VEHICLE LOCALIZATION BY UTILIZATION OF MAP-BASED OUTLINE INFORMATION AND GRAYSCALE IMAGE FEATURE EXTRACTION

Kristin Schönherr, Björn Giesler
AUDI AG
85045 Ingolstadt, Germany
kristin.schoenherr@audi.de, bjoern.giesler@audi.de

Alois Knoll
Institute of Computer Science VI
University of Technology Munich
85748 Garching, Germany
knoll@in.tum.de

ABSTRACT

Todays vehicles are more and more equipped with video cameras. These cameras are used e.g. for lane and parking assist systems. For the localization of the vehicle in the world the Global Positioning System is widely used. Unfortunately, GPS is very imprecise and insufficient for many driver assistance applications. To overcome this limitation, a more precise localization using a different approach has to be found.

High quality localization sensors are already available on the market but still too expensive for mass integration in automobiles. The aim is to extract additional information from already established car equipment and to use it for precise localization. Our approach combines map information with extracted image features using a model based algorithm.

KEY WORDS
image processing, pose estimation, object tracking

1 Introduction

Automotive sensors today are mostly used for single purpose driver assistant systems. In the future all the available sensors shall be connected for the realization of complex environmental analyses. Especially the camera and the available map information of navigation systems will be selected to take over the functional role for advanced information delivery. The combination of the sensor data aims to achieve a high precise self-localization of the vehicle. The GPS sensor is delivering a rough absolute position which is enriched by relative object information generated by model based pose estimation.

Precise localization information is necessary especially in environments with a high traffic density like urban areas. Obstacles, shade hue and multipath signal distribution are the main reasons for bad GPS localization quality in these areas.

Map matching algorithms in navigation systems already provide good localization in lateral direction, but the calculation is based on a vector oriented environmental geometry where multi lane roads are digitalized as one single vector. Therefore the lateral precision is limited and the longitudinal direction is not considered at all. However, the map content is being continuously improved in terms of detail, precision and information enrichment like e.g. model description and outline information of buildings. We exploit this additional information to build a model to match against input from car-mounted sensors, to aid in both longitudinal and lateral localization.

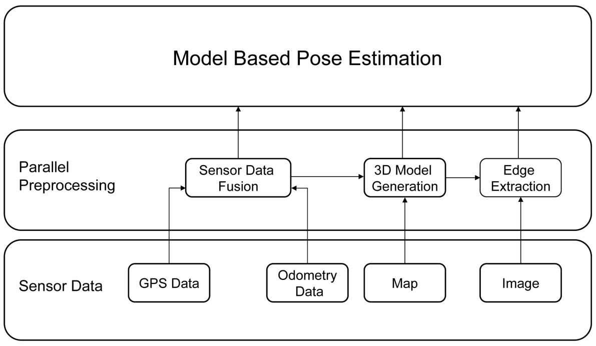


Figure 1. Flow chart of process units

In this paper, we use a monoscopic video camera to align the map based information with the vehicle environment. Additionally the movement of the car can be estimated by evaluating the odometry data of the vehicle like speed and turn rate, in a traditional dead-reckoning approach. Hence, to obtain globally precise position information for the car, we use detailed maps (GIS[1]), GPS sensor, a camera and odometry data, see Fig. 1.

The introduced method is derived from a line style approach using landmarks [1]. The main disadvantage of the line style approach is its reliance on clear visibility of the line landmarks, which cannot be guaranteed in real-world scenarios (if e.g. ground based lines like lane border marks are used as features, they are often occluded by parked cars and other obstacles). Therefore a major difference of the presented method is to utilize 3D models of buildings gen-

[1]GIS-Geographic Information System

erated from outline data, and match them with borders of reference objects in the real world.

2 Map Based 3D Model Creation

The map material we use contains several categories of object information. Not only streets but also outlines of buildings are provided. The latter are composed of closed polygon formations. Basis for the discussed method are 3D lattice models which are computed in several steps from the raw map material with the differentiation between visible and non visible edges.

2.1 Visible 3D Lattice Model

The initial step is the assumption and simplification that the outline polygon is a projection of the building to the ground plane, and that the nodes of the outline polygon represent more or less sharp creases in the surface of the building which can be detected as edges in an image of the building. Since height information is not available, an initial 3D model is generated by extruding the ground plane vertically to a certain assumed building height (currently assumed to be 10m), see Fig. 2.

