Relative Pose Estimation from Straight Lines using Parallel Line Clustering and its Application to Monocular Visual Odometry

Naja von Schmude[1,2], Pierre Lothe[1] and Bernd Jähne[2]
[1]*Computer Vision Research Lab, Robert Bosch GmbH, Hildesheim, Germany*
[2]*Heidelberg Collaboratory for Image Processing, Ruprecht-Karls-Universität Heidelberg, Heidelberg, Germany*

Keywords: Relative Pose Estimation, Lines, Clustering, Monocular Camera, Visual Odometry.

Abstract: This paper tackles the problem of relative pose estimation between two monocular camera images in textureless scenes. Due to a lack of point matches, point-based approaches such as the 5-point algorithm often fail when used in these scenarios. Therefore we investigate relative pose estimation from line observations. We propose a new approach in which the relative pose estimation from lines is extended by a 3D line direction estimation step. The estimated line directions serve to improve the robustness and the efficiency of all processing phases: they enable us to guide the matching of line features and allow an efficient calculation of the relative pose. First, we describe in detail the novel 3D line direction estimation from a single image by clustering of parallel lines in the world. Secondly, we propose an innovative guided matching in which only clusters of lines with corresponding 3D line directions are considered. Thirdly, we introduce the new relative pose estimation based on 3D line directions. Finally, we combine all steps to a visual odometry system. We evaluate the different steps on synthetic and real sequences and demonstrate that in the targeted scenarios we outperform the state-of-the-art in both accuracy and computation time.

1 INTRODUCTION

Relative pose estimation is the problem of calculating the relative motion between two or more images. It is a fundamental component for many computer vision algorithms such as visual odometry, simultaneous localization and mapping (SLAM) or structure from motion (SfM). In robotics, these computer vision algorithms are heavily used for visual navigation.

The classical approach to estimate the relative pose between two images combines point feature matches (e.g. SIFT (Lowe, 2004)) and a robust (e.g. RANSAC (Fischler and Bolles, 1981)) version of the 5-point-algorithm (Nister, 2004). This works well under the assumption that enough point matches are available, which is usually the case in structured and textured surroundings.

As our target application is visual odometry in indoor environments (e.g. office buildings) where only little texture is present, point-based approaches do not work. But lots of lines are present in those scenes (cf. Figure 1), hence, we investigate lines for relative pose estimation.

In this paper, we present a novel relative pose estimation scheme for lines. In our work, which got inspired by Elqursh and Elgammal (Elqursh and Elgammal, 2011), we propose to start the relative pose estimation process with a new 3D line direction estimation step. Moreover, we then use this direction information through all steps of the processing which enables us to drastically improve both the robustness and the computation time of the relative pose estimation.

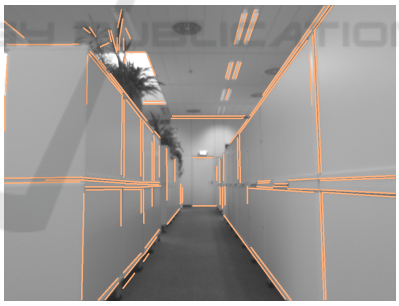


Figure 1: A typical indoor scene with few texture in which a lot of lines are detected.

Our contributions are:

- an improved line clustering algorithm for estimation of the 3D line directions per image with a novel clustering initialization.

Schmude, N., Lothe, P. and Jähne, B.
Relative Pose Estimation from Straight Lines using Parallel Line Clustering and its Application to Monocular Visual Odometry.
DOI: 10.5220/0005661404210431
In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2016) - Volume 3: VISAPP*, pages 423-433
ISBN: 978-989-758-175-5

- a new line matching scheme in which only lines with corresponding 3D line directions are considered. The proposed scheme improves the robustness of the line matching since there is until now no "SIFT" equivalent for lines.
- a robust and fast framework combining all necessary steps for relative pose estimation using lines.

2 RELATED WORK

The trifocal tensor is the standard method for relative pose estimation using lines. The trifocal tensor calculation requires at least 13 line correspondences across three views (Hartley, 1994; Hartley and Zisserman, 2004). For two views, it is in general not possible to estimate the camera motion from lines as shown by Weng et al. (Weng et al., 1992), unless further knowledge of the observed lines can be assumed. If for example different pairs of parallel or perpendicular lines are given, as it is always the case in the "Manhattan world"[1], the number of required views can be reduced to two.

As the number of views is reduced, the number of required line correspondences declines as well, because only five degrees of freedom have to be estimated (three for the rotation and two for the translation up to scale) compared to 26 for the trifocal tensor. Problems with small degrees of freedom are beneficial for robust methods like RANSAC as the number of iterations can be decreased. Concerning the method proposed in this paper, only two views are required where the rotation is estimated from a minimal of two 3D line direction correspondences and the translation from at least two intersection points.

In their work, Elqursh and Elgammal (Elqursh and Elgammal, 2011) employ the "Manhattan world" assumption and require only two views as well. They try to find "triplets" of lines, where two of the lines are parallel and the third is perpendicular to the others. The relative pose can then be estimated from one triplet. The pose estimation process is split up into two steps: first, the vanishing point information of the triplet is used to calculate the relative rotation. Then, the relative translation is estimated using the already calculated rotation and intersection points between lines. The detection of valid triplets for rotation estimation is left over to a "brute force" approach, in which all possible triplet combinations are tested through RANSAC. As the number of possible triplets is in $O(n^3)$ for n lines, this computation is very expensive. Contrarily, our rotation estimation method is more efficient as we calculate it from 3D line directions and the number m of different 3D line directions per image is much smaller than the number of lines (in our cases $m < 10$ whereas $n > 100$). In addition, we do not need the restricting "Manhattan world" assumption which would require orthogonal dominant directions but a less stricter form where we allow arbitrary directions.

