Vehicle pose estimation using stereo information

Darwin Vickers
Summer student
NICTA CRL
(Supervisor: Xuming He)

2015

1 Introduction

Vehicle pose estimation could be a useful component of autonomous vehicle or computer assisted driving systems. Stereo camera equipment is relatively affordable and can feasibly be mounted on a moving vehicle. This paper assesses the feasibility of using stereo information to assist in vehicle pose estimation, and establishes one possible pipeline for such a system.

Stereo images can be used to generate a 3D point cloud representation of a scene, effectively serving the same purpose as a LIDAR scanner or RGB-D camera, albeit with less accuracy. Many techniques have been, and continue to be, developed to achieve this task, with some designed especially for road scenes.

Once computed, the point cloud data can be used in conjunction with 2D vehicle recognition to provide an estimate of the location and pose of vehicles in the scene. An outline of the whole process is shown in Figure 1.

This project focuses on two components of the process: depth mapping from stereo images, and pose estimation. The investigation of stereo depth mapping consists of implementing and evaluating an algorithm recently proposed by Einecke & Eggert [1]. The second part involves the design, implementation and evaluation of an algorithm for computing vehicle pose estimation, using 2D labels and 3D point cloud data as input.

2 Stereo scene reconstruction

2.1 Approach

Einecke & Eggert's stereo matching algorithm, which they call "Relaxed block matching" or RBM, is presented in [1]; its core innovation is to relax the assumption, present in typical block-matching algorithms, that all surfaces are oriented to face the camera - which is especially inaccurate in typical images taken from vehicle-mounted cameras, where the road is heavily slanted. The core of the algorithm was implemented as detailed in the original paper; however, one post-processing step involving "removal of small disparity segments" was not included – it was not explained in enough detail to be surely recreated.

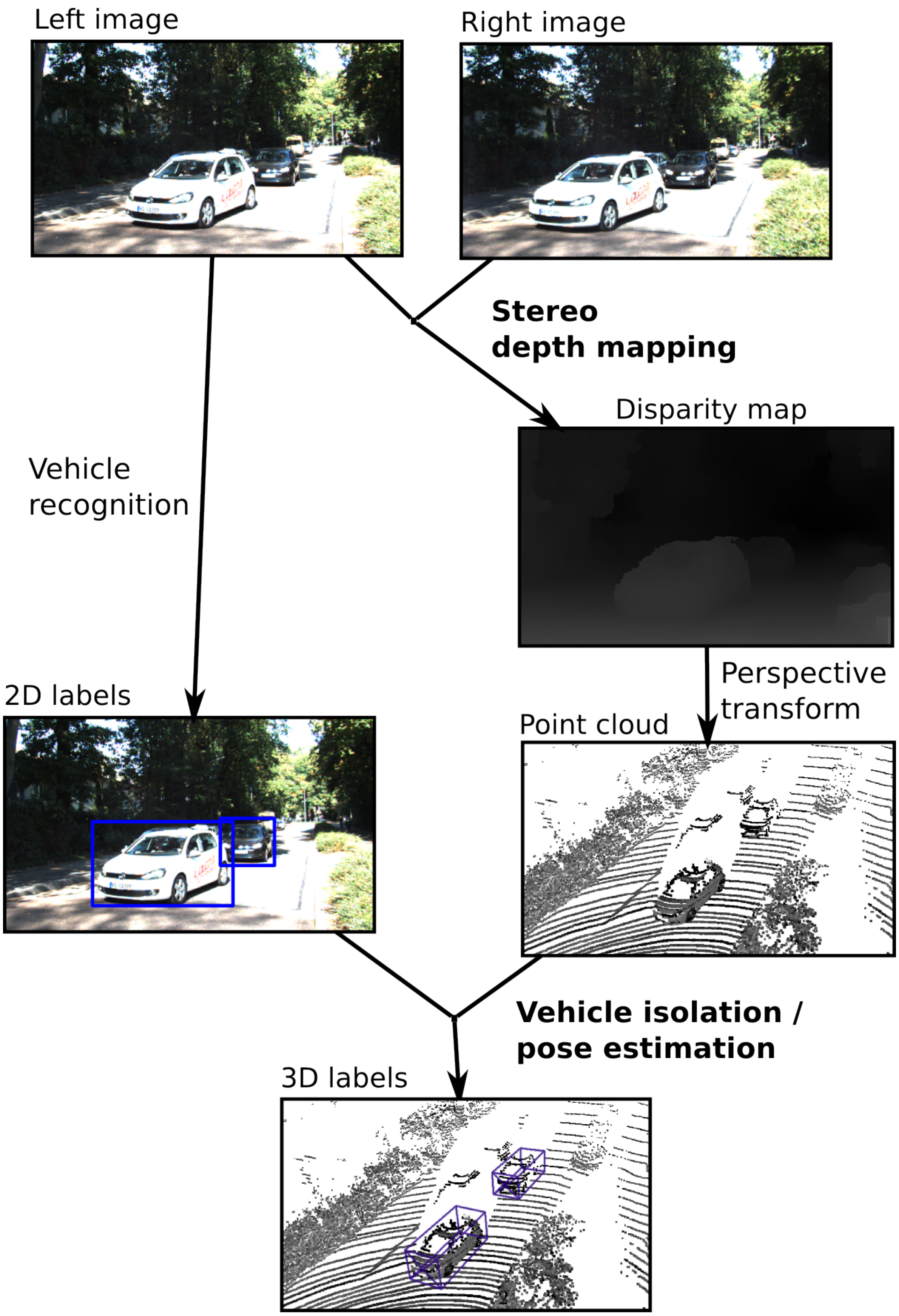
Both standard block matching and RBM are compatible with a range of scoring functions. For simplicity's sake, we use census transform (CT) with both algorithms; out of the four scoring functions tested by Einecke & Eggert, they report the best results with this one.

