A Framework for Object Detection, Tracking and Classification in Urban Traffic Scenarios Using Stereovision

Silviu Bota and Sergiu Nedevschi
Technical University of Cluj-Napoca
{Silviu.Bota, Sergiu.Nedevschi}@cs.utcluj.ro

Abstract

Driving assistance systems provide either safety or comfort functions. Such systems must evaluate the state of the world and take necessary actions. A preliminary step for evaluating the state of the world is to detect, track and classify scene objects. The classification step becomes especially important in complex urban traffic scenarios. In such scenarios the sensors of choice are vision based, as they provide detailed scene information. In this paper we present the architecture of a detection, tracking and classification framework based on stereovision. We detect objects using either a points grouping algorithm (for large objects) or a density map grouping algorithm (for small objects). We perform first a rough classification based on objects’ dimensions and track objects according to the motion model of each class. We extract motion features and perform a refined classification. Class specific dichotomizers are subsequently used to filter the classified objects, rejecting incorrect classification. A large database of manually labeled objects is used for determining motion models, training the classifiers and measuring the performance of the system.

1. Introduction

Traditional driving assistance systems such as adaptive cruise controls are limited to highway environments. The highway environment is relatively simple, the only moving obstacles being other vehicles, and static obstacles being usually limited to side fences. For such environments precise but simple active sensors, such as radar, are able to provide sufficient information to the driving assistance systems.

Urban driving assistance systems are however faced with the much more difficult problem of understanding extremely complex scenarios. In urban scenarios other classes of obstacles such as pedestrians, cyclists, poles and parked cars are common. In the presence of intersections, high curvature roads, varying lane width, parked cars and pedestrians traditional radar based approaches to ACC quickly become useless. However, both comfort and safety systems are most useful in urban environments because these environments are the ones more trying to the driver. As stereovision sensors provide an increased amount of information about the scene, they become the sensors of choice for implementing comfort and safety systems for driving in urban environments.

Some of the problems that must be solved by urban driving assistance systems are: static obstacle detection, especially for avoiding them by the front and lateral assist function and for emergency breaking or active protection devices (airbags) deployment. Moving obstacle detection and tracking, for stop and go ACC functions, emergency breaking and collision mitigation. An especially useful requirement for collision mitigation is the detection of the occupant cell of the collision partner (car), in order to avoid causing injury to its passengers. Pedestrian recognition and protection is also a major requirement in urban environments, as they are the most vulnerable traffic participants. Another difficult problem is the detection and recognition of traffic signs and traffic lights.

From the above requirements, it is evident that, in urban environments, object detection, tracking and classification represent necessary steps for implementing driving assistance functions.

In this paper we present the architecture of our generic framework for obstacle detection, tracking and classification for urban traffic scenarios using stereovision. We detect objects using either a points grouping algorithm (for large objects) or a density map grouping algorithm (for small objects). We perform first a rough classification based on objects’ dimensions and track objects according to the motion model of each class. We extract motion features and perform a refined classification. Class specific dichotomizers are subsequently used to filter the classified objects, rejecting incorrect classifications. A large database of manually labeled
objects is used for determining motion models, training the classifiers and measuring the performance of the system.

1.1. Paper Structure

In the next section we present the architecture of our system. In section 3 we present other similar approaches for object detection and classification. In the subsequent sections we present the parts of our system that represent original contributions: in section 4 we present our algorithm for small object detection using density maps. In section 5 we present our algorithm for merging the object detected using our two detection methods. In section 6 we present our tracking algorithm for pedestrians and fixed obstacles. In section 7 we present our motion signature and motion periodicity feature extraction. In section 8 we present our classification algorithms and their experimental results.

2. System Architecture

We use a pair of grayscale cameras for acquiring images at high resolution (1376 x 1030). Subsequently the images are undistorted, rectified and downsampled [1] to the resolution of (512 x 383) for the purpose of stereo reconstruction which is performed using a hardware system [2]. The result of this step is a cloud of 3D points.

Following a point classification by the lane’s vertical profile [3] or by digital elevation map [4], two algorithms (points grouping [5] and density map [6]) are employed for object detection. The results are two lists of 3D boxes, corresponding to the two algorithms. Because objects detected by the two algorithms may overlap, the two lists of objects are merged into a hierarchical list, where large objects (usually detected by the points grouping algorithm) may contain smaller parts (usually detected by the density map algorithm).

