A fast Visual Line Segment Tracker

Peer Neubert[*] and Peter Protzel
Chemnitz University of Technology
Straße der Nationen 62
Chemnitz, Germany
peer.neubert@informatik.tu-chemnitz.de
peter.protzel@etit.tu-chemnitz.de

Teresa Vidal-Calleja and Simon Lacroix
LAAS-CNRS
7 Avenue du Colonel Roche
Toulouse, France
Teresa.Vidal@laas.fr
Simon.Lacroix@laas.fr

Abstract

We present a fast line segment tracker which does not require any knowledge about the motion of the camera nor the structure of the observed scene. It runs on 320 x 240 pixel images at 30 Hz. We adapted the RAPiD tracker with a new way of handling multiple line hypotheses to deal with the simple model of a single line segment. We discuss the difficulty of using a χ^2-test as merging criterion and also present a new approach to overcome it. Furthermore, instead of making assumptions about the camera motion, a constant velocity motion model to predict the line segment position in the following frame is used. We explain how to deal with the instability of the endpoint extraction in this motion model to avoid unintentional motion along the line. Finally, we present results on real world indoor and urban outdoor image sequences.

1 INTRODUCTION

Whoever had a look at floor plans or constructional drawings probably knows that line segments are a very expressive feature to represent the structure of man made environments or objects. In order to use their expressive power in the field of Computer Vision or Robotics, we need robust algorithms to extract, track and match these features. Various algorithms exist for the extraction of a set of line segments (further referred to as *global extraction*, contrary to the recovery of a line segment at an assumed position in the image, called *local extraction*). A common way to obtain line segments is to link gradient maxima points (*e.g.* as in [3]) in order to assume a contour and apply one of the methods summarised in [10]. However, the achieved line segments are not stable and a small motion of the camera or in the scene can result in a very different set of line segments. Thus these algorithms are not sufficient for tracking. Another approach extracts a set of assumed corner points of the image and tests hypothetical lines between these corners [11]. But this yields in a limitation to a certain kind of line segments.

Nevertheless, in the beginning of the 90s a seminal contribution has appeared in line segment tracking [4], relying on a global extraction. Meanwhile, the RAPiD tracker [7], became the corner stone for a class of trackers for more complex three dimensional models using a local search (see [9] for an overview). There have been approaches to apply the idea of the RAPiD tracker on line segment tracking [11], but without special handling of the problems caused by the very simple model of a single line segments.

In this paper, we present a new tracking scheme that uses the basic idea of the RAPiD tracker, but deals with the problem of the simple model by generating and testing multiple line segment hypotheses in an efficient and robust way.

Line segments are often broken into several parts or vary in length in consecutive images, as can be seen in figure 1. The main problem when dealing with line segments is the instability of the endpoint extraction along the line. Thus we show how to take care of that in each tracking step. In particular, we show how to apply a robust χ^2-test based merging criterion and how to use a motion model for line segments avoiding the spurious motion along the underlying line.

The reminder of the paper is organised as follows. In section 2 the line segment representation used is presented. Our algorithm is composed by three main parts; extraction, merging and motion model. The line segment extraction is presented in section 3. The merging and motion model are presented in section 4 with the tracking scheme. Section 5 presents experimental results. Finally, conclusions and future work are presented in section 6.

2 Representation of an image line feature

We represent the image line features with endpoint coordinates and direct the line by assigning the brighter side in the image as the left side of the line. As we mentioned before, the extraction of the endpoints of a line segment along the line is not stable. Nevertheless, the endpoint

[*]Peer Neubert was partially with LAAS-CNRS

coordinates are convenient to represent the segment uncertainties.

