**Preceding Vehicle Tracking in Stereo Images via 3D Feature Matching**

Daniel Weingerl, Wilfried Kubinger, Corinna Engelhardt-Nowitzki

*UAS Technikum Wien: Department for Advanced Engineering Technologies, Höchstädtplatz 6, 1200 Wien, Austria*

**Abstract**

The reliable detection of the preceding vehicle position is a very important component of most semi-autonomous cars. However, most vehicle detection systems rely on expensive sensors or need a big amount of training images. In this work a vehicle tracking system, which is using images from a stereo camera system, was developed. In the first part a state of the art stereo matching algorithm was chosen based on the operating environment of the vehicle. The depth information from the stereo algorithm is then transformed into a point cloud. In the second part of this work an object detection module, which detects the preceding vehicle in the point cloud, was developed. The algorithm is based on 3D feature detection and matching. The 3D pose of the tracked vehicle is determined directly via the position of the detected vehicle in the point cloud. The module was tested with several test sequences and it has good tracking results in off-road situations with trucks as tracked vehicles.

**Keywords:** Object Detection; Stereo Vision; Point Cloud; Feature Matching; Pose Estimation


DOI: 10.2507/26th.daaam.proceedings.140
1. Introduction

Semi-autonomous convoys can be used to save lives and expensive equipment in war or crisis areas. In these convoys only the first vehicle needs to be armored and controlled by a human driver. All the other vehicles follow their preceding vehicle. To realize such an autonomous convoy, a cheap and reliable vehicle tracking system is needed.

In this work a reliable, real-time capable tracking module had to be developed. The program should use an existing stereo camera system and had to be implemented as a “Robot Operating System” program. It should detect the preceding vehicle via the information of the stereo camera system without using markers. The algorithm must work reliable in different environmental conditions. The tracking system has to return the current pose of the tracked vehicle (coordinates and roll, pitch and yaw angle) and it should only track a specific vehicle.

The convoy will be driving in rough terrain. Therefore the first problem to solve was to find a stereo algorithm, which is robust against vibration and which is working under different environmental conditions. Also the object recognition method has to work under these conditions. It has to be robust against different lighting conditions and should detect the vehicle, even if it is partially hidden.

1.1. Materials and methods

Two cameras from Imagingsource (type DFK 22BUC03) with lenses from Computar (T0812 FICS-3) were used to test the stereo matching algorithm. The tracking and the stereo matching part of the program were tested using test sequences from the project partners. Fig. 1 shows an example image from one of the sequences. Furthermore, test sequences from the Kitti project [1] were used to test the program.

![Fig. 1. Example of a test sequence](image)

The program was realized as a ROS-Node for the “Robot Operating System” [2]. For the stereo matching part of the program, functions and classes from the OpenCV library [3] were used. The “Point Cloud Library” (PCL) [4] was used for visualization and object recognition tasks.


2. Realization of the tracking program

A C/C++ program, which handles the stereo matching and the vehicle tracking, was created.

2.1. Stereo matching module

The first part of the program calculates the depth information from the stereo images and transforms the data into a point cloud. The program reads the synchronized stereo images and the camera calibration matrices. The images are then undistorted and rectified. Next the program calculates the disparity image. For this part of the program a stereo algorithm had to be chosen. The three algorithms, which are implemented in the OpenCV library were taken into account. The “Semi Global Block Matching” (SGBM), the “Block Matching” (BM) and the “Variational Matching” (VM) Algorithm were tested and compared with each other.

The results from the VM algorithm were not suitable for this project. The other two algorithms had good matching results and were tested further (see Fig. 2). The BM algorithm was faster, but it only calculated disparity values on edges of objects and on objects with high texture. The SGBM algorithm calculated a more complete disparity image. Based on the test results, the SGBM algorithm from [9] was chosen as the final algorithm for this project. The main reason for this decision was, that the BM algorithm computes incomplete depth information of the truck, which needs to be tracked, because the back of the vehicle is not textured.
After computing the disparity image, the image is projected into 3D space. The projected image is then converted into the point cloud format of the PCL. The created point cloud also includes the intensity or RGB values from the camera image, which can be used by the object recognition module to compute more distinct keypoints. Fig. 3 shows a visualization of the resulting point cloud.

2.2. Tracking module

The second part of the program reads the point cloud, tracks the preceding vehicle and returns the current pose of the vehicle. Fig. 4 shows the main steps of the object recognition process.

The program needs a model of the object, which will be tracked. The model has to be provided as a point cloud with intensity or RGB information, saved as a file in the PCL ASCII format. The model can consist of the whole 360° view of the vehicle or only one side of it. The accuracy of the pose estimation is higher, if the model only consists of one side of the vehicle. But if the vehicle has to be tracked through steep curves, a full model is needed.