Similar approaches requiring two views were presented by Wang et al. (Wang et al., 2008) and Bazin et al. (Bazin et al., 2010). In both works, the pose estimation is split up into rotation and translation estimation as well, where the rotation calculation relies on parallel lines. Bazin et al. estimate the translation from SIFT feature point matches. Their approach is also optimized for omnidirectional cameras. Our method requires only lines and no additional point feature detection as we calculate the translation from intersection points.

Computer vision systems requiring relative pose estimation are e.g. visual odometry, SLAM and SfM. Several line-based SfM methods exist like (Weng et al., 1992; Hartley and Zisserman, 2004; Bartoli and Sturm, 2005; Schindler et al., 2006) which formulate the SfM problem as a nonlinear optimization. The initial configuration is calculated using the trifocal tensor or is derived from other input data. The approach from Schindler et al. (Schindler et al., 2006) takes "Manhattan World" assumptions into account and includes a vanishing point clustering on pixel level. Our method could be used in these SfM algorithms as an alternative for initialization requiring only two views. An EKF-based SLAM method called "StructSLAM" (Zhou et al., 2015) has recently been published which extends a standard visual SLAM method with "structure lines" which are lines aligned with the dominant directions. Witt and Weltin (Witt and Weltin, 2013) presented a line-based visual odometry algorithm using a stereo camera setup. The relative pose is estimated using a nonlinear optimization of the 3D line reconstruction similar to ICP which requires the use of a stereo camera whereas our approach is designed for monocular cameras. In section 3, we present our relative pose estimation approach: first, we introduce a method to estimate line directions through parallel line clustering using a single image (section 3.3). In section 3.4, we describe a novel matching algorithm where only lines with the same 3D line direction are considered. Afterwards, the relative pose is estimated as described in section 3.5. In chapter 4, we present a visual odometry system based on our relative pose

[1] A scene complies with the "Manhattan world" assumption if it has three dominant line directions which are orthogonal and w.l.o.g. can be assumed to coincide with the x-, y- and z-axis of the world coordinate system.

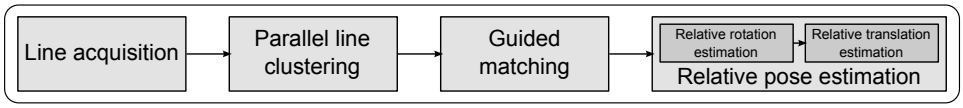


Figure 2: Process chain for relative pose estimation.

estimation method. We finally evaluate our method in section 5 and show that we outperform the current state-of-the-art.

3 RELATIVE POSE ESTIMATION FROM LINES

In this section, we describe our relative pose estimation pipeline in detail which is visualized in Figure 2.

3.1 Notation

In relative pose estimation, we deal with entities in different coordinate systems: there are objects in the image (denoted by subscript i, like the line $\mathbf{l}_i$ or point $\mathbf{p}_i$) and in the camera frame (denoted by subscript C, like the 3D line direction vector $\mathbf{d}_C$).

A projective mapping is used to project a point $\mathbf{X}_C$ from the camera frame onto the image plane. $\mathbf{K}$ is the known 3×3 calibration matrix of the camera.

$$\mathbf{x}_i = \mathbf{K}\mathbf{X}_C \tag{1}$$

The transformation between two coordinate systems is defined by $\mathbf{T}_{to \leftarrow from} = [\mathbf{R}_{to \leftarrow from}, \mathbf{t}_{to \leftarrow from}]$ with $\mathbf{R}_{to \leftarrow from} \in \mathbb{SO}(3)$ the rotation and $\mathbf{t}_{to \leftarrow from} \in \mathbb{R}^3$ the translation. The transformation from a point $\mathbf{X}_{C_a}$ in camera a into the coordinate system of the camera b is done with

$$\mathbf{X}_{C_b} = \mathbf{R}_{C_b \leftarrow C_a}\mathbf{X}_{C_a} + \mathbf{t}_{C_b \leftarrow C_a} \ . \tag{2}$$

3.2 Line Acquisition

At first, we need to detect lines in the image. We use the line-segment extraction from (Witt and Weltin, 2013), but any other line-segment or line extraction method would do. Prominent other approaches would be the line-segment detector "LSD" (Gioi et al., 2010) or Hough transformation-based approaches (Duda and Hart, 1972). We only use line-segments with a minimum length of 30 pixels.

It is worth mentioning that even if we extract line-segments defined by two endpoints $\mathbf{p}_{i_1}$ and $\mathbf{p}_{i_2}$, we solely operate on the homogeneous line representation which can be calculated from $\mathbf{l}_i = \mathbf{p}_{i_1} \times \mathbf{p}_{i_2}$ in the following steps.