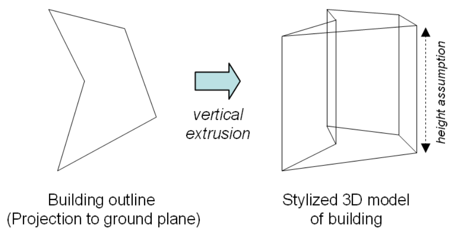

vertical extrusion
height assumption
Building outline (Projection to ground plane)
Stylized 3D model of building

Figure 2. 3D model creation from outline

The resulting pseudo-3D model is a wireframe model that contains lines that would normally be invisible due to occlusion. Since trying to match those lines with image edges would be a source of error, we eliminate them using computer graphics techniques.

2.2 Backface Culling

Backface culling is a known operation of 3D computer graphics computation [3]. The vector in the viewing direction is compared to the normal vector of the surface. Only surfaces are plotted whose normal vector direction is opposite to the direction of the viewing vector.

The backface culling is adpated to the 2D outline plane, whose polygon formations are always defined in a clockwise direction. With the knowledge of direction the normal vector of every polygon line can be computed and compared with the viewing direction. All polygon lines will be discarded whose normal vector is in viewing direction, see Fig. 3.

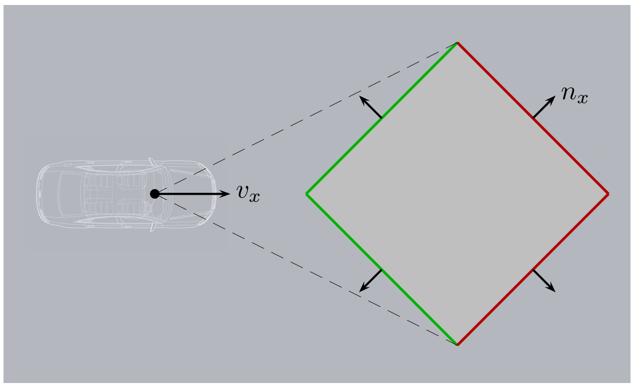


Figure 3. Backface culling

The visibility of the outline is determined using the following scalar product:

$$\begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} \cdot \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \qquad \begin{cases} > 0: & \text{unvisible} \\ \leq 0: & \text{visible} \end{cases} \tag{1}$$

n_i = normal vector of polygon line;
v_i = view direction

2D backface culling discards most of the non-visible lines but inter-object occlusions, as well as concavities in object outlines, are not handled by backface culling.

2.3 Ground Plane Based Angle Separation

To solve the occlusion problem the outlines based on the cameras viewing direction are dynamically categorized. Starting with the nearest visible element of the outlines an angle map is created that acts as an accumulator for occlusion information. Elements in the front will always be plotted; the angular range they cover is retained in the angle map. Elements behind are compared against the already known angle ranges of the front elements. Known ranges will not be plotted again. Only the remaining part of the outline elements in the back will be displayed by cutting the line.

The Fig. 4 shows the three cases of the angle separation which are possible, where the green part of line a is plotted and the red one is not visualized.

- The case number one shows the complete occlusion of line b.
- The case number two shows no occlusion, where the angle range of line a is already determined and is not influencing the visualization of line b.
- The case number three shows partly occlusion, where a segment of line b is overlapping with line a in the front and their respective angle range.

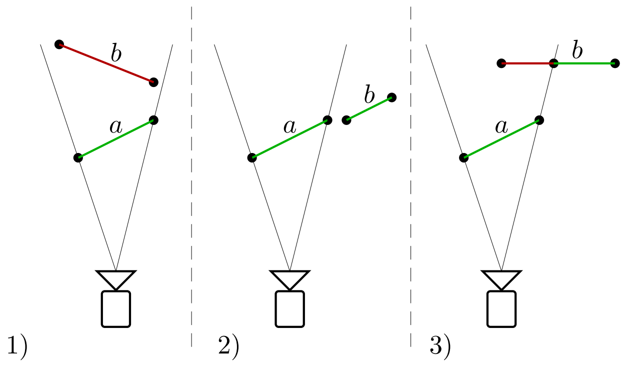


Figure 4. Different cases of angle separation

In view of the already discribed different cases the algorithm of angle separation is applied to all frontfacing outlines. The Fig. 5 shows the result of the angle separation treatment.

With this computation all the visible elements are known and the creation of the 3D building model is completed by the assumed height. But in addition a modification of the real camera image is necessary to prepare the whole system for the following video-model-combination.