2.2 Evaluation

The results of RBM are compared with the 'baseline' block-matching algorithm without the fronto-parallel relaxation. The dataset used is the KITTI Stereo dataset [5]. We only use the 'training' component, which consists of 194 stereo image pairs with corresponding disparity maps. Of these, the first 50 are used for determining best parameters, while the remaining 144 are used for evaluation.

Evaluation consists of finding the average percentage of 'bad' pixels – that is, pixels whose given disparity value differs by more than 3 pixels from the ground truth value.

Figure 1: Outline of process to go from stereo images to 3D pose information. The steps in bold are the focus of this research.



Before comparing the two algorithms, it was necessary to determine the best parameters for each. The CT block size, and blur parameters used in pre- and post-processing, are fixed to match the parameters used by Einecke & Eggert. The threshold used in the left-right check is not recorded in Einecke & Eggert's paper; we use a value of 2 pixels. The only remaining parameter for standard block matching is the block size, while RBM requires this as well as an additional 'penalty factor' parameter. These parameters are determined experimentally. Block sizes of 5, 7 and 9 are tested, and for RBM penalty factors in the range [1.5,1.6,...,2.5] are tested. Evaluation of performance on 50 images (using the 'bad pixels' metric detailed above) suggests that a block size of 7 is most effective for both algorithms, and that the best penalty factor for RBM is 1.6. These are the parameters used in evaluation. Before processing, the images are scaled down by a factor of two, in keeping with Einecke & Eggert's method.

2.3 Results

Over the 144 images used, regular block-matching produces 10.40% bad pixels on average. RBM produces 10.20% bad pixels on average. The improvement is minor but noticeable.

In contrast to these results, Einecke & Eggert report that their implementation of regular block-matching produces 6.34% bad pixels on average. The reason for this difference is most likely the extra small segment removal included in their post-processing. Einecke & Eggert do not report a figure for RBM using census truth, but they provide a plot showing that the level of improvement is roughly 0.2%, corresponding with our results.

