

3D Reconstruction of Vehicles for Traffic Surveillance

Hanna Källén, Håkan Ardö and Olof Enqvist
Mathematical Imaging Group, Lund University
Email: {hannaka, ardo, olofe}@maths.lth.se

Abstract—To improve traffic safety it is important to evaluate the safety of roads and intersections. Today this requires a large amount of manual labor so an automated system using cameras would be very beneficial. We focus on the geometric part of the problem, that is, how to get accurate three-dimensional data from images of a road or an intersection. This is essential in order to correctly identify different events and incidents, for example to estimate when two cars gets dangerously close to each other.

The proposed method uses a standard tracker to find corresponding points between frames. Then a RANSAC-type algorithm detects points that are likely to belong to the same vehicle. To fully exploit the fact that vehicles rotate and translate only in the ground plane, the structure from motion is estimated using an optimization approach based on the L_∞ norm. The same approach also allows for easy setup of the system by estimating the camera orientation relative to the ground plane. Promising results for real-world data are presented.

I. INTRODUCTION

To reduce the number of road traffic injuries it is important to know how safe certain roads and intersections are. There are different ways to evaluate this. One method is to count the number of actual accidents taking place. Since accidents are rare it can take years to get a good assessment of safety this way. A faster approach is to predict the numbers of accidents that will happen by manually observing certain events, conflicts, during a much shorter time period. This is done by letting trained personnel study video of the intersection. Naturally, this is very expensive and time-consuming and an automated method would be very beneficial.

In [1] a method for automated surveillance was proposed. To calculate the position of a vehicle, the 2D image of that vehicle was projected onto the road plane. If the camera can be placed right above the intersection, this will work fairly well. In most cases though, this is not possible. The projection of the vehicle gets stretched out and the estimated position incorrect. A better way to estimate the position would be to make a three-dimensional representation of the object and use this to calculate the position of the vehicle. This is the approach considered in this paper.

There are many methods to do 3D reconstructions from images. The approach described in this paper differs from most methods in at least one important aspect. The reprojection errors are minimized with respect to the L_∞ -norm rather than the more common L_2 -norm. This makes it possible to find the global optimum and it also makes

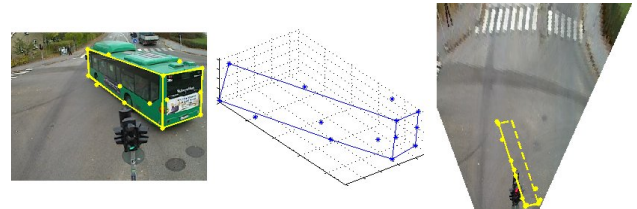


Fig. 1. Original image of a bus, a 3D model of the bus and the 3D model projected to a map of the intersection

it easier to impose extra constraints, such as the fact that vehicles are only moving in the ground plane. The presented work is a continuation of [2].

II. OVERVIEW OF THE SYSTEM

We start with captured video from an intersection. For the reconstruction we need to find corresponding points between different frames. This is achieved in the following way. Every few frames we pick an image to use as a starting image. In this image a corner detector is used to find strong corner points. These points are then tracked a few frames using a KLT tracker [3], to get corresponding image points in a later frame.

To reduce the amount of outliers in the 3D reconstruction, we need to detect which points belong to the same vehicle. First, all points that have not moved are removed. For the remaining points a RANSAC-type algorithm is performed. For two points at a time, a minimal solver finds the rotation and translation between the frames. This is used to triangulate the other points and the triangulated 3D points are projected in both cameras. Next, the reprojection errors are calculated and the inliers are counted. To be an inlier a point has to have a small reprojection error in both cameras. The solution with the highest number of inliers is chosen and the points classified as inliers are tracked. The tracker follows the points both forward and backward in time until the vehicle leaves the camera field of view or until the tracking fails.