3 Image Processing

The video image is treated with edge extracting operations to isolate them from the picture.

We are using a combination of Canny operator [10] and subsequent Hough transformation [11] to extract long edges from the image assuming that these edges are the most probable to represent building walls. However the discarded short edges mostly represent unmapped environment features (parking cars etc.) or relatively small features on the buildings themselves which are also not contained in the model. One important discriminating feature for building edges are their edge direction within the image, which should be approximately the same like those of the projected model edges. Therefore, we examine the image gradient direction obtained by Sobel-filtering in image x and y direction, see Fig. 6. Once the pretreatment of the map material and camera image is done the combination of both follows.

4 Model and Image Combination

The GPS sensor yields a rough position estimate for the car in the world coordinate system. To a certain degree, the direction of the vehicle can be extracted from the GPS data too (by filtering noisy position data over time and extracting the direction of the motion vector). Given position and direction, relevant parts of the map material similar to a limited sensor view can be extracted. Both the 3D model based on the map material and the camera are referenced to

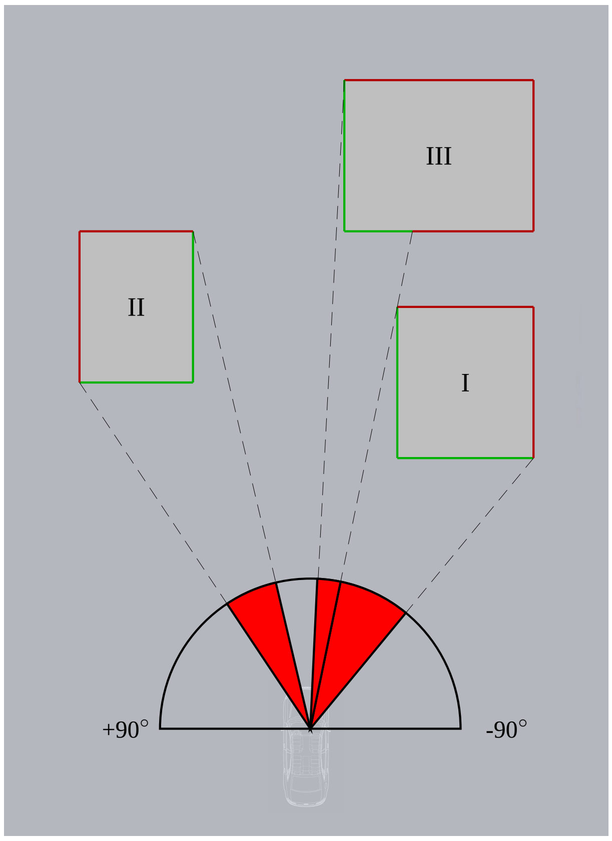


Figure 5. Angle separation

the vehicle coordinate system, making the 3D model transformable to the camera coordinate system to create a model overlay over the camera image.

The alignment and convergency between the overlayed model and real buildings in the camera image (compare [4], [9]) generates a relative position error estimate, which can be used to improve the absolute localization. The Fig. 7 shows the projected model edges of the ground truth position. Futhermore the difficult regions of a outdoor scene are highlighted, which have to be handled by the object tracking algorithm. Interfering image edges, occlusion by obstacles and the assumed building height are sources of error, which influence the quality of pose estimation.

4.1 Object Tracking

The RAPID[1] tracker by Harris [5] is a realtime model based algorithm which combines 3D model information with image data for pose estimation. In the process 3D model edges are projected on the image based on an initial pose hypothesis, where they are compared with image edges. The steps described above are in preparation of the use of the RAPID algorithm on outdoor scenes. The analysis should show whether the model based pose estimation han-