3.3 Parallel Line Clustering

The goal of this phase is to cluster lines of an image which are parallel in the world and to extract the shared 3D line direction for each cluster. This problem is closely related to the vanishing point detection, as the vanishing point $\mathbf{v}_i$ of parallel lines is the projection of the 3D line direction $\mathbf{d}_C$ (Hartley and Zisserman, 2004):

$$\mathbf{v}_i = \mathbf{K}\mathbf{d}_C \tag{3}$$

We suggest to work directly with 3D line directions $\mathbf{d}_C$ and not with the 2D vanishing points. Working in 3D space is beneficial, because it is independent from the actual camera (perspective, fisheye etc.) and allows an intuitive initialization of the clustering. In (3), we introduced how the vanishing point and the 3D line direction are related. Now, we have to transfer the line $\mathbf{l}_i$ into its corresponding 3D expression. The line $\mathbf{l}_i$ is the intersection of the image plane with the plane Π_C. Π_C is the line's equivalent for a ray and contains the camera center and the observed 3D line $\mathbf{L}_C$, we call this plane the "back-projected plane". The normal vector of Π_C is given by (Hartley and Zisserman, 2004):

$$\mathbf{n}_C = \mathbf{K}^{\mathrm{T}}\mathbf{l}_i \tag{4}$$

This relation is visualized in Figure 3.

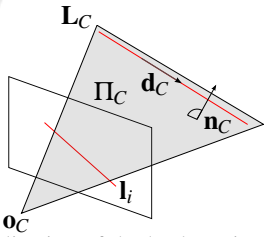


Figure 3: Visualization of the back-projection of an image line $\mathbf{l}_i$. Π_C is the back-projected plane with normal vector $\mathbf{n}_C$. The observed world line $\mathbf{L}_C$ has line direction $\mathbf{d}_C$.

Many of the vanishing point detection algorithms employ the Expectation-Maximization (EM) clustering method (Dempster et al., 1977) to group image lines with the same vanishing point (Antone and Teller, 2000; Košecká and Zhang, 2002). We got inspired by the work of Košecká and Zhang (Košecká and Zhang, 2002) and we adapt their algorithm so that it directly uses 3D line directions instead of vanishing

points. This enables us to introduce a new and much simpler initialization for the clustering in which we directly set initial directions derived from the target environment.

The EM-algorithm iterates the expectation and the maximization step. In the expectation phase, the posterior probabilities $p(\mathbf{d}_C{}^{(k)}|\mathbf{n}_C{}^{(j)})$ are calculated. The posterior mirrors how likely a line $\mathbf{l}_i{}^{(j)}$ (with plane normal $\mathbf{n}_C{}^{(j)}$) belongs to a certain cluster k represented by direction $\mathbf{d}_C{}^{(k)}$. Bayes's theorem is applied to calculate the posterior:

$$p\left(\mathbf{d}_C{}^{(k)}|\mathbf{n}_C{}^{(j)}\right) = \frac{p\left(\mathbf{n}_C{}^{(j)}|\mathbf{d}_C{}^{(k)}\right) p\left(\mathbf{d}_C{}^{(k)}\right)}{p\left(\mathbf{n}_C{}^{(j)}\right)} \quad (5)$$

We define the likelihood as

$$p\left(\mathbf{n}_C{}^{(j)}|\mathbf{d}_C{}^{(k)}\right) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(\frac{-\left(\mathbf{n}_C{}^{(j)\,\mathrm{T}}\mathbf{d}_C{}^{(k)}\right)^2}{2\sigma_k^2}\right) \quad (6)$$

The likelihood reflects that a 3D line in the camera frame (and its direction $\mathbf{d}_C{}^{(j)}$) should lie in $\Pi_C{}^{(j)}$ and is therefore perpendicular to the plane normal $\mathbf{n}_C{}^{(j)}$. If we substitute $\mathbf{n}_C{}^{(j)}$ and $\mathbf{d}_C{}^{(k)}$ with (3) and (4), we come to a likelihood term in the 2D image space which is the same as in the work of Košecká and Zhang (Košecká and Zhang, 2002).

In the maximization step, the probabilities from the expectation step stay fixed. The direction vectors are in this phase re-estimated by maximizing the objective function:

$$\underset{\mathbf{d}_C{}^{(k)}}{\operatorname{argmax}} \prod_j p\left(\mathbf{n}_C{}^{(j)}\right) = \underset{\mathbf{d}_C{}^{(k)}}{\operatorname{argmax}} \sum_j \log p\left(\mathbf{n}_C{}^{(j)}\right) \quad (7)$$

with

$$p\left(\mathbf{n}_C{}^{(j)}\right) = \sum_k p\left(\mathbf{d}_C{}^{(k)}\right) p\left(\mathbf{n}_C{}^{(j)}|\mathbf{d}_C{}^{(k)}\right) \quad (8)$$

As pointed out in (Košecká and Zhang, 2002), in the case of a Gaussian log-likelihood term, which is here the case, the objective function is equivalent to a weighted least squares problem for each $\mathbf{d}_C{}^{(k)}$:

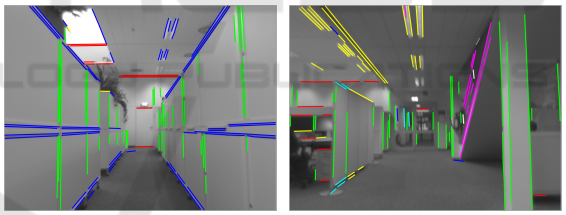
$$\mathbf{d}_C{}^{(k)} = \underset{\mathbf{d}_C}{\operatorname{argmin}} \sum_j p\left(\mathbf{n}_C{}^{(j)}|\mathbf{d}_C\right)\left(\mathbf{n}_C{}^{(j)\,\mathrm{T}}\mathbf{d}_C\right)^2 \quad (9)$$

After each EM-iteration, we delete clusters with less than two assignments to gain robustness.