After reading the model point cloud, the program performs the following steps: First keypoints are detected in the model point cloud. Then descriptors for the keypoints are calculated. After computing the model features, the initialization of the program is completed. Now the program starts processing the scene point clouds. The keypoint detection and description steps are performed for every frame of the scene point cloud. After that, the program searches for corresponding keypoints between the model and the scene. Next the vehicle is detected using the correspondences. At the end of the detection process, a fitness score, which is mainly based on the euclidean distances between the model and the scene points, is calculated. If this fitness is above a configurable threshold, the object pose is estimated and returned by the program.
2.2.1. Algorithms and methods

For every phase of the program (e.g. keypoint detection, description) several algorithms were considered. The different algorithm combinations were evaluated via a score, which included the fitness score, the average calculation time of the whole process and the percentage of frames, in which the program could not estimate the vehicle position. First a keypoint detection algorithm had to be chosen. The following PCL implementations of keypoint detection algorithms were tested:

- Harris 3D
- Intrinsic Shape Signatures (ISS)
- SIFT
- SUSAN
- Harris 6D

The edge and/or corner based algorithms, especially the Harris 3D and SUSAN [10] detectors, had the best results. Fig. 5a and b show the detected keypoints of both algorithms. Table 1 shows the test results of the four best performing keypoint algorithms.

<table>
<thead>
<tr>
<th>Keypoint Type</th>
<th>Descriptor Type</th>
<th>Average Time / Frame (s)</th>
<th>Fitness</th>
<th>Median Fitness</th>
<th>False Negatives (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harris</td>
<td>ColorSHOT</td>
<td>4.80092</td>
<td>0.96629</td>
<td>0.991548</td>
<td>0</td>
</tr>
<tr>
<td>SUSAN</td>
<td>ColorSHOT</td>
<td>2.45064</td>
<td>0.946</td>
<td>0.967549</td>
<td>3</td>
</tr>
<tr>
<td>ISS</td>
<td>SHOT</td>
<td>3.7128</td>
<td>0.91079</td>
<td>0.970384</td>
<td>22</td>
</tr>
<tr>
<td>SIFT</td>
<td>SHOT</td>
<td>8.54802</td>
<td>0.83943</td>
<td>0.921743</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 1. Comparison between different keypoint detection algorithms.

The object detection module had good matching results with the Harris algorithm, because it was finding enough keypoints on the back of the tested truck. The matching results with the SUSAN algorithm were less precise than the results when using Harris keypoints (see fitness score), but the calculation time was reduced by one third. To combine the advantages of both algorithms, the Harris and SUSAN keypoints are fused. Therefore a concave hull is wrapped around the SUSAN keypoints and inliers from the Harris keypoints are added to the SUSAN keypoints. The resulting SUSAN/Harris keypoints are shown in Fig. 5c.

For the keypoint description several descriptor algorithms were compared to each other. The ColorSHOT descriptor from [11] had the best results. This descriptor is a combined texture and shape descriptor. It computes a signature of shape and texture histograms from the surroundings of the keypoint. The ratio between shape and texture information is configurable.

In the next step the tracking program searches for corresponding keypoints between model and scene. For this process a k-d tree, which contains the descriptors, is generated. Then a nearest neighbor search for the most similar scene keypoint is performed for every model keypoint. The resulting correspondences are filtered by a one-to-one rejector and by rejecting correspondences with a high nearest neighbor distance ratio (NNDR). The NNDR rejection filters out correspondences, which are not distinct.

A RANSAC algorithm is used to determine the transformation between the model and its position in the scene point cloud. The found position is refined by fine registration via an ICP algorithm. Then the model position is validated by calculating a fitness score. This score is based on several parameters. Some of these parameters are:
- The euclidean distance between the model and scene points
- The roll, pitch and yaw angles related to the ground plane position
- The number of found correspondences
- The average similarity of all descriptor pairs

If the fitness is above a threshold, the object pose is determined using the transformation matrix from the RANSAC algorithm, which describes the transformation between the model and its estimated position in the scene.

2.2.2. Optimizations

Several optimizations, which solve the main issues of the tracking process were implemented. One of these optimizations is an angular constraint for the SAC algorithm. This constraint was implemented, because there were several cases, were the vehicle was found in a wrong position. There were three main cases, which lead to a wrong detected angle:

- The model was flipped vertically, if the keypoints were only detected in the lower area of the vehicle
- The model was flipped horizontally, if the keypoints were only found in symmetrically identical areas
- The roll angle estimation was wrong, if the keypoints were only found in the center of the model

A roll, pitch and yaw angle range for the SAC algorithm was implemented to solve these issues. The angular values of this range were determined empirically.

To decrease the calculation time of the object recognition process, the program searches for the ground plane via a RANSAC search with a planar model. The biggest plane found in the point cloud is assumed to be the road surface and only points above this plane are used for the object detection process.