[1]Real-time Attitude and Position Determination

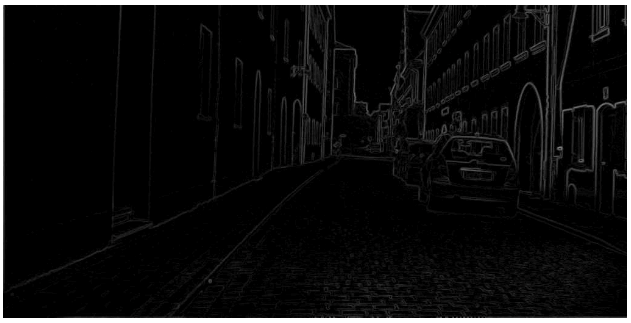


Figure 6. Image of Sobel amplitude

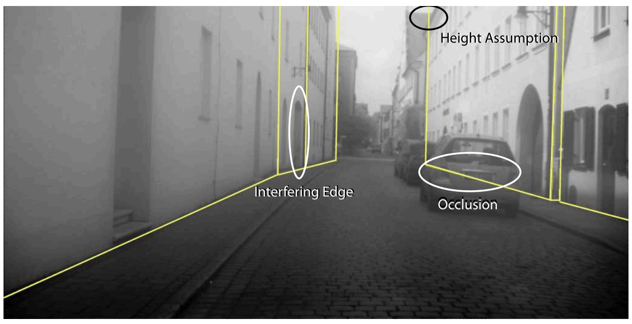


Figure 7. Image with overlayed visible model edges using ground truth position and with idendified sources of errors for pose estimation on outdoor scenes

dles the different sources of irritation and delivers a precise result.

The aim is to estimate the exact pose of the object (here, the entire world model) compared to the camera, see also [8]. In every frame, the model edges of the 3D object are projected into the image and are being compared to the extracted edges of the video image. Starting from projected model edges, the algorithm searches the image for appropriate edge pixels in orthogonal direction [6]. This allows to determine the deviation between the model and object edges. The RAPID algorithm estimates the pose based on the object coordinate system where the root is given at $T = (T_x, T_y, T_z)^T$ in the camera coordinates [7]. Additionally the axes of the object coordinate systems have to be aligned to the camera coordinate system. The Fig. 8 shows the relation between object and camera coordinate system.

A control point located on a model edge $P = (P_x, P_y, P_z)^T$ given in object coordinates can be described in camera coordinates by

$$X = T + P = (X, Y, Z)^T. \tag{2}$$

After a movement δp, where the object is rotating at the object root T by $\delta\omega$ and is displaced by δt, the control points are located at

$$X' = T + \delta t + P + \delta\omega \times P. \tag{3}$$

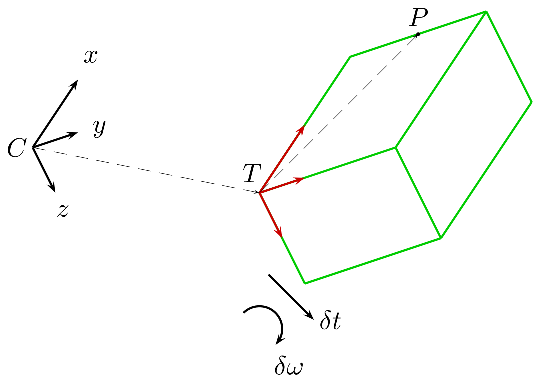


Figure 8. Object coordinate system in relation to the camera coordinate system C

To simplify the projection of the points X' into the image the focal length is assumed to be 1 and the lens distortion is completely ignored. Additionally the center of the image defines the root of the image coordinate system. A point in pixel coordinates is finally given by

$$x' = \left(\frac{X'}{Z'}, \frac{Y'}{Z'}\right)^T. \tag{4}$$

Afterwards the projection term x' of X' can be extended by $\delta\omega_x, \delta\omega_y, \delta\omega_z$ and $\delta t_x, \delta t_y, \delta t_z$.

$$x' = x + \frac{\delta t_x + \delta\omega_y P_z - \delta\omega_z P_y - x(\delta t_z + \delta\omega_x P_y - \delta\omega_y P_x)}{T_z + P_z}, \tag{5}$$

$$y' = y + \frac{\delta t_y + \delta\omega_z P_x - \delta\omega_x P_z - y(\delta t_z + \delta\omega_x P_y - \delta\omega_y P_x)}{T_z + P_z}. \tag{6}$$

Transformed to matrix notation the term can be given by:

$$x' = x + W\delta p, \tag{7}$$

with $\delta p = (\delta\omega_x, \delta\omega_y, \delta\omega_z, \delta t_x, \delta t_y, \delta t_z)^T$, a vector composed of rotational and translational coefficients and W, of a 2×6 Matrix, which describes a function of the coordinates T and P. The distance l from figure 9 is consequentialed by

$$l = \vec{n} \cdot (x' - x). \tag{8}$$

Based on the equations 7 and 8 the following equation

$$\vec{n}_i^T \; W_i \; \delta p = l_i. \tag{9}$$

can be derived.