Following the scheme presented in [4], we assign uncertainty perpendicular and parallel to the line and calculate the resulting covariance matrix Σ_{EP} of endpoint coordinates $\begin{bmatrix} x_1 & y_2 & x_2 & y_2 \end{bmatrix}$ as follows (α being the angle of the line to positive x-axis):

$$\sigma_x^2 = \sigma_{\parallel}^2 \cdot \cos\alpha^2 + \sigma_{\perp}^2 \cdot \sin\alpha^2 \tag{1}$$

$$\sigma_y^2 = \sigma_{\perp}^2 \cdot \cos\alpha^2 + \sigma_{\parallel}^2 \cdot \sin\alpha^2 \tag{2}$$

$$\sigma_{xy} = \left(\sigma_{\parallel}^2 - \sigma_{\perp}^2\right) \cdot \cos\alpha \cdot \sin\alpha \tag{3}$$

$$\Sigma_{EP} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & 0 & 0 \\ \sigma_{xy} & \sigma_y^2 & 0 & 0 \\ 0 & 0 & \sigma_x^2 & \sigma_{xy} \\ 0 & 0 & \sigma_{xy} & \sigma_y^2 \end{pmatrix} \tag{4}$$

where σ_x^2, σ_y^2 are the variances of the endpoint coordinates along the x and y-axis respectively and σ_{xy} the cross-correlation term.

For the merging (section 4.2) we consider only an uncertainty $\sigma_{\perp}^2$ perpendicular to the line but take the uncertainty of the direction into account (with l as length of the line segment)

$$\sigma_\alpha^2 = \frac{2 \cdot \sigma_{\perp}^2}{l} \tag{5}$$

$$\sigma_{x1,\alpha} = \frac{\sigma_x^2 - \sigma_{xy}}{l^2} \tag{6}$$

$$\sigma_{y1,\alpha} = \frac{\sigma_{xy} - \sigma_y^2}{l^2} \tag{7}$$

$$\sigma_{x2,\alpha} = -\sigma_{x1,\alpha} \tag{8}$$

$$\sigma_{y2,\alpha} = -\sigma_{y1,\alpha} \tag{9}$$

3 Extraction

We use two different approaches to extract line segments. If no prior information about the location of line segments is known we apply a global line segment extraction on the whole image. If there is some prior information, *e.g.* a predicted line segment from a prior image or an external 3D model, we process a local search in a small search range around the assumed position. Figures 2 and 3 illustrate the local extraction procedure.

3.1 Global Extraction

The global extraction uses the Canny edge detector [3] to find pixels contained in the contours of objects in the image. After a contour linking step, the Douglas Peucker algorithm [5] is used for a polygonal approximation of each contour. All resulting edges of the polygon with a minimum length are taken as line segment features.

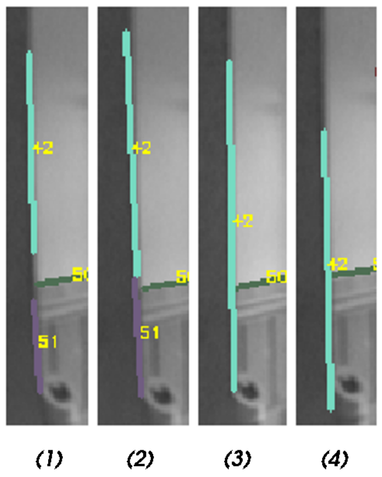

Figure 1. Instability of endpoint extraction. The line segment extraction from four slightly different points of view can be seen. (1) Two collinear line segments (42, 51) are extracted. (2) Both vary in length and shift along the underlying infinite line. (3) They are merged to one line. (4) The resulting line vary in length and position along the underlying line. It can shrink or even be split into several line segments.

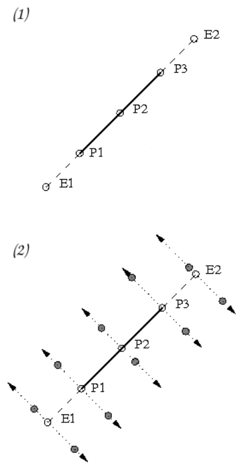

Figure 2. Local search illustration part I. (1) The line with control points P1, P2, P3 and extended control points E1, E2. (2) The gradient maxima points perpendicular to the line, one per side for each control point.