3 Pose estimation

3.1 Approach

Here we present a method for determining vehicle pose, given point cloud data and a 2D region of interest. This algorithm can be used with point cloud data derived from stereo information, or with data directly from a LIDAR scan or other method.

The first step is the extraction of the points which belong to the surface of the vehicle of interest. Firstly, the 2D region is extended by a factor of 5%, to ensure that all points on the vehicle are included.

The region is projected into 3D and all points lying within this volume are extracted. This set of points will contain extraneous data along with the vehicle points, for example, sections of road or other objects in front of or behind the desired vehicle. We make the assumption that other objects will be physically separated from the vehicle, and apply Euclidean cluster segmentation to the point set, taking the largest identified cluster to be the vehicle. Since the road is in contact the vehicle, portions of road may still be present in the extracted cluster. The solution is to remove a thin layer of points from the base of the cloud.

The core of the algorithm lies in the next step - determining an oriented 3D bounding box which ideally reflects the orientation of the vehicle. To make this achievable, it is assumed that the vehicle is upright; i.e., only the rotation around the z (height) axis is considered to be non-zero. When finding the z-rotation, point height is irrelevant, and so the problem is reduced to finding a 'birds-eye view' 2D bounding box around the points.

One obvious approach is to find the minimum-area oriented bounding box. This works quite well, and the minimum-perimeter box is even better, but neither is ideal. There are cases where, although the car is close to 'box-shaped' in the x-y plane, the orientation of the minimum-area or minimum-perimeter box does not correspond to the orientation of the car. Instead of area or perimeter, we use an alternative cost function to be minimised: the sum of distances (in the x-y plane) from each point to the nearest edge on the bounding box. The intention is to ensure that the points along the edges of the car are aligned with the edges of the bounding box. Indeed, initial testing shows that this metric is more effective than area or perimeter.

Once the 'birds-eye view' 2D bounding box has been found, we must determine which pair of sides is the front and back of the car, and which pair is the sides. This is done by comparison with 'model' car dimensions. When a vehicle is parallel or perpendicular to the camera and far away, it is common that only one face of the car is captured accurately. So both side lengths are compared to both the model width and model length. Out of the four possible correspondences, the best one is used to determine the alignment of the car. The bounding box is reshaped to match the model dimensions. The corner nearest the camera is assumed to be the most accurately positioned, so it is kept in place while the other corners are moved accordingly.

The final step is simply to extrude the bounding box into 3D, using the minimum and maximum points along the z-axis.

3.2 Limitations

The algorithm does not distinguish between opposite orientations: the rotation returned is in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

The algorithm is designed for situations where the road is perfectly flat. When either the camera or the vehicle being examined is on a steep hill, the x- and y- axis orientation is neglected and the quality of the result will degrade.

The reported dimensions of the 3D bounding box are just the 'model' dimensions; they do not reflect the actual dimensions of the car. Cars with unusual dimensions may not be represented accurately.

3.3 Evaluation

This evaluation has two aims: assess the performance of this pose estimation method, as well as determine its feasibility in a stereo vision pipeline. For this reason, evaluation is performed separately with two different sets of input.

The first set uses 'ideal' input: accurate point-cloud data taken from LIDAR scans, and manually created object labels. The intention is to evaluate the pose estimation independently of the quality of its input. The second set uses 'realistic' data that could be generated using only a stereo camera setup: point clouds are generated from stereo images using the SPS-Stereo algorithm [7], and labels are generated by the Latent SVM object detector as implemented in OpenCV [6], trained on the Pascal VOC 2007 dataset [2].

The base data is taken from the KITTI Object dataset [4]. We only use the 'training' component, which consists of 7480 images with corresponding 2D labels and 3D bounding box information.

A core part of evaluation is determining whether a detection is considered to match a given ground truth bounding box. Two boxes are counted as matching if the volume of intersection divided by the volume of union is greater than 50%. Before evaluation can be performed, it is necessary to find a correspondence between detections and actual objects. Using the 'ideal' data as input, the correspondence is trivial as the output data are in the same order as the input data. However, with the 'realistic' evaluation, it is necessary to compute the best correspondence by maximising the sum of 'intersection over union' scores (using the Hungarian Algorithm).