There might be more than one vehicle in view at the same time. To handle this, points belonging to the first vehicle are removed and the algorithm is repeated to find a new vehicle. This is repeated until there are too few points left to track.

Since new starting images are chosen every few frames, we typically get several tracks from the same vehicle. To avoid counting one vehicle several times, we try to detect this by comparing the positions of image points in the

respective images. Tracks having large overlap in position are likely to represent the same vehicle.

This procedure yields long tracks of corresponding image points likely to belong to the same vehicle. From these tracks we choose a few views to perform our 3D reconstruction, which will be described thoroughly in the following sections. Having a 3D model we use this to estimate the pose of the vehicle in all views. This gives us accurate information on the movement of the vehicle throughout the sequence.

System Overview

1. Track points between frames.
2. Cluster points from the same vehicle.
3. Reconstruct the vehicle using a few views.
4. Compute the vehicle pose in all views.

III. STRUCTURE FROM MOTION

From the tracking algorithm, we get corresponding points in different frames. To reduce the amount of data, we pick just a few frames to do the 3D reconstruction. If the rotations between these frames are known, this problem can be solved optimally with respect to the L_∞ -norm, using second-order cone programming [4]. To find these rotations we can perform a branch-and-bound search through rotation space as in [5]. The theory from these two papers will form the basis for our approach, but we modify it to fit our application.

We assume that the ground is planar and that vehicles are rigid bodies moving only in this plane. Further, we assume that the camera has fixed, known internal parameters. We will show how to estimate the position, or rather the rotation, of the ground plane relative to the camera. In reality we have one stationary camera and moving vehicles, but from the vehicle point of view it looks like the camera is moving instead. Choosing this view we can use the standard notation with multiple cameras and stationary 3D points.

A. Structure from Motion with Known Rotations

We first consider the case when the rotations between different views are known. This means that we only need to find the translations and the 3D points. We do this by minimizing the L_∞ -norm of the reprojection errors, i.e. the largest of the reprojection errors.

Due to noise there is no exact solution to the reconstruction problem. Thus we allow the reprojected points to deviate somewhat from the measured image points. To get linear constraints, we allow the reprojected point within a small square around the measured image point. This differs from the approach in [4]. The simplification allows us to solve the optimization using linear programming rather than second-order cone programming.

The square has side length 2ϵ , where ϵ is the tolerated error, and is defined by four lines in the image plane, see Fig. 2. These lines are the intersections between the image plane and planes going through both the camera center and the image plane. The planes through the camera center form a generalized cone in which the 3D point

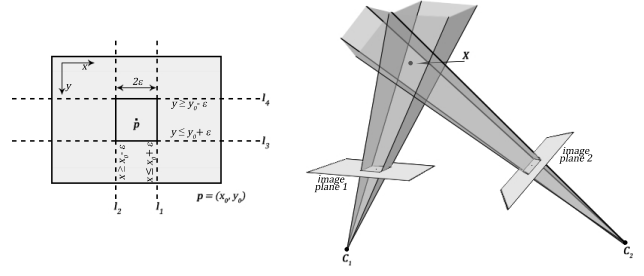


Fig. 2. Linear constraints in the image plane and in space. The left figure shows the lines that define the box in which the reprojected point must lie inside. The right figure shows constraints in space from two cameras, the 3D point must lie inside both cones.

has to be. A bigger value on ϵ results in a wider cone. For every camera, we get another cone constraint on the position of the 3D point. The constraints from two views can be seen to the right in Fig. 2.

