from the model set *P* is transformed according to the following relation:

$$\begin{bmatrix} p_{i,x}^{k+1} \\ p_{i,y}^{k+1} \\ p_{i,z}^{k+1} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \alpha & t_x \\ 0 & 1 & 0 & t_y \\ -\alpha & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{i,x}^{k} \\ p_{i,y}^{k} \\ p_{i,z}^{k} \\ 1 \end{bmatrix} \tag{14}$$

Finally, a global transformation M_G is updated:

$$M_G^{k+1} = M_G^{k} \cdot M \tag{15}$$

4.5 Testing the convergence

The error metric is estimated by computing the average Euclidean distance (AED) of every corresponding pair of data set *Q* and transformed model set.

$$Err = \frac{1}{N}\sum_{i=1}^{N}\left\| p_i - q_i \right\| \tag{16}$$

If the error is greater than a given threshold, the process continues with a new iteration. The algorithm stops when the computed error is below the selected error threshold or when a maximum number of iterations have been achieved.

5 Experimental Results

The proposed dynamic environment representation method has been tested in different traffic situations. For our experiment we used a 2.66GHz Intel Core 2 Duo Computer with 2GB of RAM. Fig. 7 presents some qualitative results obtained in a dynamic urban traffic scenario.

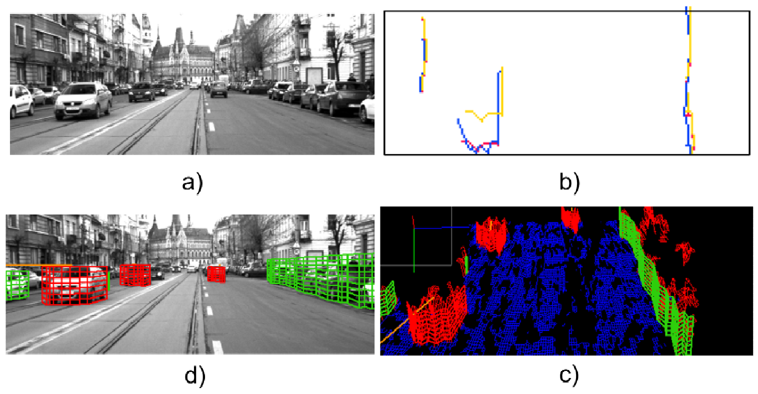

Fig. 7. a) An urban traffic scenario. b) The alignment result (red color) between the model delimiter extracted in previous frame (yellow) and the data contour, extracted in the current frame (blue). c) The virtual view of the scene. The static obstacles are represented with green delimiters while the dynamic obstacles are colored with red. The speed vectors are associated to each dynamic entity. d) The representation result, projected on the left camera image.

In Fig. 7.b the model delimiter that was extracted in previous frame is colored with yellow, while the data contour (extracted in the current frame) is drawn with blue. The result of the alignment is illustrated with red color. It can be observer that in the case of the incoming vehicle, as well as for the lateral static vehicles, the aligned model is superimposed almost perfectly on the data set. In the Fig. 7.c, the virtual view of the scene is shown. The static obstacles are represented with green delimiters while the dynamic obstacles are colored with red. The speed vectors are associated to the each dynamic entity (yellow color). The representation result is also projected on the left camera image (see Fig. 7.d). We considered that the obstacles with a speed greater than 8km/h are dynamic.

Fig. 8 shows a comparative result between the ICP algorithm that includes all correspondence points (blue color) and the alignment method that uses the Contour Boundary Rejection strategy (red color). We used the Average Euclidean Distance (AED) as the error metric. It can be observed that the ICP algorithm based on

Boundary Rejection strategy converge more quickly than the ICP method without a filtering mechanism and proves to be more accurate having a lower alignment error. For our experiments we used a maximum number of 10 iterations. The average processing time was about 38 ms.

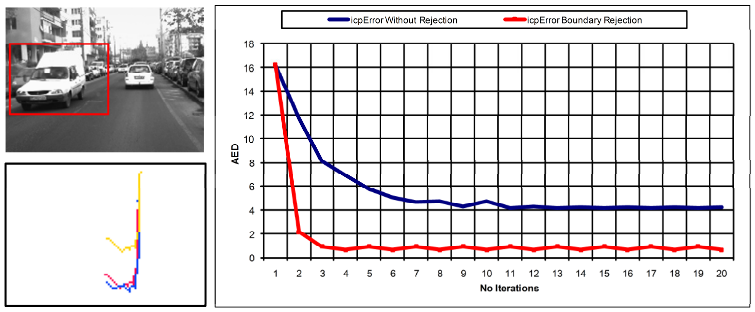

Fig. 8. The computed Error Metric in the case of ICP algorithm that does not use outlier rejection and ICP method that uses a Boundary Rejection.