For initialization, we define a set of 3D directions which are derived from the targeted environment as follows: We apply our method in indoor scenes, hence we find the three dominant directions of the "Manhattan world". In addition, the camera is mounted pointing forward with no notable tilt or rotation against the scene, therefore we use the three main directions $(1\ 0\ 0)^{\mathrm{T}}$, $(0\ 1\ 0)^{\mathrm{T}}$, $(0\ 0\ 1)^{\mathrm{T}}$ for initialization. For robustness, we add all possible diagonals like $(1\ 1\ 0)^{\mathrm{T}}$, $(1\ -1\ 0)^{\mathrm{T}}, \ldots, (1\ 1\ 1)^{\mathrm{T}}$ (e.g. to capture the staircase in Figure 4b)) and end up with overall 13 line directions. All line directions are normalized to unit length and have initially the same probability. The variance of each cluster is initially set to $\sigma_k^2 = \sin^2(1.5°)$ which reflects that the plane normal and the direction vector should be perpendicular up to a variation of 1.5°. Note that this derivation of initial directions is easily adoptable for other scenes or camera mountings. If the camera is for example mounted in a rotated way, we can simply rotate the directions accordingly. If such a derivation is not possible, we suggest to use the initialization technique proposed in (Košecká and Zhang, 2002) where the initial vanishing points are calculated directly from the lines in the image.

If we process an image sequence, we additionally use the directions estimated from the last image in the initialization as "direction priors". In this case, we assign these priors a higher probability.

Results from the clustering step are visualized in Figure 4.



(a) (b)

Figure 4: Results of the parallel line clustering. Lines with the same color belong to the same cluster and are parallel in 3D. Each cluster has a 3D direction vector $\mathbf{d}_C$ assigned which represents the line direction viewed from this image.

3.4 Guided Line Matching

In this section, we describe the matching of lines between two images. As we have clustered parallel lines in both images, we can use this information to guide the matching, so that only clusters of lines with corresponding 3D direction are considered.

For this, we need to establish correspondences between the 3D line directions (the clusters) of the two images. This is done using RANSAC.

The mathematical basis for the algorithm is that the transformation of a direction $\mathbf{d}_{C_1}$ from the first camera to the direction $\mathbf{d}_{C_2}$ in the second camera de-

pends only on the rotation $\mathbf{R}_{C_2 \leftarrow C_1}$:

$$\mathbf{d}_{C_2} = \mathbf{R}_{C_2 \leftarrow C_1} \mathbf{d}_{C_1} \tag{10}$$

In the presence of noise, this equation does not hold, so we use the angular error ε between the directions:

$$\varepsilon = \arccos\left(\frac{\mathbf{d}_{C_2}^{\mathrm{T}} \mathbf{R}_{C_2 \leftarrow C_1} \mathbf{d}_{C_1}}{\|\mathbf{d}_{C_1}\| \|\mathbf{d}_{C_2}\|} \right) \tag{11}$$

The RANSAC process tries to hypothesize a rotation which has a low angular error over a maximized subset of all possible correspondences. A rotation can be hypothesized from two randomly selected direction correspondences as described in 3.5.1. The idea behind this procedure is that only the correct set of correspondences yields a rotation matrix with small angular errors. Therefore, the correct rotation matrix only selects the correct correspondences into the consensus set.

The method has one drawback: it happens always that different rotation hypotheses with different sets of correspondences result in the same angular error. This happens especially in an "Manhattan world" environment. This ambiguity is illustrated in Figure 5. As we

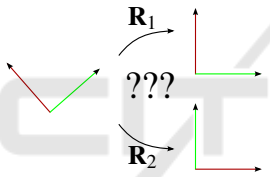


Figure 5: Ambiguity in the direction matching. Which is the correct correspondence assignment for the directions from the left to the right? In this setting, the angular errors for $\mathbf{R}_1$ and $\mathbf{R}_2$ are the same.

have no other sensors in our system to resolve these ambiguities, we assume small displacements between the images and therefore restrict the allowed rotation to less than 45°. As we target a visual odometry system, this is a valid assumption. Alternatively, if e.g. an IMU is present, its input could also be used.

Once the correspondences between the directions (the clusters) are determined, lines in the corresponding clusters can be matched. To match the lines, we apply a descriptor-based matching using the LEHF descriptor from Hirose and Saito (Hirose and Saito, 2012). Here, the gradients in the neighborhood of a line are arranged in histograms which are combined to form the descriptor. The distance between two descriptors is simply the euclidean distance. We add a global threshold of 0.4 and a left-right consistency check (LRC) (Wang et al., 2009) to robustify the matching.

Compared to a brute-force matching, we can reduce the search space for each line to the corresponding cluster and gain robustness.

3.5 Relative Pose Estimation

As the lines between two images are now matched, we can estimate the relative motion $\mathbf{T}_{C_2 \leftarrow C_1} = [\mathbf{R}_{C_2 \leftarrow C_1}, \mathbf{t}_{C_2 \leftarrow C_1}]$ between the two camera locations. We split the estimation into two steps: at first, the rotation is estimated using only the line direction correspondences. Then, the translation is calculated relying on the rotation and intersection points.