With a sufficient amount of control points available the pose δp can be derived by minimizing the vertical lengths l where the the least squares method is applied.

$$\delta p = arg\ min \sum_i \left[\vec{n}_i^T \; W_i \; \delta p - l_i\right]^2 \tag{10}$$

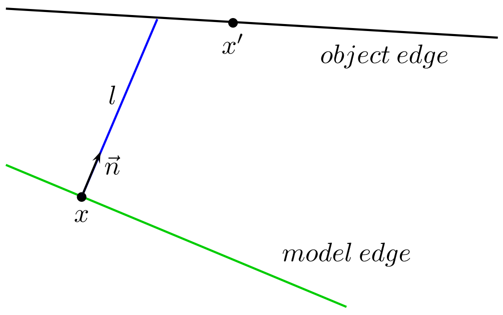


Figure 9. Starting from the projected model edge, the RAPID algorithm searches of the object edge along the normal within the video image and calculates the located distance l.

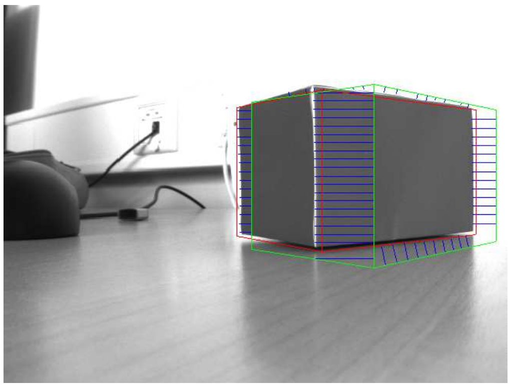


Figure 10. Correctly executed RAPID algorithm (red lines)

With the summarization of $\vec{n}_i^T$ and W_i to the matrix A the equation

$$l = A\,\delta p. \tag{11}$$

can be derived. The solution of δp can be found by the usage of the pseude inverse of A.

$$\delta p = (A^T A)^{-1} A^T l \tag{12}$$

The result δp of the RAPID algorithm displays the difference between the projected model and object edges. Based on the utilization of 3D information of the map material δp describes a 3D pose consisting translation and rotation. To improve the inaccurate position of the vehicle, the erroneous initial position estimate is transformed iteratively with the translation and rotation yielded by RAPID. A visual checking of the tracking method is the transformation of the map material by δp. The new edge model is displayed in red in the following images. If the RAPID algorithm is executed correctly, the red edge model is congruent to the building object in the video image.

The Fig. 10 shows a correctly executed model matching. The visualized model in green is perfectly fitted to object in the video image. The model transformed by δp will be displayed in red and is congruent to the captured object. The initial inaccuracy of the vehicle position is improved by using the RAPID algorithm.

5 Practical Results

The functionality of RAPID solving the challenging task of self-localization with complex environment images is verified by using two different successive test series.

- *Internal scene:* As a simplified test, we apply the algorithm to a simple cuboid model and analyze the performance.
- *External scene:* Afterwards, we examine the recorded sensor data of a real vehicle movement in an urban area. This provides much more complex image data. The position ground truth is elaboratly determined by using reference sensors such as differential GPS. This precise position information is purposely distorted by an adaptive normal distributed error.

In both cases the result of the RAPID algorithm is not adopted directly. Instead the algorithm is modified using a step by step approach so the movement of the model can be observed and plotted, see Algorithm 1.

```
Input: initial pose of object and image
Output: estimated precise pose of object
Calculate edge map from the image
while $\sum l_i < epsilon$ do
    foreach model line do
        • Project every model line point and
          determine the normal at current pose
        • Match every point with the nearest edge point
          and get distances $l_i$
        foreach edge point do
            Compare model and object edge
            direction
        end
        • Determine outliers by RANSAC
    end
    • Minimize distances by solving
      $\delta p = arg\ min \sum_i \left[\vec{n}_i^T\ W_i\ \delta p - l_i\right]^2$
    • Use certain percentage of $\delta p$
end
```

Algorithm 1: Pose estimation by stepwise RAPID

5.1 Internal Scene

To determine the accuracy of an image based method to estimate the position, the RAPID algorithm is evaluated using a model cuboid. The Fig. 11 shows the iterative approximation of the model towards the object, which is expressed through the error reduction. Especially the x and y position and yaw angle are being considered because they represent the main characteristics of the positioning evaluation, in particular the position and the direction of the vehicle. The absolute errors of the remaining evaluational parameters are shown in Table d. It can be observed that the accuracy of the translation is in the range of millimeters under optimal conditions. Also the angles show a small deviation in the range of tenths to hundredths degrees.