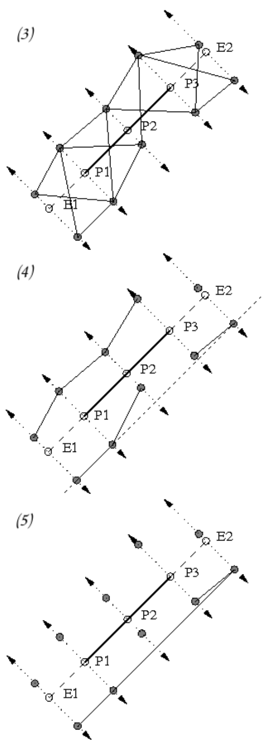

Figure 3. Local search illustration part II. (3) All line hypothesis. (4) Some hypotheses are rejected because of the angular difference. One gradient maxima point of E2 is associated to a hypothesis of E1, P1. (5) If these are the remaining hypothesis after the validation, they will probably be merged to the final line segment.

3.2 Local Extraction

The local extraction starts from a given assumption of the position of a line segment in the image.

We use an approach similar to the RAPiD tracker [7], a tracker for rigid 3D models. RAPiD assigns a set of control points to the contour of the objects according to the model that should be tracked. To recover the contour of the objects in successive images, the positions of the control points are predicted in the new image and a one dimensional search for gradient maxima perpendicular to the according contour is performed. From the new position of the control points the position of the object is recovered.

In our case the model is a simple line segment. Our approach uses the RAPiD idea and expands it to deal with the ambiguity raised by the simple model we want to track. The main extension are generation and handling of multiple hypotheses for a line.

The algorithm consists in the following steps:

1. Assign a small number of **control points** (*e.g.* 10) to the line and distribute them uniformly on the assumed line segment.

2. Search each control point perpendicular to the assumed line for **gradient maxima** with respect to the direction and orientation of the gradient. The search is performed independently for both sides of the line, and receives a set of points with gradient maxima, two for each control point (one for each side). The line segment is slightly extended in both directions and the search is also performed for these extended endpoints to cover motions of the line segment along the line.

3. Try to **recover a line** from the set of points. If there are several lines close to the line assumption or the area is well textured, the rate of points not belonging to the same line may be very high. Because of the small number of control points, a RANSAC approach can not be used. The reason is the line we want to find is maybe only indicated by a very small rate of gradient maxima points (*e.g.* 4 of 20 gradient maxima points). Thus, a set of hypothetical lines which defines all possible lines is created and a validation test applied.

   Define a **hypothetical line** by two gradient maxima points belonging to neighbouring control points. The total number of lines with n control points and the resulting $2n$ gradient maxima points is $4 \cdot (n-1)$. To decrease that number, all maxima gradient points with a small distance (*e.g.* < 0.25 pixel) to the hypothetical line, are assigned to that line. All further lines defined by two points that already belong both to another hypothesis are discarded. This constriction can be implemented efficiently by the use of bit-vectors, that handle the sets of gradient maxima points belonging to the hypothetical lines.

   The endpoints of a hypothetical line are defined by the outermost gradient maxima points assigned to that line.

4. The **validation test** first checks the angular difference to the assumed line position against a threshold. If the angle test is passed, the average values of the directed and oriented gradients of the pixels on the hypothetical line are checked against the gradient values of the maxima gradient points defining that line. If the difference is significant or the rate of pixel with considerable smaller gradient is high, the validation test fails.

5. **Merge** remaining collinear hypothetical lines. Because all hypothetical lines stem from the same old line, all lines that pass the χ^2-test described in section 4.2 disregarding the gap between the collinear line segments along the line are merged.

6. Keep the longest line. The resulting set of lines typically contains only one line or a small number of

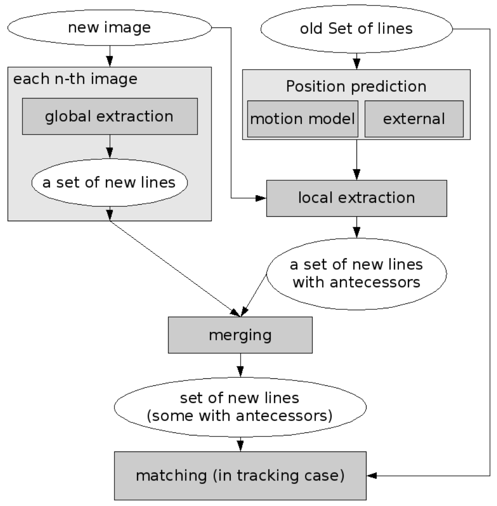

Figure 4. Tracking scheme. The input in each step are the current image and the set of line segments seen in the prior image. The local search is applied on each line of that set, for each n-th image (e.g. n=10) the global extraction is processed. The lines of both sets are merged. The resulting set contains line segments with antecessor and some without. The matching step assigns the former their antecessor as matching and the latter are treated as new features.

lines. All lines can be taken into account for further processing (maybe in a matching step), but our experiments showed that it is a good heuristic to use only the longest line.