3.5.1 Rotation Estimation from Line Directions

As shown in (10), the transformation of a 3D line direction depends only on the rotation. Given m corresponding (and possible noisy) directions, we want to find the rotation $\mathbf{R}_{C_2 \leftarrow C_1}$ which minimizes

$$\mathbf{R}_{C_2 \leftarrow C_1} = \underset{\mathbf{R}}{\operatorname{argmin}} \|\mathbf{D}_{C_2} - \mathbf{R}\mathbf{D}_{C_1}\| \tag{12}$$

where $\mathbf{D}_{C_1}$ and $\mathbf{D}_{C_2}$ are $3 \times m$ matrices which contain in each column the corresponding directions. This problem is an instance of the "Orthogonal Procrustes Problem" (Gower and Dijksterhuis, 2004). We employ the solution presented by Umeyama (Umeyama, 1991) which returns a valid rotation matrix as result by enforcing $\det(\mathbf{R}_{C_2 \leftarrow C_1}) = 1$:

$$\mathbf{R}_{C_2 \leftarrow C_1} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}} \tag{13}$$

with

$$\mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}} = \operatorname{svd}\left(\mathbf{D}_{C_2}\mathbf{D}_{C_1}^{\mathrm{T}}\right) \tag{14}$$

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \operatorname{sign}\left(\det(\mathbf{U})\det(\mathbf{V})\right) \end{pmatrix} \tag{15}$$

At least two non-collinear directions are required to calculate a solution. This algorithm is used in the RANSAC of the guided matching step (cf. 3.4). In practice, the rotation is already estimated there and we do not have to compute it again.

3.5.2 Translation Estimation from Intersection Points

The translation is estimated in the same way as proposed by Elqursh and Elgammal (Elqursh and Elgammal, 2011). Intersection points of coplanar 3D-lines are invariant under projective transformation and therefore fulfill the epipolar constraint (Hartley and Zisserman, 2004):

$$\mathbf{p}_{i_2}^{\mathrm{T}} \mathbf{K}^{-\mathrm{T}} [\mathbf{t}_{C_2 \leftarrow C_1}]_{\times} \mathbf{R}_{C_2 \leftarrow C_1} \mathbf{K}^{-1} \mathbf{p}_{i_1} = 0 \tag{16}$$

If two intersection point correspondences and the relative rotation are given, the epipolar constraint equation can be used to solve $\mathbf{t}_{C_2 \leftarrow C_1}$ up to scale.

As we have no knowledge which intersection points from all $\frac{n(n-1)}{2}$ possibilities (with n the number of line correspondences) belong to coplanar lines, we follow the idea from (Elqursh and Elgammal, 2011) and use RANSAC to select the correct correspondences from all possible combinations while minimizing the Sampson distance defined in (Hartley and Zisserman, 2004). In contrast to (Elqursh and Elgammal, 2011), we can reduce the initial correspondence candidates as we take the clusters into account and only calculate intersection points between lines of different clusters.

As errors on the rotation directly influence the translation estimation, we use the inlier intersection points from the RANSAC step for a nonlinear optimization of the overall pose.

4 APPLICATION: VISUAL ODOMETRY

Now we introduce a visual odometry system using the presented relative pose estimation algorithm.

The pipeline is pictured in Figure 6.

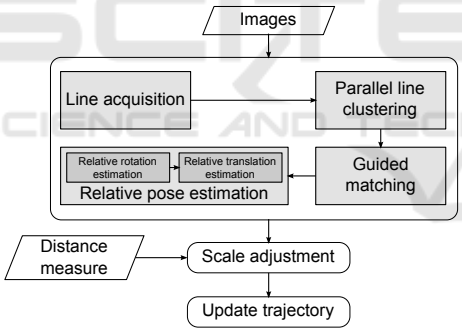


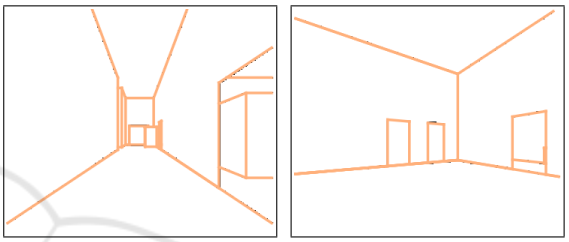
Figure 6: Processing pipeline for the visual odometry.

For each successive pair of images from the input stream, the described relative pose estimation is executed. If for some reason (e.g. not enough line matches found) the relative pose estimation fails, we use the previously estimated relative pose. As the relative pose has no global scale, we need to adjust it to reflect the real distance traveled. We get these distance measures from other sensors, like wheel odometry which is common on robotic platforms. Once the relative motion is properly scaled, we update the trajectory.

We want to mention that this is a very basic approach using only information from two successive frames. The robustness could be improved significantly by using more images (e.g. tracking lines over multiple images and doing a sliding-window bundle-adjustment on the relative poses). We will focus on that in our future work.

5 EVALUATION

For our experiments, we use synthetic and real image sequences. For the synthetic data, we created a typical indoor scene with a 3D wireframe model and generated images from it by projecting the line-segments from the model into a virtual pinhole camera. The generated sequence consists of 1503 images. Example images are shown in Figure 7.



(a) (b)

Figure 7: A synthetic images showing a typical indoor scenario.

To see how image noise affects the processing, we add Gaussian noise on the image lines. We do not add noise on the endpoints of the line-segments, because this effects segments of various length differently. We rather rotate the segments around their center point, where σ is the standard deviation of the rotation in degree.

For the experiments on real data, we recorded a circular sequence in a typical office environment, using an AVT Guppy camera mounted on top of a robot. The image resolution is 1032×778 pixel. The total distance of the "Office Circle" sequence is about 70 m and 1735 images. Example images from the sequence are depicted in Figure 8.