If we express the 3D point in cartesian coordinates we can write the camera equation on the form

$$\lambda \mathbf{x} = \mathbf{K} \mathbf{R} (\mathbf{X} - \mathbf{C}),$$

where λ is the depth of the point, $\mathbf{x}$ the image point in homogeneous coordinates, $\mathbf{K}$ the camera calibration matrix, $\mathbf{R}$ the rotation of the camera, $\mathbf{X}$ the 3D point in cartesian coordinates and $\mathbf{C}$ is the camera center. For calibrated cameras we can exclude the calibration matrix

$$\lambda \mathbf{x} = \mathbf{R} (\mathbf{X} - \mathbf{C}). \quad (1)$$

From each one of the lines in Fig. 2 we get constraints on the form

$$\mathbf{l}^T \mathbf{x} \leq 0,$$

where $\mathbf{l}$ is the equation of the line and $\mathbf{x}$ is the image point in homogeneous coordinates. Using the camera equation in (1) we can rewrite the constraints to

$$\mathbf{l}^T \frac{1}{\lambda} \mathbf{R} (\mathbf{X} - \mathbf{C}) \leq 0.$$

The depths has to be positive if the point is visible in the camera, therefore we can eliminate the depth from the equation. If we set $\mathbf{a} = \mathbf{l}^T \mathbf{R}$ we get

$$\mathbf{a} (\mathbf{X} - \mathbf{C}) \leq 0,$$

where $\mathbf{a}$ is a plane going through the camera center and the image plane as shown on the right in Fig. 2.

For every 3D point there are four inequalities per camera, one for each plane. We collect all these inequalities into a matrix

$$\mathbf{A} \mathbf{u} \leq \mathbf{0},$$

where $\mathbf{A}$ holds the coefficients and $\mathbf{u}$ holds the unknowns, being the 3D points and the camera centers. Checking if there exists a solution that fulfills all constraints for a fixed error tolerance ϵ is a linear programming feasibility problem. To find the minimal ϵ we can use bisection, but first we will consider how to find the rotations.

B. Searching Rotation Space

To find the rotations between the different views we perform a branch-and-bound search, similar to that in [5]. Any rotation can be written

$$\mathbf{R} = \mathbf{R}_x \mathbf{R}_y \mathbf{R}_z. \quad (2)$$

where $\mathbf{R}$ is the total rotation and $\mathbf{R}_x$, $\mathbf{R}_y$ and $\mathbf{R}_z$ are the rotations around the x -, y - and z -axes respectively. We choose coordinate system, such that the z -axis is perpendicular to the ground plane. This means that vehicles rotate only around the z -axis and thus $\mathbf{R}_x$ and $\mathbf{R}_y$ are equal for all views. They specify the orientation of the ground plane relative to the camera. For now, let us assume that this has already been estimated. Thus we seek one rotation angle for each camera, apart from the first one, which we can choose to be zero. Each of these angle can be chosen between $-\pi$ and π . Thus our search space can be identified with the product space $[-\pi, \pi]^{(N-1)}$, where N is the number of cameras.

Starting out we have a list of just one block $[-\pi, \pi]^{(N-1)}$ as well as an initial error threshold ϵ . At each iteration, we pick a block from the list and try to determine if this block can contain any solution with reprojection errors $< \epsilon$. This is determined by solving an LP feasibility problem as described in the previous section but with error tolerance $\epsilon + \Delta$ instead of ϵ . The Δ is an extra uncertainty that accounts for the size of the block. Details can be found in [2].

If this test falls out positive, then the block is divided and the new blocks are added to the list. Otherwise it is simply deleted. This goes until the remaining set of rotations is small enough.

Branch-and-bound Algorithm

Iterate until desired precision is reached.

1. Pick the first block from the list.
2. Calculate the constraints and set up the LP problem.
3. Determine if there is a solution to the LP problem.
4. If there is a solution.
 - Divide the block into smaller blocks and add them to the list.
 - Try to update the error threshold by performing the bisection algorithm.
5. Remove the current block from the list.

C. Bisection

In Section III-A we showed how to check feasibility for a fixed error tolerance ϵ . To find the L_∞ optimal solution we also need a method to update this tolerance. This is done with a bisection algorithm. For blocks that pass the feasibility test we try to find a solution having a smaller reprojection error. We fix the rotations to the middle of the block and perform a feasibility test with error tolerance ϵ . If this passes we know that we have found a better solution. To know how good, we use bisection.