The local search can track a line segment over a sequence of images and provides the matching information for that tracking process, but it will not find new line features. The global extraction takes more time and the line segments often disappear from one image to the other. Moreover there is no information about a matching provided by the global extraction because there is no correspondence to prior line segments as it is the case for the local search. Another problem is the dependency of the global extraction results on the parameters of the Canny edge detection. Using high thresholds we will loose lots of lines, with a smaller threshold we have a lot of noise in the contour image and the linking step will probably not find long straight lines. The local search adapts the gradient threshold to the gradients of the maximum gradient points in the search range. Therefore, we prefer a combination of both methods, *e.g.* run the global search each ten images and the local search in each iteration.

4 Tracking

Our line segment tracking algorithm can be used standalone to track lines over a sequence of images. It uses the global and local extraction of section 3. In the following we will first describe the scheme of our tracking approach and in section 4.2 the important step of merging line segments is discussed. To focus the search space for the local extraction we assign a motion model to each line which is described in section 4.3. Figure 4 gives an overview of the tracking algorithm.

4.1 Tracking scheme

Each iteration of the tracking handles the set of line features of the prior step and invokes a new image. The global line segment extraction can be processed in each iteration or, because it is computationally expensive, only each n-th image. By skipping the global extraction we risk to loose new line features that appear for the first time in the current image, thus the rate has to be chosen with respect to how rapidly the observed scene changes.

If there is a set of prior seen line features, we use a prediction of their position in the current image and apply the local search on each line segment. The prediction stems either from a kinematics model we assign to each line (described in 4.3) or from an external source (e.g. in SLAM application the projection of a 3D model of the line).

The two sets of current line segments, the one of the global extraction and the set of locally recovered lines, contain a high rate of similar lines that correspond to the same line in the environment. To fuse them we process the merging step described in section 4.2 on the union of both sets.

The final matching in the tracking scheme is based on the relation between new and old lines, thus a new line is matched to its antecessor from the local search. If there are several resulting line candidates in the local search we match only the longest line.

Finally, since to calculate the orientation of a line segment we use the average grey-level values on parallel lines with distance 2 pixel to the line, we store these values and use them as indicator for false matches in the tracking case. Thus, we do not match lines with very different average grey-levels on one or two sides.

4.2 Merging

We use the line merging step to merge collinear line segments both extracted in the global extraction, respectively both in the local extraction, and to fuse the line segment sets obtained by global and local extraction.

To decide if two lines A and B should be merged a χ^2-test is used. However, the extraction of endpoints along the line segment is not stable, thus we can not apply the test directly on them. Instead, we apply the merging test on the infinite lines containing the line segments. A common way to represent these infinite lines are polar coordinates (ρ, Θ) in the image frame. As mentioned above,

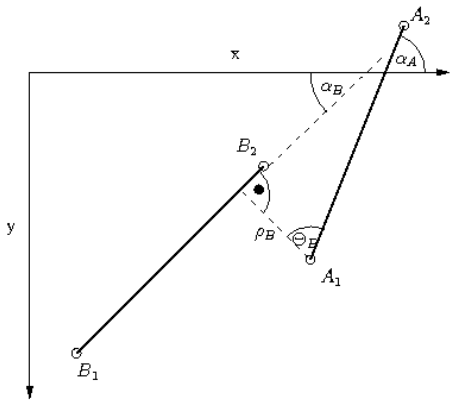

Figure 5. Local coordinate system for the merging. Line A defines the coordinate system, ρ_b, Θ_B are the polar coordinates of line B in the local system.

we compute the uncertainty of the endpoints in Cartesian coordinates during the line segment extraction process. It is possible to obtain the uncertainty in polar coordinates from this uncertainty. Unfortunately, the covariance matrix of a line represented in polar coordinates depend on its position in the image. Thus, the result of the merging test of two lines depends on their position in the image as well, even if their geometric relation is constant. To avoid that undesired effect, we apply the χ^2-test on polar coordinates in a local coordinate system with origin in one endpoint of line segment A of the tested lines (see Figure 5).