6 Conclusions

In this paper we propose a method of real-time representation of the dynamic environment by using the pairwise alignment of free-form models. Instead of registering the whole 3D point cloud, the most visible obstacle points from the ego car are extracted and are subjected to the alignment process. We extend the classical ICP algorithm with a set of preprocessing tasks. First, we associate the delimiters at the blob level. Then, a list of associated candidates is passed to the alignment stage. For the registration process we use free-form polygonal models that minimize the erroneous results caused by occlusions, or by stereo reconstruction errors. As future work we propose to improve the stability of the environment perception by extending our system with a temporal filtering of the estimated speeds.

References

1. Franke, U., Rabe, C., Badino, H., Gehrig, S.: 6d-vision: Fusion of stereo and motion for robust environment perception. In: 27th Annual Meeting of the German Association for Pattern Recognition DAGM '05, pp. 216-223 (2005).
2. Danescu, R., Nedevschi, S., Meinecke, M.M., Graf, T.: Stereovision Based Vehicle Tracking in Urban Traffic Environments. In: IEEE Intelligent Transportation Systems Conference (ITSC 2007), Seattle, USA (2007).
3. Pfeiffer, D Franke., U.: Efficient Representation of Traffic Scenes by Means of Dynamic Stixels. In: IEEE Intelligent Vehicles Symposium (IEEE-IV) 2010, pp. 217-224 (2010).

4. Wang, C.CThorpe., CHebert., MThrun., S., Durrant-Whyte, H.: Simultaneous localization, mapping and moving object tracking. In: International Journal of Robotics Research, 26(6) (2007).
5. Prakash, S., Thomas, S.: Contour tracking with condensation/stochastic search. In: Dept. of CSE. IIT Kanpur (2007).
6. Thomas de Candia: 3d tracking of dynamic objects with a laser and a camera. In: Technical report, Autonomous System Lab, ETH Zurich (2010).
7. Madhavan, R.: Terrain aided localization of autonomous vehicles. In: Symposium on Automation and Robotics in Construction, Gaithersburg (2002).
8. Yokoyama, M., Poggio, T.: A Contour-Based Moving Object Detection and Tracking. In: 14th International Conference on Computer Communications and Networks (ICCCN '05). IEEE Computer Society, pp. 271-276, Washington, DC, USA (2005).
9. Fairfield, N., Kantor, G. A., Wettergreen, D.: Real-Time SLAM with Octree Evidence Grids for Exploration in Underwater Tunnels. In: Journal of Field Robotics (2007).
10. Danescu, R., Oniga, F., Nedevschi, S.: Modeling and Tracking the Driving Environment with a Particle Based Occupancy Grid. In: IEEE Transactions on Intelligent Transportation Systems, vol. 12, No. 4, pp. 1331-1342 (2012).
11. Danescu, R, Oniga, F., Nedevschi, S., Meinecke, M-M.: Tracking Multiple Objects Using Particle Filters and Digital Elevation Maps. In: IEEE Intelligent Vehicles Symposium (IEEE-IV 2009), pp. 88-93, Xi'An, China (2009).
12. Hess, J. M.: Characterizing dynamic objects in 3d laser range data. In: Master's thesis, Albert-Ludwigs-Universitt Freiburg (2008).
13. Fischler, M., Bolles, R.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: Communications of the ACM, vol. 24, no. 6, pp. 381– 395 (1981).
14. Besl P., McKay, N.: A method for registration of 3d shape. In: Trans. Pattern Analysis and Machine Intelligence, 12(2) (1992).
15. Rusinkiewicz, S., Levoy, M.: Efficient Variants of the ICP Algorithm. In: Third International Conference on 3D Digital Imaging and Modeling (2001).
16. Vatavu, A., Nedevschi, S., Oniga, F.: Real Time Object Delimiters Extraction for Environment Representation in Driving Scenarios. In: ICINCO-RA 2009, pp 86-93, Milano, Italy (2009).
17. Woodill, J. I., Gordon, G., Buck, R.: Tyzx deepsea high speed stereo vision system. In: IEEE Computer Society Workshop on Real Time 3-D Sensors and Their Use, Conference on Computer Vision and Pattern Recognition (2004).
18. Oniga, F., Nedevschi, S.: Processing Dense Stereo Data Using Elevation Maps: Road Surface, Traffic Isle, and Obstacle Detection. In: IEEE Transactions on Vehicular Technology, Vol. 59, No. 3, pp. 1172-1182 (2010).
19. Douglas, D., Peuker, T.: Algorithms for the reduction of the number of points required to represent a digitised line or its caricature. In: Canadian Cartographer, Vol 10, pp. 112-122 (1973).
20. Borgefors, G.: Distance transformations in arbitrary dimensions. In: Comput. Vis. Graph. Image Proc. 27, 321–345 (1984).
21. Bentley, J.L.: Multidimensional binary search trees used for associative searching. In: Communications of the ACM. 18(9):509-517 (1975).
22. Nedevschi, S., Danescu, R., Marita, T., Oniga, F., Pocol, C., Sobol, S., Tomiuc, C. Vancea, C., Meinecke, M. M., Graf, T., To, T. B., Obojski, M. A.: A sensor for urban driving assistance systems based on dense stereovision. In: Intelligent Vehicles 2007, pp. 278-286, Istanbul (2007).