Results are reported according to three separate metrics. The first, orientation error, only considers the absolute angular deviation. The second, localisation error, considers the error in position, measured from the centre-base of the bounding box. The localisation error is normalised by the volume of the vehicle's bounding box. These two errors are averaged over all the cases where the detection is considered correct.

The third metric differs between the two evaluations. For the evaluation with 'realistic' input, we report the F1-score. Detections which correspond to vans are not counted as false positives. For the 'ideal' evaluation, there is no potential for erroneous detection, so false positives are insignificant. In this case only the recall value is reported.

All three metrics are calculated for the three different 'difficulty levels', as used by the KITTI evaluation. It should be noted that each category is a superset of the 'easier' categories.

3.4 Results

Table 1. Results of pose estimation algorithm with ‘ideal’ input

Difficulty level	Orientation error (degrees)	Localisation error (metres)	Recall
Easy	1.49	0.430	0.823
Medium	1.45	0.427	0.557
Hard	1.45	0.428	0.363

Table 2. Results of pose estimation algorithm with ‘realistic’ input

Difficulty level	Orientation error (degrees)	Localisation error (metres)	F1 score
Easy	1.99	0.408	0.0925
Medium	1.94	0.399	0.0624
Hard	1.97	0.401	0.0508

The results for the ‘ideal’ evaluation are quite good, implying that the pose estimation algorithm itself is effective. However, the quality degrades significantly when imperfect input is used. The main issue is the point cloud derived from stereo. Although it gives a good representation of large-scale 3D information, the cloud is quite inaccurate at a smaller scale, which makes it difficult to extract the car from its surroundings. The bounding box algorithm relies on the car being extracted cleanly, and so the reported bounding box data is quite inaccurate.

It is worth noting that despite precision and recall being low in the ‘realistic’ evaluation, the results for orientation and localisation are still quite good.

4 Conclusion

The relaxed block-matching stereo method proposed by Einecke & Eggert improves on the standard block-matching algorithm when used in road scenes, but the improvement is minor.

The proposed algorithm for pose estimation proved to be effective when used with ideal input, but did not adapt well to imperfect data. However, in systems where accurate point cloud data is available, this method may still be a useful tool.

For the proposed pipeline to be effective in real-world scenarios, more work is needed in one or both of two areas: improving the stereo-to-disparity process, or adapting the pose estimation algorithm to deal more effectively with ‘messy’ input. Development in either of these areas is feasible.

References

[1] Einecke, N. & Eggert, J. *Block-Matching Stereo with Relaxed Fronto-Parallel Assumption.* 2014 IEEE Intelligent Vehicles Symposium (IV)

[2] Everingham, M., Van-Gool, L., Williams, C., Winn, J., & Zisserman, A. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007)* Available at [`http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/`].

[3] Felzenszwalb, P., Girshick, R., McAllester, D., & Ramanan, D. *Object Detection with Discriminatively Trained Part Based Models.* PAMI, vol. 32, no. 9, pp. 1627-1645, September 2010.

[4] Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. The KITTI Vision Benchmark Suite (Object). Available at [`http://www.cvlibs.net/datasets/kitti/eval_object.php`].

[5] Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. The KITTI Vision Benchmark Suite (Stereo). Available at [`http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php`].

[6] OpenCV, 2014. *Discriminatively Trained Part Based Models for Object Detection.* Based on method proposed in [3] Documentation at [`http://docs.opencv.org/modules/objdetect/doc/latent_svm.html`].

[7] Yamaguchi, K., McAllester, D., & Urtasun, R. *Efficient Joint Segmentation, Occlusion Labeling, Stereo and Flow Estimation.* European Conference on Computer Vision (ECCV), 2014. Code available at [`http://ttic.uchicago.edu/~dmcallester/SPS/spsstereo.zip`].