Feature Based RGB-D SLAM for a Plenoptic Camera

Andreas Kühefuß, Niclas Zeller, Franz Quint
Karlsruhe University of Applied Sciences
Moltkestr. 30, 76133 Karlsruhe, Germany
andreas.kuehefuss, niclas.zeller, franz.quint@hs-karlsruhe.de

Uwe Stilla
Technische Universität München
Arcisstr. 21, 80333 Munich, Germany
stilla@tum.de

***Abstract*—This paper presents a method to estimate the camera poses for images of a plenoptic camera. For this, a feature based RGB-D SLAM is used. A new method for matching the features between two images will be presented. Finally the result of the algorithm is compared with the trajectory from the Google Project Tango.**

Index Terms - RGB-D; SLAM; feature based; bundle adjustment; focused plenoptic camera

I. Introduction

With rising computing power SLAM (simultaneous localization and mapping) algorithms will gain more and more importance. In this paper a feature based RGB-D SLAM (red-green-blue-distance SLAM) algorithm for a plenoptic camera is presented. Plenoptic cameras are able to deliver (with a certain accuracy) depth information from a single image, which in turn allows the SLAM algorithm to generate a scaled 3-D map and a scaled trajectory. In our approach the SLAM works on RGB images and corresponding depth maps generated with a Raytrix R5 plenoptic camera.

Several SLAM algorithms are known from literature. A. Davison presented in [1] for example an Extended Kalman filter (EKF) based monocular SLAM that is able to recover a 3D trajectory for a uncontrolled camera with a frame-rate of 30 Hz. A keyframe-based SLAM algorithm has been presented by G. Klein in [2] to recover a 3D trajectory.

II. The Plenoptic Camera

The model for a plenoptic camera described in [3] is used in this paper. A plenoptic camera has, in contrast to a normal pinhole camera, a micro lens array (MLA) between the main lens and the image sensor. This leads to a point from object space being projected, as shown in Fig. 1 not only to a single image point but to several, which are located in different micro images. By finding the points in the micro image which correspond to the same object point, the so called virtual image point can be calculated for each object. Each virtual image point is characterized by an associated virtual depth which is defined as the distance between the virtual image point and the MLA in relation to the distance between the sensor and MLA, cf. eq. (1). Please refer also to Fig. 1 for the definition of the variables.

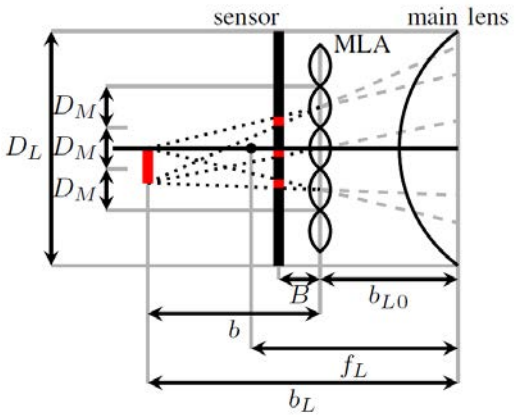

Fig. 1. Optical path inside a plenoptic camera. The MLA projects a single object to different points on the sensor. [3]

$$v = \frac{b}{B} \tag{1}$$

Substituting eq. (1) in the Fresnel-equation for pinhole cameras, the object distance z_C can be calculated from the virtual depth

$$z_C = \left(\frac{1}{f_L} - \frac{1}{B \cdot v - b_{L0}} \right)^{-1}. \tag{2}$$

The object distance is used to project an image coordinate $\boldsymbol{x}_i$ into camera coordinate $\boldsymbol{x}_C$ with

$$\boldsymbol{x}_C = z_C \cdot (K_S \cdot K)^{-1} \cdot \boldsymbol{x}_i. \tag{3}$$

The matrices K_S and K are defined in [3] and contain the intrinsic camera parameters, as shown in (4).

$$\boldsymbol{x}_C = z_C \cdot \left(\begin{pmatrix} s_p^{-1} & 0 & 0 & c_x \\ 0 & s_p^{-1} & 0 & c_y \\ 0 & 0 & B^{-1} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} b_L & 0 & 0 & 0 \\ 0 & b_L & 0 & 0 \\ 0 & 0 & b & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \right)^{-1} \cdot \boldsymbol{x}_i \tag{4}$$

The parameter b_L is the distance of the virtual image to the optical center of the main lens and is calculated using the Fresnel equation for the main lens

$$b_L = \left(\frac{1}{f_L} - \frac{1}{a_L} \right)^{-1}. \tag{5}$$

III. Feature Acquisition

This section describes which feature types are used and how features are matched between images. First the raw images of a plenoptic camera have to be converted into RGB and depth map images. For this, the method presented in [4] is used. After feature extraction, a matching between features in two consecutive images is established.

A. Image acquisition

The scene is recorded with a plenoptic camera. After recording, all raw images are converted to RGB and depth-map images, which in our case have the size 1024 x 1024 pixels. To do this, we use the method presented in [4]. The depth map contains for each pixel a virtual depth, which is coded as a 16-bit grayscale value. Conversion to the virtual depth v cf. eq. (1) is performed as suggested in [5] with

$$v = \frac{1}{1 - \frac{grayscale_{16-bit}}{65535}}. \tag{6}$$

This virtual depth is used together with the internal camera parameters f_L, B and b_{L0}, which have been determined in a previous calibration step, to calculate the object distance z_C with eq. (2). Finally, the camera coordinates of a point can be calculated with eq. (3) out of the image coordinates.