A rough pre-classification of the 3D boxes is employed, based on object dimensions (width, height, length). Three large classes are determined, vehicles (cars, trucks), pedestrians and fixed obstacles (poles, infrastructure parts etc.). An object may belong to multiple classes at these stage (if it’s dimensions do fit multiple classes). Because each of these large classes has a different motion model, they are tracked using different methods.

Vehicles are tracked using a tracking algorithm tuned for urban ACC functions [7]. Pedestrian and fixed obstacles are tracked using Kalman filters with different parameters and an optimal association based on bipartite graph matching. The result of tracking is both an

Figure 1. System Architecture
improved object positioning and link between an object in the current frame and the same object in the previous frame. This temporal link allows the computation of temporal features. These are the object speeds on longitudinal and lateral directions, a “motion signature” and the motion periodicity.

Following the extraction of this improved set of features, a further classification stage takes place. In this step we classify objects into trucks, cars, pedestrians, poles, and “others”. Subsequently, based on the object’s class we extract some class specific features. These are based on the histogram of gradient orientation (HOG) for cars and pedestrians and on contour matching for pedestrians and poles. These features are mostly orthogonal with the features used in the previous classification step. Therefore they are used to further test the if an object is really an instance of its class and reject false positives. Following this step some possible class dependent processing that can be performed include occupant cell detection, danger assessment, traffic sign recognition etc.

3. Related Work

There are various methods for object detection. Some are based on 2D information, for example the detection of image regions where there are a significant number of vertical edges [8]. Other methods are based on some type of additional information like IR images [8], or depth information [9],[8]. Most of pedestrian detection methods, which use depth information, rely on the disparity map and make some kind of segmentation on this map to detect objects [9], or use a v-disparity approach [8]. However, although the approaches based on disparity maps are faster as compared to approaches based on full 3D information, these approaches rely heavily on a dense and error free disparity map, which are hard to obtain in real life scenarios. Some methods to reduce errors are cumulation using v-disparity or the Generalized Hough Transform paradigm. A similar approach to our own, which uses a criterion based on reconstructed 3D points density is described in [10]. The authors divide the ground plane into cells and compare the number of reconstructed points with the expected number. However, this approach may be unable to accurately detect such small objects as pedestrians.

Object detection and object classification based on pattern matching are traditionally limited to 2D image intensity information [11]. Sometimes techniques such as Adaboost with illumination independent features are used [12]. The advantage in using the 2D information consists in the fact that all the information have a high degree of accuracy and a high level of trust since the image represents an accurate projection of the real scene (without taking into account the image noise, which is not a significant factor for high quality video cameras; of course, a production system would use lower quality cameras and should apply some filtering to remove noise before processing). The disadvantage of using only 2D image information is that we do not have any additional spatial information about where the objects are and what their size is. A detection system based on pattern matching using only the intensity image will usually try to fit all the models using all the positions and scales that are plausible in order to find a match in the image. This generates an extremely large search space, which cannot be reduced because the 3D information is missing. Pattern matching approaches based on methods similar to the distance transform allow a limited degree of difference between the model and the features of the matched object but still require a large number of scales and positions for each model.

The 3D information generated by a stereo reconstruction system provides depth for objects which make up the scene. There are some classification methods that use this information directly [13], [14]. The classification based solely on 3D information is difficult as the 3D reconstruction systems do not provide sufficiently dense and reliable 3D data to allow the reliable extraction of 3D shapes and surfaces (however, for a promising approach in surface extraction see [4]).

The use of pattern matching in conjunction with 3D information has not been extensively explored, mainly because real time dense stereo reconstruction systems have not been available until recently. Some approaches aimed at pedestrian detection have used dense 3D information, but only as a validation method [15]. The 3D data generated by these real-time dense stereo reconstruction devices is still noisy and has a much lower degree of confidence than intensity data. However, by careful calibration of the stereo rig and careful filtering and usage of the 3D data, it is now possible to extract quality results from the 3D data.

Another important feature for walking pedestrians detection is their walking pattern. There are a number of works, e.g. [16], [17] which have used motion cues in pedestrian detection. A typical approach for motion based pedestrian detection is to determine if the motion associated with the presumed pedestrian is periodic. However, in cluttered environments, it is usually very hard to distinguish object motion from background motion. Also, due to high car speed, close range and low framerate, it is possible that objects will move many pixels from one frame to the next, thus making local methods for motion detection infeasible. In motion based pedestrian detection too, 3D features were
Figure 2. a) Greyscale image and objects b) 3D points projected on the xOz plane (the big box represents the result of a different object detection algorithm [1])

not extensively explored. The main advantage of a 3D approach in motion analysis is the possibility of correctly segmenting foreground and background in complex scenarios and with moving cameras. Also, having 3D information means that the true scale of the motion can be recovered.