(a) (b)

Figure 8: Examples from the sequence "Office Circle". Note the textureless environment in (a) and (b).

In addition, we use the "Corridor" sequence from

Table 1: Evaluation of the parallel line clustering and the estimated line directions on the synthetic data.

σ (in °)	Cluster Initialization	Avg. error (in °)	Std. dev. (in °)
0.0	Predefined directions	0.4458	3.9923
0.0	Predefined directions + priors	**0.0005**	**0.0018**
0.0	Košecká and Zhang's method	4.9025	12.3382
0.0	Košecká and Zhang's method + priors	0.0052	0.1275
0.5	Predefined directions	3.2674	9.4691
0.5	Predefined directions + priors	**0.5871**	**0.6741**
0.5	Košecká and Zhang's method	9.2458	15.4053
0.5	Košecká and Zhang's method + priors	0.5890	0.6749
1.0	Predefined directions	5.8362	11.5567
1.0	Predefined directions + priors	**2.8989**	**7.0723**
1.0	Košecká and Zhang's method	10.8633	15.5824
1.0	Košecká and Zhang's method + priors	4.7644	9.7788

Oxford[2] which consists of 11 frames showing a corridor with attached ground truth poses. Although this sequence is very short, other methods used it for evaluation which makes a direct comparison possible.

As our test setup is now introduced, we evaluate the different steps of our processing pipeline. First, the performance (5.1) and the computation time (5.2) of the different proposed algorithms are analyzed. Then (5.3), we evaluate the visual odometry system.

5.1 Performance of Proposed Algorithms

5.1.1 Parallel Line Clustering

In this section, we propose to evaluate the accuracy of the 3D line direction estimation using the synthetic data with different noise levels.

For each image, the proposed parallel line clustering is executed, and the calculated line directions are compared to the ground truth. The comparison is done by calculating the angular error between the ground truth and the estimated direction. Since the number of estimated clusters may differ from the actual number, we associate each estimated direction vector with the ground truth direction which results in the smallest angular error.

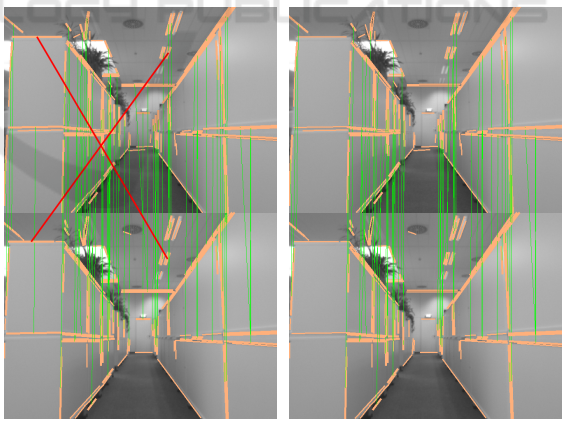
We evaluate our proposed initialization technique where we predefine a set of directions for initialization (cf. 3.3) against the initialization approach proposed by Košecká and Zhang (Košecká and Zhang, 2002) where the initial directions are estimated directly from the line observations in the image. We also combine both techniques with "direction priors" where the estimated directions from the previous image are used additionally.

The results of this experiment are summarized in Table 1. As expected, the accuracy of the estimation drops as more and more noise is added to the data. On this kind of data, we clearly see that using our initialization gives better results compared to the method from Košecká and Zhang. It can also be seen that using direction priors is beneficial as the mean and standard deviation of the angular error is reduced. Therefore we choose the "direction prior" strategy with predefined directions for all following experiments.

[2]http://www.robots.ox.ac.uk/~vgg/data1.html

5.1.2 Matching

We compare the brute-force matching to our proposed guided matching in which only corresponding line clusters are considered. An example is shown in Figure 9.



(a) Brute-force matching. (b) Guided matching.

Figure 9: Brute-force matching vs. guided matching. Detected lines are drawn in orange, matches are indicated by green connections between bottom and top image. The wrong matches (red lines) are avoided in the guided matching because these lines have different 3D directions and belong to different clusters.

This example highlights that our proposed guided matching approach is better in terms of robustness than a brute-force matching.

Table 2: Accuracy of the relative pose computation on the synthetic sequence.

		Rotation error (in °)			**Translation error** (in °)		
Approach	σ in °	Avg.	Median	Std. dev.	Avg.	Median	Std. dev.
"Triplet"	0.0	0.388	0.066	1.006	60.568	55.947	**50.488**
Ours	0.0	**0.008**	**0.000**	**0.017**	**41.019**	**0.054**	55.156
"Triplet"	0.5	31.150	5.293	48.048	79.174	82.673	50.884
Ours	0.5	**1.096**	**0.646**	**1.202**	**76.720**	**76.970**	**43.305**
"Triplet"	1.0	35.049	6.597	49.967	84.459	86.723	46.080
Ours	1.0	**3.264**	**1.368**	**6.048**	**82.161**	**83.711**	**45.271**

5.1.3 Relative Pose Estimation

In this experiment, we compare the accuracy of our relative pose computation with the "Triplet" approach (Elqursh and Elgammal, 2011). We evaluate the rotation and translation error between consecutive image frames, where the rotation error is the rotation angle of $\mathbf{R}_{est}\mathbf{R}_{gt}^{\mathrm{T}}$ and the translation error the angle between the groundtruth translation vector and the estimated translation. The results for the synthetic dataset are listed in Table 2.