We start with the interval $[0, \epsilon]$. Let $\gamma = \epsilon/2$ and check feasibility with error tolerance γ . If this is feasible we know that the best reprojection error is somewhere between 0 and γ and we set the upper bound to γ . If there is not a solution, we set the lower bound to γ instead. The



Fig. 3. Rotation of the camera, the camera is first rotated around the z -axis, then around the x' -axis and at last around the z' -axis.

interval is now half the length of the original interval. Again we try to find a solution in the middle of the interval and change either the upper or the lower bound. This is repeated until the interval is as short as desired. Finally we update the error threshold to the upper bound.

D. Estimating the Ground Plane

If the camera rotation with respect to the ground is unknown, we have to calculate that as well. This makes the problem harder since we get two more dimensions to search but apart from that the same approach can be used.

The parameterization of the rotation is shown in Fig. 3. The camera is first rotated around the z -axis, then around the x' -axis and at last around the z' -axis and the total rotation can be written as

$$\mathbf{R} = \mathbf{R}_{z'} \mathbf{R}_{x'} \mathbf{R}_z,$$

where $\mathbf{R}$ is the total rotation and $\mathbf{R}_{z'}$, $\mathbf{R}_{x'}$ and $\mathbf{R}_z$ are the rotations around the z' -, x' - and z -axes respectively. The rotations around the z' - and x' -axes are unknown but equal for all cameras.

IV. EXPERIMENTAL RESULTS

A. Estimating the Ground Plane

To estimate the camera rotation with respect to the ground plane, three images of a bus driving through the intersection were used. Fig. 4 shows the images used, while the result of the reconstruction can be seen in Fig. 5. To find the position of the bus a map of the intersection was created by rectifying an image using estimated ground plane. The 3D representation of the bus was then projected into that map and the result is shown in Fig. 6. Blue dots correspond to the first time point, red to the second and green to the third.

By comparing distance in the rectified image and a real map of the intersection, we can estimate the size of the bus. The height was estimated to 2.65 m, the width to 2.15 m and the length to 12.7 m. The bus is 2.8 m high, 2.3 m wide and 12 m long, giving us an average error of 0.3 m.

B. Reconstruction with Known Ground Plane

The knowledge of the ground plane can now be used to reconstruct other vehicles. Three images used in reconstructing a car can be seen in Fig. 7.

The result of the reconstruction can be seen in Fig. 8 and the position in the intersection is shown in Fig. 9.



Fig. 4. Three images of a bus driving through the intersection. The yellow dots marks the image points which was used for the reconstruction and the yellow lines are only for visualization.

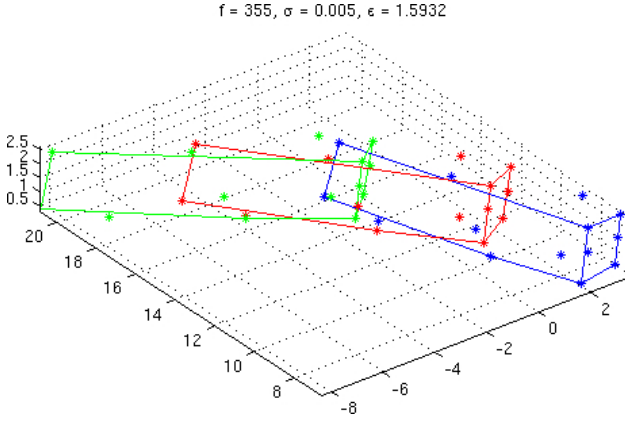


Fig. 5. Reconstruction of the bus driving through the intersection at three time points. The tracked points used for the reconstruction are marked as well as some lines used for visualization.