4. Object Detection Using Density Map

Our method for small object detection relies on the fact that objects have a higher concentration of reconstructed 3D points than the road surface (see figure 2). Also, vertical structures are, usually, very well reconstructed by the 3D reconstruction engines when the stereo camera system has a horizontal baseline (our case also). This results in a high density of 3D points in the 3D space occupied by objects in general and pedestrians in particular. By trying to determine those positions in space where there is a high density of 3D points, possible positions for objects can be determined.

The characteristics of a collision avoidance safety system require a detection area of 20m (longitudinal) and 10m (lateral). Our camera system is designed to give the best results in these range. Therefore, we select, for the purpose of small object detection a box-shaped subspace of the scene, having a 20m length by 10m width by 2m height. The height restriction is necessary in order to avoid spurious detection of tree foliage, suspended objects etc.

In order to cope with projection errors caused by errors in the cameras’ extrinsic parameters we use a very precise calibration procedure, that produces exact results up to a range of 35m [18]. The precise calibration procedure allows stereo reconstruction to be performed in principle up to the specified distance, without significant outliers caused by incorrect image rectification or un-distortion. The pitch and roll of the ego vehicle are estimated online [19]. However, some errors are still present, caused by imperfect stereo matching. These errors are filtered out by the accumulation (averaging) effect of the density map and do not significantly alter the performance of our system in the detection area given above. There is also an increase in reconstruction error with distance. The stereo reconstruction, performed by the TYZX hardware computes disparities with sub-pixel accuracy, and because of this the errors are not too serious for the considered range.

A density map is constructed from the 3D points located within this limited subspace. The 3D points are projected on the xOz (road) plane. In this limited area that we have considered for detection, the road can be considered flat, so it is no problem that a plane is used.

The density map is an accumulation buffer. Each projected point adds a value to the accumulation buffer. A cell in the accumulation buffer covers an area of 50mm × 50mm on the ground plane. The weights that the point adds to the density map have a Gaussian form, with the maximum at the center cell, and decreasing in the neighboring cells. Because points become sparser as we move away from the camera the size of the patch is increasing with the distance. The diameter of the patch is 1 cell (50mm) at the nearest range and increases up to 6 cells (300mm) at the far range (20m). The amount by which the patch increases was determined empirically by testing the system in various scenarios. A better approach would be to consider a probabilistic error model for stereo reconstruction and to compute the required patch size from it. Also, at the far end of the detection range, there exists the risk of multiple persons being grouped together, because of the large patch size.

Because the influence of the 3D points on the density map is cumulative, the density map will contain large values in areas with a high density of 3D points. Segmentation is performed on the density map, using a region growing algorithm in order to determine possible object candidates. The region growing threshold is based on the total amount of 3D reconstructed points. This allows the segmentation to adapt to the situations where the reconstruction is less than perfect.

The result of the segmentation is a list of object hypotheses on the density map. 3D boxes are generated based on the position and size of the object hypotheses in the density map.

The detector’s resilience to occlusions depends mainly on the amount of occluded area. Usually, if at least 50% of an object is visible then the density map approach is able to detect it. Of course, it rather difficult, by any imaginable approach, to detect completely occluded objects. The tracking module is able to propagate the objects for a few frames if they become temporarily occluded.

Usually, our detector is able to segment individual pedestrians from pedestrian groups in most cases. In the
Figure 3. Object detection using density map: 
a) Greyscale image b) Density Map c) 3D points projected on the xOz plane

Figure 4. Points grouping and density map objects left camera and top view

area considered for detection we found that we are also able to detect small children.

Figure 3, shows the results of density map based object detection. The top-right cluster is caused by points which are outside the considered region. Because of this, they are not considered when the density map is build. In this particular scenario, these points were caused by bad reconstruction (they “slipped” to the far range). The large, dark-gray boxes are the result of a different object detection algorithm, suitable for large objects such as car [1].

5. Object Merging

In order to take advantage of the best characteristics of both object detection algorithms (the points grouping algorithm and the density map based algorithm) we developed an algorithm for object merging.