The last major optimization is the tracking by using a Kalman filter. If a valid position is found by searching the whole point cloud, the next position of the vehicle is estimated by a Kalman filter. In the subsequent frame the object recognition algorithm only searches in a cuboid volume at the position estimated by the Kalman filter. If no object is detected in this area, the search is expanded to the whole point cloud. The calculation time of the recognition process was reduced by 75% through the Kalman filter.

3. Results

The tracking program was evaluated with several test image sequences.

3.1. Stereo matching module

First the SGBM algorithm used in this project was compared to other stereo algorithms. Fig. 6 shows the resulting disparity images from the SGBM algorithm and the algorithm used by one of our project partners. The algorithm from the project partner only calculates disparities for the edges of the vehicle (red rectangle) and at areas with strong texture, whereas the SGBM algorithm calculates a more complete disparity image. However, the algorithm of the project partner reproduces finer details of the structure of objects.

![Fig. 6. Camera image (a) and disparity images of the algorithm from the project partner (b) and the SGBM algorithm (c)](image)

Fig. 6. Camera image (a) and disparity images of the algorithm from the project partner (b) and the SGBM algorithm (c)

Fig. 7 shows a comparison between the SGBM algorithm and the Velodyne laser scan from the test data of the Kitty project. In this case the algorithm detects the shadow casted by the car as a part of the car (green rectangle). Furthermore, some disparity values are calculated wrong (red ellipse). The evaluation of the stereo algorithms has shown, that the considered stereo algorithms do not calculate the disparity values precise and/or complete. They only provide coarse structural information about the objects. Therefore, a feature descriptor, which includes texture information (e.g. ColorSHOT) has to be used to detect an object in the point cloud reliable.
Fig. 7. Comparison between the camera image (a) and the results from the SGBM algorithm (b) and a laser scan (c)

3.2. Tracking module

The tracking module was evaluated with several test sequences. There was no ground truth data available for the test images, the comparative vehicle pose was evaluated manually. Fig. 8 shows an example of a found match of the tracking module. The green lines show the correspondences between the model and the scene.

Fig. 8. Found match of the tracking module

Fig. 9a shows the euclidean distance between the real object positions and the positions estimated by the tracking module. The green line shows the relative distance related to the distance between the vehicle and the camera system, the blue line shows the absolute deviation. The diagram shows, that the positional error increases proportional to the object distance.

Fig. 9b shows the absolute angular error of the estimated object pose. The tracking module has problems in estimating the roll angle of the vehicle. The calculation time per frame is shown in Fig. 9c. The recognition process is
faster, if the distance between the vehicle and the camera system is bigger. The process is faster at high distances, because less keypoints are found on the vehicle, if it is farther away from the camera system. In frames 49 and 83 the program was unable to find the vehicle position, which lead to a search in the whole point cloud and to a higher calculation time. The position loss in frame 49 was caused by an intentionally added position jump of the vehicle.

The tests above were carried out with the test sequence shown in Fig. 1. The algorithm produced no false positives or negatives in this sequence. However, the tracking module had problems with many of the test sequences. It is not suitable for vehicles with reflective surfaces. Vehicles with a small back plane, like compact cars can only be tracked, if the distance between the vehicle and the camera system is below approximately 15 meters. This distance depends on the car type.

The average calculation time of the algorithm is 0.86 seconds. The algorithm has a mean false negative rate of 2%. The false positive rate is strongly depending on the type of vehicle and the environment. In off-road situations without traffic and vehicles with distinctive features the false positive rate is below 1%. In urban areas with many similar vehicles the false positive rate can be above 50%.

4. Conclusion

In this work an object tracking module based on 3D feature matching was developed. Furthermore, an established stereo matching algorithm, which can be used for object recognition tasks, has been chosen and implemented. The tracking program is suitable for the specified task. It can be used to track large vehicles with distinctive features in non-urban areas. As the average calculation time is nearly one second, the tracking method should only be used to track objects without rapid movements. The program could also be used as a validation module for validating the position estimated by another tracking program.

In future the tracking program could be extended by a tracking module, which estimates the coarse position of the vehicle by using a different object recognition method, which is optimized for urban areas. The module proposed in this paper could then be used to refine this coarse position. By combining two tracking algorithms, the tracking could also be used in urban areas. The second tracking system could also be marker-based.

Because the noise and low accuracy of the stereo processing algorithm is the main limitation to this tracking method, the stereo camera system could be replaced by a more reliable and accurate system for generating 3D data (e.g. a Velodyne laser scanner). However, this would lead to a higher cost of the system.

5. Acknowledgements

This project has been funded by the Austrian Security Research Programme KIRAS – an initiative of the Austrian Federal Ministry for Transport, Innovation and Technology (bmvit).

6. References