We are not interested in the local polar coordinates of the lines but in their difference. However to compute the Jacobian of that transformation we need some information about the calculation.

The local coordinates of line segment A which defines the system are directly given by

$$\rho_A = 0 \tag{10}$$

$$\Theta_A = \pm\frac{\pi}{2} \tag{11}$$

In polar coordinates, the representation of a line that goes through the origin is not one-to-one. That ambiguity is not needed to be solved, because the derivative of Θ_A with respect to the endpoints is always zero.

Then,

$$\rho_B = \frac{(x_{A1} - x_{B1}) \cdot (y_{B2} - y_{B1}) - (y_{A1} - y_{B1}) \cdot (x_{B2} - x_{B1})}{\sqrt{(x_{B2} - x_{B1})^2 + (y_{B2} - y_{B1})^2}} \tag{12}$$

$$\Theta_B = \pm\left(\frac{\pi}{2} + \Theta_B - \Theta_A\right) \pm \frac{\pi}{2} \tag{13}$$

where ρ_B is the distance of the line to the origin of the frame, which is an endpoint of line segment A. Θ_B can be expressed as the difference in the direction of both line segments and a multiple of $\frac{\pi}{2}$ which reflects the necessary rotation to get the polar coordinate Θ from the direction of a line (see Figure 5). Again its derivative is zero, thus we do not care about it. But the sign of the difference term in (13) influences the correlation term in the covariance matrix (not the variance terms, the value appears only squared).

Without solving that ambiguity we can calculate the difference in polar coordinates of the two lines as follows,

$$\Delta\rho = \rho_B \tag{14}$$

$$\Delta\Theta = \pm(\Theta_B - \Theta_A) \tag{15}$$

To avoid multiples of $\frac{\pi}{2}$ in (15) we can use the ambiguity in (11) to solve the one in (13). Nevertheless we stay with the uncertainty in the difference term.

Our state transition is

$$(x_{A1}\ \ y_{A1}\ \ x_{B1}\ \ y_{B1}\ \ x_{B2}\ \ y_{B2}\ \ \alpha_A\ \ \alpha_B)^T \rightarrow (\rho\ \ \Theta)^T \tag{16}$$

This state is over-determined, because the orientation is inherent in the endpoint coordinates. The covariance matrix for the origin state yields directly from (4-9). Fortunately, it is sparse because the values of one line are independent of the other line, which decreases the added computational effort. Further this state transition solves the ambiguity in (15), because the sign of the resulting correlation term changes with the sign in (15).

The main drawback of the χ^2-test in the local coordinate system is the need of the calculation of a new covariance matrix for each test. To avoid non useful tests, we first apply a one dimensional χ^2-test on the direction of the line segments only.

If both tests are passed we calculate the new line as weighted mean of both line segments. The used algorithm is described in [8].

4.3 Motion model

The local line feature extraction yields a need for a good assumption on the position of the line in the image. If there is an external 3D model of the line and information about the camera position, we can receive the prediction of the line by its projection in the image plane. Otherwise we need to assign a motion model to each line.

The simplest motion model would be constant position in the image. That means if a line has been seen in the last image, the same position in the current image is assumed. The larger the real motion between consecutive images is, the larger we have to chose the search range for the local search to cover the line. However, if the search range of the local search covers several lines or regions with considerable gradients, the local search can fail to recover the line. Thus it is useful to have a better prediction of the line motion.