5.1. Object Overlap Computation

The first step of the algorithm is to compute 3D objects overlap. We consider objects to be right convex quadrilateral prisms (the base is a convex quadrilateral, and the joining edges between the bases are vertical). To compute the overlap between two objects we must compute the intersection of the two prisms. We first compute the intersection between the prisms' bases. We use a generic convex polygon intersection computation algorithm, based on the Sutherland-Hodgman polygon clipping algorithm. The base quadrilateral of the first object is clipped against the base quadrilateral of the second (see 5). If the intersection is void, then the clipping is also performed in reverse (it could be that the clipping object is included in the clipped object, in which case the clipping would be void). The clipped polygon represents the common base, shared by the two objects. The next phase is to compute the 1D segment intersection of the segments represented by the lateral edges of the 2 right prisms representing the objects. The final step is two compute the 3D polyhedron intersection volume of the two objects. This is done by multiplying the 2D common base area by the 1D height intersection.

5.2. Object Merging and Hierarchy Building

Merging the two types of objects presents a few practical problems. The most important of these problems is the fact that the density map approach may artificially split a large object into 2 meaningless small objects. In order to eliminate this problem, we designed and implemented a hierarchical object list approach. The first step is to compute the associations between the 2 lists of objects.

We use the following algorithm to compute the associations of two lists of objects, based on object intersections. The first step is to compute the intersections of each object from the first list with each other object
in the second list. Let \( V_1 \) be the volume of the first object, \( V_2 \) be the volume of the second object and \( V_3 \) be the volume of the intersection. We define the overlap of the first object by the second object \( O_{1<2} \) as \( V_1 / V_3 \) and the overlap of the second object by the first object, \( O_{2<1} \) as \( V_3 / V_2 \). If \( O_{1<2} > 0.6 \) then we consider that \( O_1 \) is a part of \( O_2 \) and if \( O_{2<1} > 0.6 \) then \( O_2 \) is a part of \( O_1 \). A special case is when both overlaps are greater than 0.6. This means that the two object overlap each other, and are very likely to represent the same object.

If both objects overlap each other, then a single object is generated and included in the output. If one object overlaps another, but the reverse is not true, then the first object is considered the parent and the other the child. Non-associated (non-overlapped) objects from both lists are retained. Following the classification stage, the children of an object are only retained if their class is different from the class of the parent object.

6. Object Tracking

For tracking vehicles we used the algorithm described in [7], which was developed for urban ACC functions and is therefore best suited for tracking vehicles. However, this algorithm is not well suited for tracking other types of objects, such as infrastructure elements (poles, walls, trees) or pedestrians. For these object types we developed a different tracking algorithm.

First, the ego vehicle motion is estimated, using a yaw rate sensor (the serial production sensor incorporated in the ESP system), a speed sensor (again, the serial production sensor) and knowing the time stamp of the previous and current frame. As the yaw rate sensor is noisy, we track its output using a Kalman filter, for more stable results.

We are thus able to obtain a rotation matrix and translation vector, which together describe the way in which the ego vehicle has moved relative to the fixed reference frame, from the time of the previous frame to the time of the current frame. Alternatively, the translation and rotation matrix describe how objects (considered fixed relative to the fixed scene) are moving relative to the ego vehicle.

Second, we consider that each pedestrian hypothesis moves in a straight line, at constant speed. We are thus able to predict their position relative to the ego vehicle. Of course, pedestrians do not always move in straight lines, but this is a good first order approximation. We also consider that fixed obstacles remain stationary.

The association phase is the most difficult and sensitive step. In order to make the association more resilient, we used image based validation for matching targets from the previous frame with targets in the current frame. The images used for matching are depth-masked (i.e., only the points for which 3D information is available and which, according to this 3D information, belong to the object’s foreground) are used. Also, because of the 3D information available, we are able to scale the images (in order to compensate the zoom-in effect caused by the ego vehicle motion). Image validation is performed on depth masked, scaled down object images, using SAD as a similarity measurement.

Unfortunately, most association errors occur when object trajectories cross. In this case, one object usually occludes the other, and we are left with very few unoccluded pixels to perform the image based validation. Furthermore, pedestrians tend to be similar in appearance. As a further improvement, we used the method of bipartite graph matching, described in [20], to find globally optimal associations.

7. Motion Signature and Motion Periodicity

In this section we present the motion signature and motion periodicity features computation. These two features are useful for discriminating between pedestrians and other object types. The motion signature represents the variability in the motion vector field associated with a given object. The motion periodicity feature represents the periodicity of this variation. As these features make sense only for moving objects they are not computed for stationary objects.