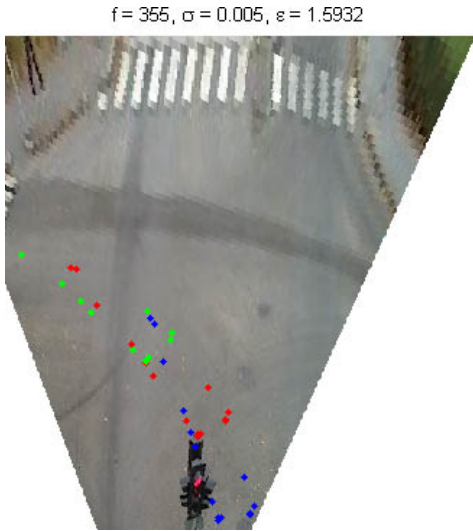


Fig. 6. Projections of the bus to a map of the intersection at three different time points.

V. CONCLUSIONS

We presented some ideas on how to achieve accurate 3D reconstructions for traffic surveillance. The approach builds on recent research in optimal methods for computer vision. This makes it possible to fully exploit restrictions such that the ground is planar. It also allows for easy setup since the camera position can be estimated automatically.

REFERENCES

- [1] A. Laureshyn and H. Ardö, "Automated video analysis as a tool for analysing road user behaviour," in *ITS World Congress*, London, UK, 2006.
- [2] H. Källén, "3d reconstruction for traffic surveillance," Master's thesis, Dept. of Mathematics, Lund University, Sweden, 2009.



Fig. 7. Three images of a car driving through the intersection. The red dots mas the image points which was used for the reconstruction, some lines used for visualization are also shown in the images.

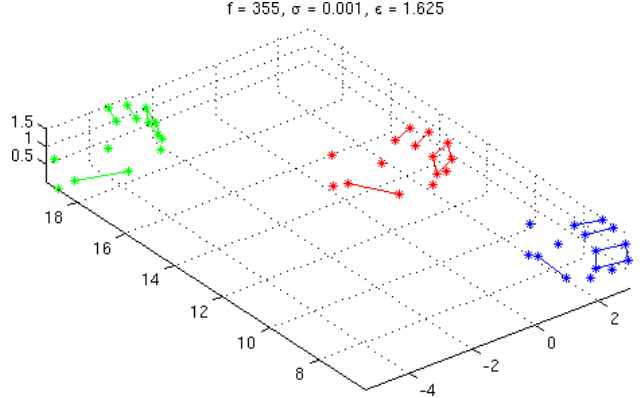


Fig. 8. The reconstruction of a car driving through the intersection at three different time points. The tracked points used for the reconstruction are marked as well as some lines that are only used for visualization.

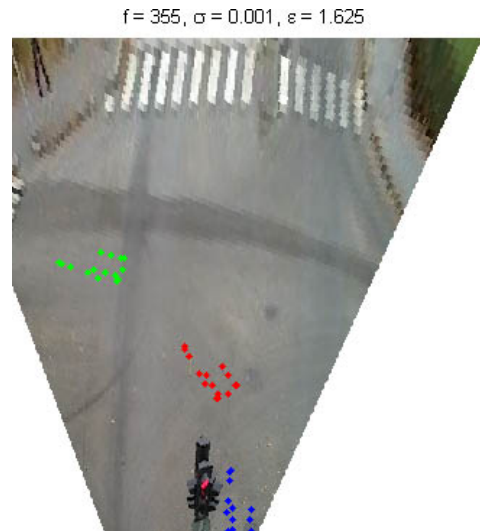


Fig. 9. Projections of the car to a map of the intersection at three different time points.

- [3] C. Tomasi and T. Kanade, "Detection and tracking of point features," Computer Science Department, Pittsburgh, PA, Tech. Rep. CMU-CS-91-132, April 1991.
- [4] F. Kahl and R. Hartley, "Multiple view geometry under the L_∞ -norm," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 30, no. 9, pp. 1603–1617, 2008.
- [5] R. Hartley and F. Kahl, "Global optimization through rotation space search," *Int. Journal Computer Vision*, 2009.