We chose to use a constant velocity model similar to the one introduced in [4], where the motion of a line represented with midpoint coordinates, orientation and length is modelled with four independent Kalman filters, one

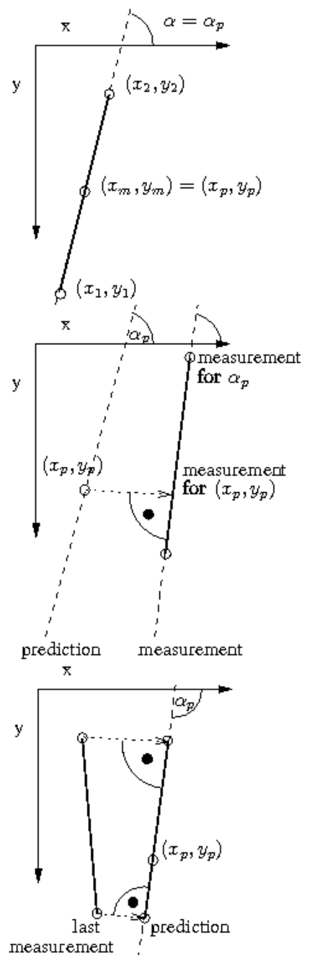

Figure 6. Motion model illustration. The figure shows the handling of the line segment representation in the motion model. Top: Prediction of the initialised infinite line. Center: Measurement of the new extracted line. Bottom: Update of the new line segment using the old line segment and the prediction for the underlying infinite line.

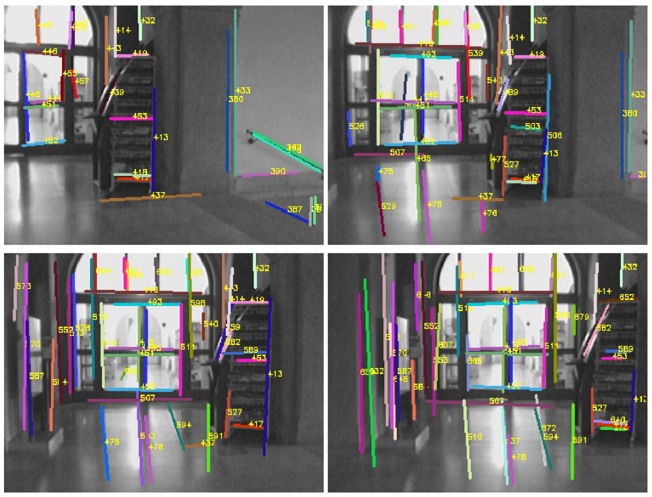

Figure 7. Indoor sequence shots. These four images show a challenging part of the indoor sequence with difficult illumination conditions, reflections and shadows. In clockwise order (starting from top left) they are taken with approximately 20 images between. Related lines have equal color and number.

for each element of the state vector. This representation needs a stable extraction of the endpoints, otherwise the motion model indicates an unintentional motion of the line segment along the underlying infinite line. Because the endpoint representation is not stable, the prediction is done over the underlying infinite line. We use the point-direction representation of a line, the estimated state being $\begin{pmatrix} x_p & y_p & \alpha_p \end{pmatrix}$.

Therefore we have three independent Kalman filters, one for each coordinates x_p, y_p of the point and one for the direction α_p, given by the angle to positive x-axis. The equations for updating the Kalman filters follow the common constant velocity model. For the initialisation, we use the coordinates of the midpoint and direction of the image line segment as is shown in Figure 6 (top). For updating with the new position measurement of that line, we use the orthogonal projection of the estimated point $\begin{pmatrix} x_p & y_p \end{pmatrix}$ on the measured lines, instead of the new position of the midpoint (see Figure 6 (middle)). The update measurement for the direction is directly the direction of the measured line. That means, we decompose the motion of the line in the image in a translation perpendicular to the line and a following rotation. Thus, the estimation is independent from the position of any point on the line.

5 Results

We tested the capability of the tracker with image sequences of indoor and urban outdoor scenes, captured from a hand held camera with smooth motion. We run the tracker online at 30 Hz with image size 320 x 240. All the computations have been done on an Intel Centrino with 1700 MHz (single core machine). The effort is focused on extracting and tracking as much useful line segments as possible. Useful means with a minimum length (set to 30 pixels in all experiments) and a repeated extraction from consecutive images.