5.2 External Scene

The second test evaluates the RAPID algorithm against real-world test data acquired with a moving car in a city context. The ground truth position and direction of the car in the scene are determined using reference sensors[2]. The position value in x direction is manipulated by a deviation of 2m evaluating the perfomance of the RAPID algorithm. The Fig. 12 shows the results of the iterative RAPID estimation in a complex image context. It can be recognized that the x value is stepwise improved by the RAPID algorithm. For observation the translation value in y direction and the yaw angle are also plotted over all RAPID steps. Both values vary in a very small range so the correction relates basically to the x value as expected. The used scenery is already challenging for the tracking algorithm because building outlines are occluded by vehicles and the upper building edges can not be considered since the heights of the buildings are unknown. But less edge information are sufficient for pose estimation by stepwise RAPID algorithm. The sum of all distance values can be used as the stop criterion of the algorithm which converges into a local minimum. Results of ongoing iteration steps show that the algorithm stays within that local mimimum which proofs the robustness. In contrast the non-iterative RAPID can not obtain this precise result.

6 Conclusion and further work

The proposed approach enables to adapt known methods from robotics science to vehicles. Further the presented method enables the utilization of already established vehicle sensors for additional sensoring tasks. It is shown that vehicle self-localization in urban areas is possible with only a monoscopic video camera. Since the method relies on precreated map material, high quality results demanding highly precise maps.

The proposed solution offers a memory efficient data base application where only building outlines are stored. Any additional content necessary for the computation is generated in real time. Especially the calculation of visible elements based on 2D information fastens up the process compared to classic 3D operation procedures like e.g. z-Buffer algorithm [2].

Further work will try to make position estimation even more accurate and more robust by incorporating Kalman and Particle filtering and will examine the possibility of extending / correcting incomplete map information by doing our own mapping.

[2]We use differential GPS and a high-precision inertial unit to obtain ground truth in centimeter and tenth-of-degree accuracy.

References

[1] U. Scheunert, H. Kramer, G. Wanielik: *Precise Vehicle Localization Using Multiple Sensors and Natural Landmarks,* Proc. of the 7th International Conference on Information Fusion, Chemnitz, University of Technology, 2004

[2] E. Catmull: *Subdivision Algorithm for Computer Display of Curved Surfaces,* PhD Thesis, Dept of Computer Science University of Utah, Salt Lake City, Utah, U.S.A., 1974

[3] A. Nischwitz, M. Fischer, P. Haberäcker: *Computergrafik und Bildverarbeitung,* Friedr. Vieweg u. Sohn Verlag, GWV Fachbuchverlag, Wiesbaden, 2007, pp. 67-68

[4] D.G. Lowe: *Fitting Parameterized Three-Dimensional Models to Images,* IEEE Trans. on Pattern Analysis and Machine Intelligence, 1991, pp. 441-450

[5] C. Harris, C. Stennet: *RAPID - A Video Rate Object Tracker,* Proceedings of the British Machine Vision Conference, 1990, pp. 73-77

[6] M. Amstrong, A. Zissermann: *Robust object tracking,* Proceedings of the Asian Conference on Computer Vision, 1995, pp. 58-62

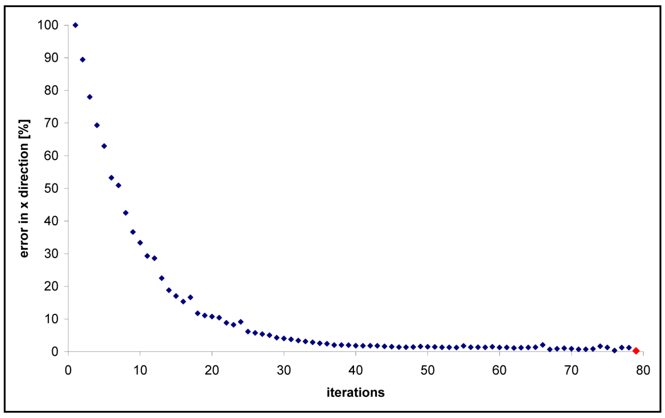
[7] V. Lepetit, P. Fua: *Monocular Model-Based 3D Tracking of Rigid Objects: A Survey,* Foundations and Trends in Computer Graphics and Vision, 2005