The first step in computing the motion field vectors of an object is to eliminate the global object motion by using the tracking information and to mask object background points (by using the 3D information obtained by stereo reconstruction). Next we compute the optical flow for foreground points. We tried various methods for computing the optical flow, based on brightness constancy constraints such as those described in [21], [22], [23] and block matching. We also tried methods based both brightness and depth as described in [24]. Unfortunately when the ego vehicle is non-stationary, the radial optical flow field components generated by the motion of the ego vehicle varies greatly from the center of the image to its edges. Also, because of imperfect tracking, global object motion cannot be totally eliminated.

The methods described in [21] and [22] do not yield good results because they are unable to estimate sufficiently large motion vectors. The method described in [24] doesn’t seem to increase the precision of the optical flow computation, because the range data used to form the linear depth constancy equation is too smooth to be useful.
Figure 6. Motion Signature and Motion Periodicity

Consequently, only two methods for optical flow computation are useful for our environment: block matching and the pyramidal approach described in [23]. Block matching gives good results, but is prohibitively computational expensive. Therefore, we used the pyramidal approach described in [23]. This approach has the advantage that it works across a large range of displacements. It also computes the optical flow only where it can be recovered exactly, at image corner points.

Using the 3D coordinates associated with 2D points we transform the 2D optical flow vector field into a 3D field. We compute the variance of this field in the horizontal plane and obtain our “motion signature” feature. This motion signature is usually large for moving pedestrians and small for other moving objects such as cars. However, incorrect optical flow vectors sometimes generate spurious large motion signatures for non-pedestrian objects. Fortunately, these large signatures occur only as spikes and are not periodic.

In order to characterize the periodic vs non-periodic nature of the motion signature generated by an object, we use Welch’s averaged modified periodogram method of spectral estimation [25] to compute the frequency spectrum. Frequency spectra corresponding to periodic pedestrian motion are typically band limited, while for other types of objects is not. This is because noise occurs with equal probability in all the spectrum, while the periodic motion caused by swinging of pedestrians’ arms and legs during walking occurs at some fixed frequencies. Figure shows in a) the ROC of a simple pedestrian/non-pedestrian classifier based on the motion signature feature, b) shows a (non-periodic) time plot of the motion signature for a car and c) the periodic motion signature of a pedestrian).

<table>
<thead>
<tr>
<th>Class</th>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>ROC Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Truck</td>
<td>0.894</td>
<td>0.063</td>
<td>0.879</td>
<td>0.977</td>
</tr>
<tr>
<td>Car</td>
<td>0.877</td>
<td>0.071</td>
<td>0.856</td>
<td>0.968</td>
</tr>
<tr>
<td>Ped.</td>
<td>0.909</td>
<td>0.048</td>
<td>0.869</td>
<td>0.974</td>
</tr>
<tr>
<td>Pole</td>
<td>0.814</td>
<td>0.01</td>
<td>0.844</td>
<td>0.978</td>
</tr>
<tr>
<td>Other</td>
<td>0.706</td>
<td>0.096</td>
<td>0.797</td>
<td>0.895</td>
</tr>
</tbody>
</table>

8. Object Classification

The most important part of classification is having enough data for training and testing. We built a large database containing cca 25000 objects (approximately 1 hour of image sequences) together with their features and their class (obtained by manual labeling). In order to generate all the classifiers for all three classification stages (dimension based, motion and dimension based and top level) we used the WEKA environment. Because WEKA generates Java implementations for classifiers and our system is written in C/C++ for real time performance, we created a Java virtual machine inside our application which executes the Java based classifiers.

We found that the best performance for our classifiers was obtained using decision trees. We used 66% of the dataset for training and the remaining 34% for testing. Table 1 presents our classification results.

Figure 7 shows some pedestrians detected using our algorithm.

Our system works in real-time at 20fps on a Pentium Core 2 at 2.6 GHz.

9. Conclusions

We developed a generic framework for object detection, tracking and classification in urban scenarios using stereovision. Our results show that our ap-
approach has a high classification rate for the five classes we considered (trucks, cars, pedestrians, poles and other objects). The object classification results may be used for further processing such as occupant cell detection, pedestrian collision risk assessment or traffic light/traffic sign recognition. In order to have a production system we must further improve our classification rate. Our flexible framework allows us to integrate additional features and classifiers to achieve this purpose.

References