Figure 7 shows sequence shots of the tracking in the indoor scene. The camera proceeds a 360 degree turn in about 30 seconds, 4 images with about 20 images between each other are shown. This scene is more challenging than a standard office environment, due to its complexity, illumination conditions, reflections and shadows. A lot of lines (those that are always visible) are tracked through the whole shown sequence (*e.g.* 413, 414, 417, 432, 445, 452, 453). However, some got lost and some appeared wrongly

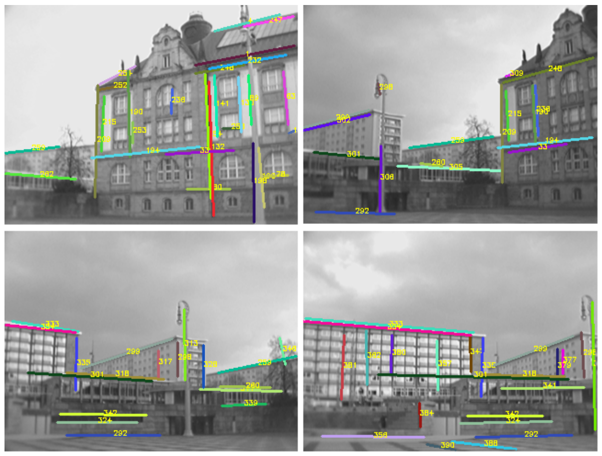

Figure 8. Outdoor sequence shots. These four images show a part of the urban outdoor sequence with a complex scene. In clockwise order (starting from top left) they are taken with approximately 30 images between. Related lines have equal color and number.

matched (*e.g.* line 510 jumps to a wrong collinear line between third and fourth shots).

The outdoor scene is even more complex. There are more line segments visible and due to the higher depth of the observed scene, they are sometimes very close to each other. Figure 8 shows some shots.

To evaluate the performance of the tracker, we measured the number of frames the line segments are tracked. The ground truth can be estimated by the number of frames for a full rotation and the viewing angle of the camera (the camera proceeds a 360 degree turn). In our case a line should be seen in about 100 frames. This number is a weak assumption because it is influenced by 2 main factors: first the time until the line is detected after it appears in the image and second the direction of the line. A vertical line appears and disappears completely between two consecutive images, while a long horizontal line can be seen much longer than any single of its points is visible (*e.g.* one can see the beginning of the line for about 100 images, but while the beginning of the line leaves the viewing field, another part of the line gets visible)

Figure 10 shows the rate of line segments that are tracked for a certain number of frames for the outdoor and for the indoor scene. The results for the indoor scene are better than for the outdoor one. This is due to the higher complexity of the outdoor scene with more line segments (about 3500 for one turn, compared to 1000 at the indoor scene) and the higher depth of the scene.

Indoors, the tracker is able to track more than 10 % of the line segments for more than 60 frames (results in more than 100 line segments). With the appropriate camera motion this should be sufficient for a line segment SLAM or structure from motion to perform fine. Outdoors, the rate is smaller (about 5 %) but results in more than 170 lines.

Table 1. Average case measurements of the indoor sequence

time per image (in *ms*)	9.15
time per line (in *ms*)	0.35
time per pixel (in μs)	4.77
number of lines per image	25.83

Table 2. Average case time for one tracking step

with global extraction (in ms)	18.1
without global extraction (in ms)	7.97
average number of lines	25.83

For a pure mapping, we can assume that a line is initialised in the map after 20 frames, this results in 230 lines (23 %) for the indoor scene and 510 lines (15 %) for the outdoor scene. Conspicuous is the rate of lines that are seen only one time, with about 50 %. Any further process, based on the tracker, should probably ignore these lines.

Table 2 shows the average duration of a tracking step with about 25 lines to track. It is necessary to invest some time for the global extraction to detect new line segments. The rate depends on how fast the observed scene changes. In our experiments the global extraction was applied on every 10th image frame. The expenditure of time for the first 300 frames of the indoor sequence can be seen in Figure 9. The peaks in time marks the tracking steps with global extraction. With each global extraction, the number of line segments increases suddenly at the beginning and decreases slowly afterwards. The decline is due to line segments leaving observed area or being no longer tracked.