[8] A. P. Gee, W. Mayol-Cuevas: *Real-Time Model-based SLAM Using Line Segments,* 2nd International Symposium on Visual Computing, 2006

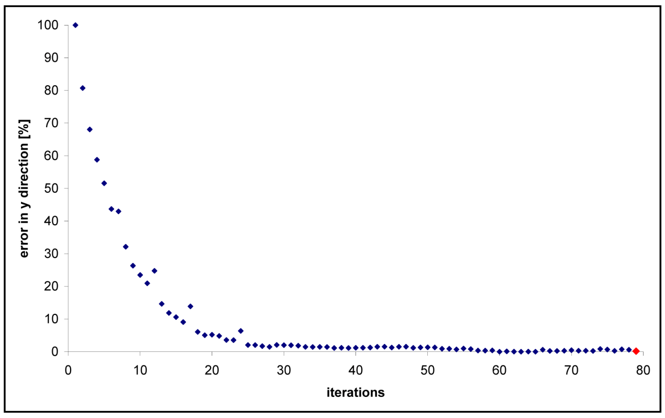
[9] L. Vacchetti, V. Lepetit, P. Fua: *Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking,* International Symposium on Mixed and Augmented Reality, Arlington, 2004

[10] J. Canny: *A computational approach to edge detection,* IEEE Trans. on Pattern Analysis and Machine Intelligence, 1986, pp. 679-698

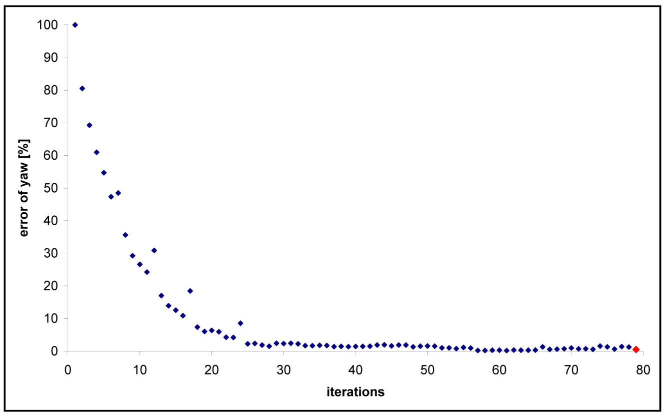
[11] R. C. Gonzales, R. E. Woods: *Digital Image Processing,* Prentince Hall, New Jersey, 2002, pp. 587ff.



(a)



(b)

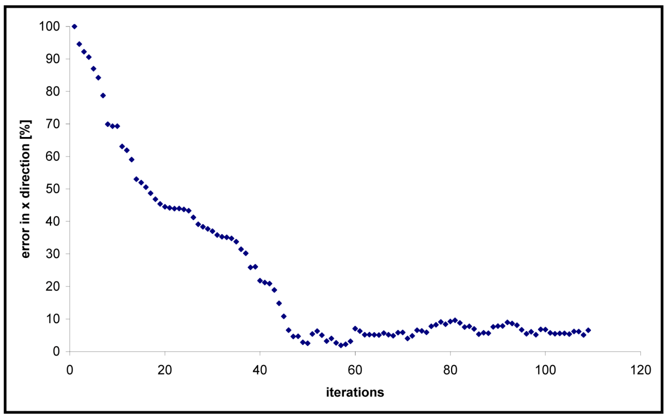


(c)

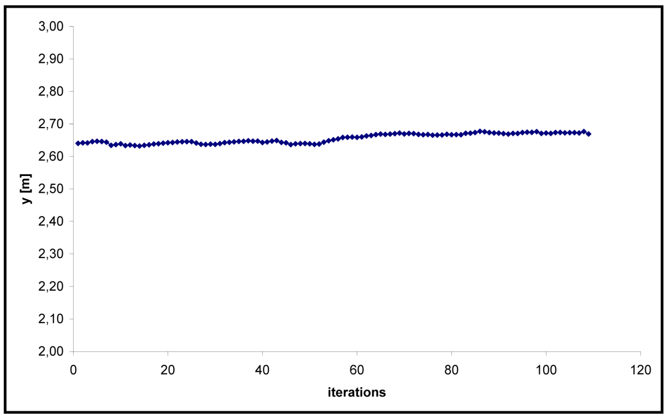
	Tx [m]	Ty [m]	Tz [m]
absolute error of best fit	0.001318	0.001176	0.000480
	Roll [°]	Pitch [°]	Yaw [°]
absolute error of best fit	0.083021	0.020454	0.238866