The rotation error of our approach is lower than for the "Triplet" approach, which shows that our method is more robust. Also, in the noise-free test the translation error of our method is significantly lower (median of 0° compared to 56°). For the tests under noise, both translation errors rise a lot. This is because the intersection points in the images do not correspond anymore and the epipolar constraint equation therefore yields wrong results.

On the "Corridor" sequence, we estimate the rotation with a mean error of 1.06°. The authors of the "Triplet" approach reported a mean error of 1.3° (we measured 0.97°). For the translation, we measured a mean error of 15.54° for our method and 26.01° for the "Triplet" algorithm as they do not reported this value. We can deduce that our method is more accurate in estimating the relative pose since the rotation accuracy is similar and the translation is estimated with higher accuracy.

Overall, we can conclude that our method outperforms the triplet approach in terms of accuracy and robustness on synthetic and real data sequences.

5.2 Execution Times

In the following, we analyze the runtime of the different steps. All experiments were conducted on a Intel® Xeon™ CPU with 3.2 GHz and 32 GB RAM. The average execution times in milliseconds per image are listed in Table 3.

For the matching itself, our proposed guided matching and the brute-force approach have comparable runtimes. But the guided matching requires the clustering step, so we have to take the time for the clustering into account. Combining the timings of guided matching and clustering, we are slightly slower than the brute-force matching. Nevertheless, the higher robustness of the guided matching compensates for the prolonged execution time.

Our approach for the relative pose estimation step significantly outperforms the "Triplet" approach by only requiring 0.9% to 3% of its runtime. This is due to the fact that in the "Triplet" approach all possible $O(n^3)$ triplet combinations (n is the number of line matches) are generated and then tested in a RANSAC scheme to calculate the rotation which is very time consuming. Contrary to that, our estimation is based on the precalculated line-directions of the clustering step. As we normally extract only around 5 different line directions, the rotation calculation is very fast. Also the number of generated intersection points is lower due to the restriction to intersection points generated from lines from different clusters. We conclude that the additional runtime spent for the clustering is largely compensated by the enormous improvement in the relative pose estimation step.

5.3 Visual Odometry System

The preceding experiments showed that our proposed algorithms give accurate results and are computational inexpensive. Hence, we plug them together to create a robust and fast visual odometry system (c.f. section 4) which is evaluated now.

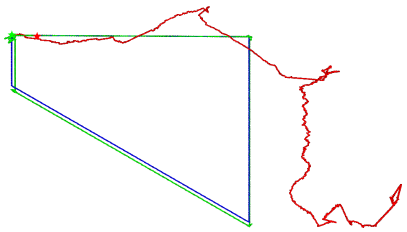
We evaluate the visual odometry system on the synthetic and real image sequences and compare it to the results that one would reach when interchanging the relative pose estimation block with the "Triplet" approach.

The trajectories generated from noise-free synthetic data applying the "Triplet" and our approach are shown in Figure 10.

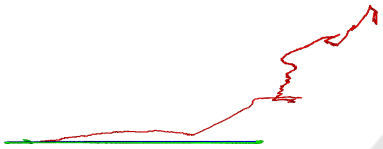
We analyze the root mean squared error (RMSE) of the relative pose error (RPE) and the absolute trajectory error (ATE) introduced by Sturm et al. (Sturm et al., 2012) in Table 4. The RPE is calculated between two consecutive frames.

Table 3: Average execution times in milliseconds.

Sequence	Avg. lines per image	Line Acquisition	Clustering	**Matching**		**Rel. Pose Estimation**	
				Brute-Force	Guided (our)	"Triplet"	Our
Synthetic ($\sigma = 0°$)	31.32	-	0.65	-	-	501.38	**10.06**
Office Circle	121.35	46.24	6.35	**3.71**	3.73	3230.76	**28.29**
Corridor	120.09	34.28	18.22	**3.17**	3.49	639.27	**16.71**



(a) Top view.



(b) Side view.

Figure 10: Top view and side view on the trajectories generated with the "Triplet" approach (red) and our approach (green) of the synthetic image sequence. The ground truth path is drawn in blue. Start and endpoint are represented by the green and red star.

Table 4: RPE / ATE analysis on the synthetic sequence.

Approach	σ in °	**RPE** RMSE	**ATE** RMSE
"Triplet"	0.0	0.073	22.577
Ours	0.0	**0.044**	**0.419**
"Triplet"	0.5	0.099	21.901
Ours	0.5	**0.093**	**13.466**
"Triplet"	1.0	0.106	23.113
Ours	1.0	**0.101**	**19.038**

Trajectories and analysis of RPE and ATE show that our method succeeds much better in reconstructing the ground truth trajectory as the "Triplet" approach.

The average runtime per image is for the triplet approach 524 ms and for our approach 26 ms.

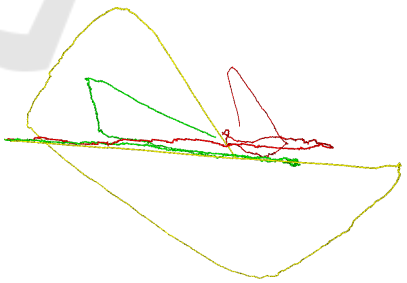
For the real data sequences, we compare our method also to a point-based approach using SIFT (Lowe, 2004) feature matching and a RANSAC-based 5-point-algorithm (Nister, 2004) for relative pose estimation. We are aware that there are better point-based visual odometry systems (e.g. using sliding-window bundle adjustment) but we choose this approach since it is very similar from its structure.