6 CONCLUSIONS

We developed a fast line segment tracker that runs on images with size 320x240 at 30 Hz with about 25 line segments per image. We incorporated the idea of the RAPiD tracker and described a new way to handle the simple model of a single line in a robust way. We dealt with the difficulty of using a χ^2-test as merging criterion using a local reference frame to evaluate it. Further, we explained how to deal with the motion along the line due to the instability of the endpoint extraction in a motion model. Finally, we presented results on real world indoor and urban outdoor image sequences.

Even if the very most matches in tracking are right, only one false matching of a line during its observed time

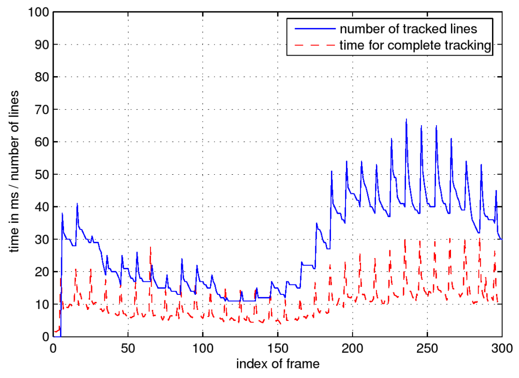

Figure 9. Time requirements. The red dotted line marks the duration for a complete tracking step per image and the blue solid line the number of tracked lines in the dedicated image frame. The values are taken from the first 300 images of the indoor sequence.

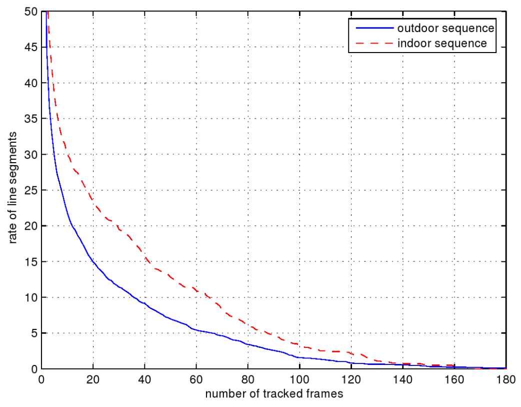

Figure 10. Tracking amount. The figure shows the rate of line segments that is tracked for at least a certain number of frames. The blue solid line marks the outdoor sequence and the red dotted line the indoor sequence

can disturb a process relying on the tracker. To enhance the robustness of the tracking, it is necessary to incorporate stronger appearance or structure based validation criteria, such as histograms or topological constraints. Preliminary results show that such criteria, carefully integrated, can be exploited with a minimal degradation of the computation time.

References

[1] K. Arras and R. Siegwart. Feature extraction and scene interpretation for map-based navigation and map building, 1997.

[2] Y. Bar-Shalom and X.-R. Li. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[3] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(6):679–698, Nov. 1986.

[4] R. Deriche and O. D. Faugeras. Tracking line segments. In *ECCV 90: Proceedings of the first european conference on Computer vision*, pages 259–268, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[5] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, Dec. 1973.

[6] A. P. Gee and W. Mayol-Cuevas. Real-time model-based slam using line segments. In *2nd International Symposium on Visual Computing*, November 2006.

[7] C. Harris. *Tracking with Rigid Objects*. MIT Press, 1992.

[8] A. P. J.M. TAVARES. A new approach for merging edge line segments. In *7 Congresso Portugułs de Reconhecimento de Padres*, Aveiro, 1995.

[9] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1–89, 2005.

[10] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis, and R. Siegwart. A comparison of line extraction algorithms using 2d range data for indoor mobile robotics. *Auton. Robots*, 23(2):97–111, 2007.

[11] P. Smith, I. Reid, and A. Davison. Real-time monocular SLAM with straight lines. In *Proc. British Machine Vision Conference*, Edinburgh, 2006.

[12] J. B. S.T. Pfister, S.I Roumeliotis. Weighted line fitting algorithms for mobile robot map building and efficient data representation. In *Proceedings. ICRA '03. IEEE International Conference on Robotics and Automation, 2003.*, 2003.