(d)

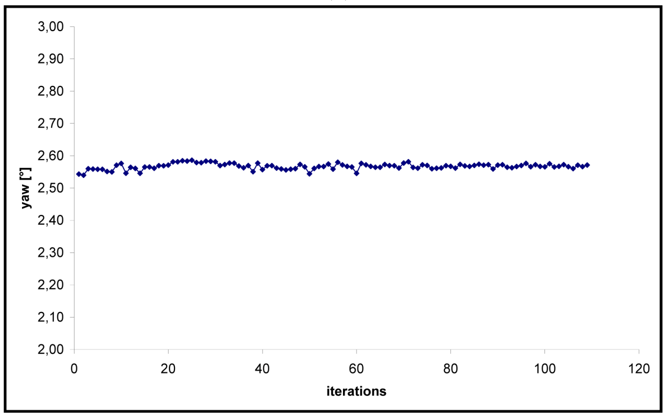
Figure 11. Results of RAPID algorithm using internal scene using an initial 4m deviation expressed by x and y value of the position and yaw angle. The diagrams show the normalized error where the best fit is displayed in red. a) Normalized error in x direction, b) Normalized error in y direction, c) Normalized error of yaw angle, d) Absolute error of best fit of all estimated RAPID algorithm parameters



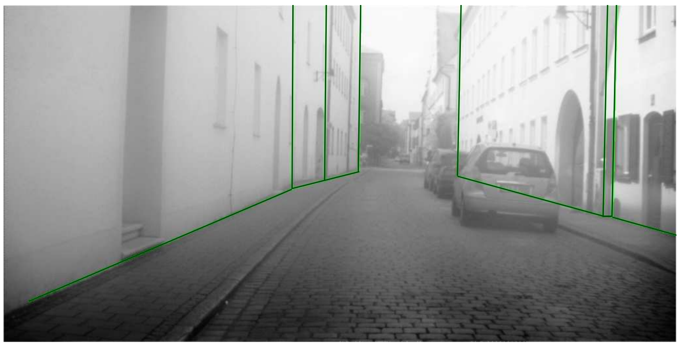
(a)



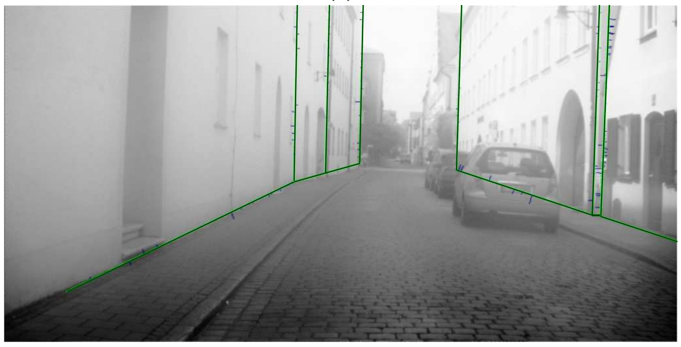
(b)



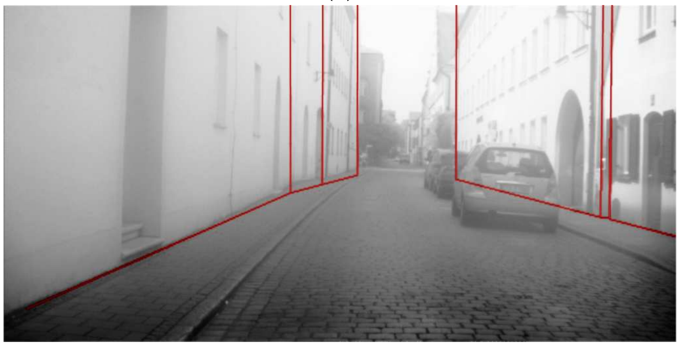
(c)



(d)



(e)



(f)

Figure 12. Results of the iterative RAPID algorithm using external scene using an initial 2m deviation in x direction. The diagrams show the results of the RAPID steps $a)$ Normalized error in x direction, $b)$ Observation of translation value in y direction, $c)$ Observation of yaw angle, $d)$ Image with inital model edges (2m deviation), $e)$ Image with final estimated model edges and distance vectors, $f)$ Compare: Image with final estimated model edges of non-iterative RAPID