For the "Corridor" sequence, our method results in an RPE of 0.027 and an ATE of 0.052. For the triplet approach we measure an RPE of 0.046 and ATE of 0.079. Using the point-based approach, the RPE is 0.007 and the ATE 0.010. This confirms the previous experiments that our method has a higher accuracy then the "Triplet" approach. But on this sequence, the point-based approach is clearly better as the images are textured and enough points are detected.

The average runtime per image is for the triplet approach 687 ms, for the point-based approach 282 ms and for our approach 73 ms.

The result for the office sequence is presented in Figure 11.

(a) Top view.



(b) Side view.

Figure 11: Top view and side view on the trajectories generated with the "Triplet" approach (red) and our approach (green) and the 5-point-algorithm (yellow) of the "Office Circle" sequence. The original path is drawn in blue. Start and endpoint are represented by the green and red star.

For the "Office Circle" sequence, the "Triplet" approach fails to reconstruct the path completely shortly after the first corner. In addition, the relative pose estimation fails two times. The same can be observed for

the point-based approach, here it fails four times because too few SIFT feature matches are found. Contrary, our algorithm processes the whole sequence. We observe, that with our algorithm the reconstructed trajectory resembles the true trajectory the most, since it has almost a rectangular form and less variation in the height dimension then the other approaches. This indicates that our rotation estimation is more robust. Please note that due to inaccurate distance measurements of our robot, the overall scaling of the trajectories is not correct.

The average runtime per image on the "Office Circle" sequence is for the triplet approach 3277 ms, for the point-based approach 146 ms and for our approach 85 ms.

We can conclude that our method is better suited for the visual odometry task than the "Triplet" approach. We also demonstrated that in low textured environments a line-based approach can function as replacement for point-based relative pose estimation.

6 CONCLUSION

In this paper, we presented a novel relative pose estimation scheme using lines. In our approach, we estimate the 3D line directions through a clustering step of parallel lines in the world and use this information throughout the whole processing pipeline. The direction information is used to guide the line matching and to calculate the relative rotation. We also presented a visual odometry using our relative pose estimation.

As the 3D line direction estimation is such an important step, we evaluated it on synthetic data and showed how the usage of "direction priors" in a sequential processing boosts the accuracy.

Furthermore, we compared our relative pose estimation to the state-of-the-art approach from Elqursh and Elgammal (Elqursh and Elgammal, 2011). We showed that our method outperforms theirs in terms of accuracy and runtime. Especially the runtime can be reduced from seconds to milliseconds.

The visual odometry system is evaluated and compared to the state-of-the-art for line-based relative pose estimation (Elqursh and Elgammal, 2011) and a comparable point-based algorithm (Nister, 2004). We showed that our method is better applicable in textureless indoor scenarios than both other approaches.

In the future, we want to extend our approach to a complete SLAM system. To reach this goal, we need to relax the restriction to small motions in the guided matching step. How this could be done is future work.

REFERENCES

Antone, M. E. and Teller, S. (2000). Automatic recovery of relative camera rotations for urban scenes. In *IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000*, pages 282–289.

Bartoli, A. and Sturm, P. (2005). Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. *Computer Vision and Image Understanding*, 100(3):416–441.

Bazin, J., Demonceaux, C., Vasseur, P., and Kweon, I. (2010). Motion estimation by decoupling rotation and translation in catadioptric vision. *Computer Vision and Image Understanding*, 114(2):254–273.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38.

Duda, R. O. and Hart, P. E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15.

Elqursh, A. and Elgammal, A. (2011). Line-based relative pose estimation. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3049–3056.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Gioi, R. v., Jakubowicz, J., Morel, J.-M., and Randall, G. (2010). Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):722–732.

Gower, J. C. and Dijksterhuis, G. B. (2004). *Procrustes problems*, volume 3. Oxford University Press.

Hartley, R. I. (1994). Projective reconstruction from line correspondences. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 903–907.

Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.

Hirose, K. and Saito, H. (2012). Fast line description for line-based slam. In Bowden, R., Collomosse, J., and Mikolajczyk, K., editors, *British Machine Vision Conference 2012*, pages 83.1–83.11.

Košecká, J. and Zhang, W. (2002). Video compass. In Goos, G., Hartmanis, J., Leeuwen, J., Heyden, A., Sparr, G., Nielsen, M., and Johansen, P., editors, *Computer Vision — ECCV 2002*, volume 2353 of *Lecture Notes in Computer Science*, pages 476–490. Springer Berlin Heidelberg, Berlin, Heidelberg.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Nister, D. (2004). An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770.

Schindler, G., Krishnamurthy, P., and Dellaert, F. (2006). Line-based structure from motion for urban environments. In *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, pages 846–853.

Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, pages 573–580.

Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380.

Wang, G., Wu, J., and Ji, Z. (2008). Single view based pose estimation from circle or parallel lines. *Pattern Recognition Letters*, 29(7):977–985.

Wang, L., Neumann, U., and You, S. (2009). Wide-baseline image matching using line signatures. In *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, pages 1311–1318.

Weng, J., Huang, T., and Ahuja, N. (1992). Motion and structure from line correspondences; closed-form solution, uniqueness, and optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(3):318–336.

Witt, J. and Weltin, U. (2013). Robust stereo visual odometry using iterative closest multiple lines. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, page XX.

Zhou, H., Zou, D., Pei, L., Ying, R., Liu, P., and Yu, W. (2015). Structslam: Visual slam with building structure lines. *IEEE Transactions on Vehicular Technology*, page 1.