Due to the small baseline between the micro images projected by the MLA, estimation of the virtual depth from a single image of a plenoptic camera is possible within a limited accuracy. Depth estimation can be improved using the larger baseline between successive images of a video. For this, corresponding points have to be matched. This is done with a new method, which we call Slope Matching and which is presented in the following.

B. Slope Matching

Interesting points in the images of the video sequence are detected using the SURF feature detector [6]. After detecting all SURF features in two consecutive images, a match for each feature from the first image is searched in the second. This is done by looking for nearest neighbour with the method presented in [7] and considering the feature quality attributes delivered by the SURF detector. The resulting matches for two consecutive images are shown in Fig. 2.

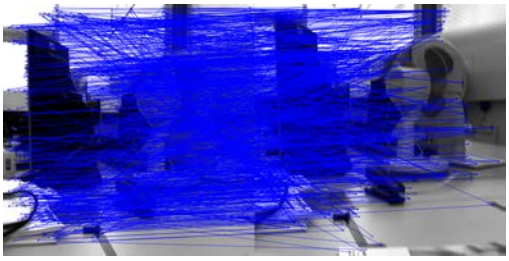

Fig. 2. Resulting matches for two consecutive images, after next neighbor search.

As one can see in Fig. 2 there are many wrong matches which have to be removed. To remove the wrong matches, we just stick the two images side by side and calculate the slope of the line connecting the matched points (blue lines in Fig. 2) with

$$m_i = \frac{y_{i,2} - y_{i,1}}{(x_{i,2} + 1024) - x_{i,1}}. \tag{7}$$

After calculating the slope for each match, the median of all slopes is computed. Each match will be marked as wrong if it doesnt satisfy condition (8), where ϵ is a suitably chosen threshold.

$$m_{median} - \epsilon \leq m_i \leq m_{median} + \epsilon \tag{8}$$

Since the rotation and the translation between two consecutive images of a video is small, the slope for each match should be within the limits given by ϵ. Inequation (8) removes all matches with slopes that differ significantly from the median.

After this operation there still might remain wrong matches, e.g. between a feature from the very left of the first image to a feature of the very right in the second image, but having the slope similar to the median. To remove these wrong matches, the two pictures are stacked one above the other and the previous step is repeated. Mathematically this is done by calculating the inverse slope $m_i^{'}$ of the matches

$$m_i^{'} = \frac{x_{i,2} - x_{i,1}}{(y_{i,2} + 1024) - y_{i,1}}. \tag{9}$$

Again the median will be determined and a constraint similar to the one presented in ineq. (8) removes all matches with a wrong slope. The result of both slope filters is shown in Fig. 3.

Fig. 3. Resulting matches for two consecutive images, after next neighbor search and slope filters.

The resulting matches will be used for pose estimation between two consecutive images.

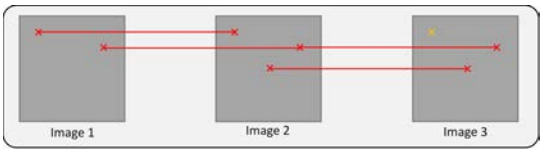

Fig. 4. Feature chain between three images.

The baseline can be further extended, if matches are not restricted to consecutive images. For this we use a feature chain as shown in Fig. 4. If a feature is matched between the first and the second and also between the second and the third image, the feature from the first image is linked to the feature from the third image. The yellow x in the third image shows an feature that could not be matched. The feature chain is broken for this feature.

IV. Pose Estimation

Pose estimation is divided into two parts: First initial poses are estimated between two consecutive images. After estimating enough poses, a local bundle adjustment will be performed.

A. Transformation Matrix

To estimate the pose for two consecutive images, the transformation matrix has to be estimated. The transformation matrix is defined by three rotations and three translations. These six parameters form the camera vector $\boldsymbol{a}$ as shown in eq. (10) and need to be estimated for every new image.

$$\boldsymbol{a} = \begin{pmatrix} \alpha & \beta & \gamma & t_x & t_y & t_z \end{pmatrix}^T \tag{10}$$

B. Initial pose estimation

Instead of estimating the twelve (mutually dependent) elements of the transformation matrix in homogeneous coordinates between the camera coordinate systems of two images, we estimate directly the six elements of $\boldsymbol{a}$ with an iterative procedure.

The first pose is estimated using the results of the slope matching shown in Fig. 3. First the features are projected from the image coordinate system in the camera coordinate system using eq. (4). Now they can be transformed from the camera coordinates corresponding to the first image in the camera coordinates corresponding to the second image. This is done by multiplying the coordinates with the transformation matrix T as shown in (11).

$$\boldsymbol{x}_C^{'} = T \cdot \boldsymbol{x}_C \tag{11}$$

The transformation matrix T is the one which is determined by the vector $\boldsymbol{a}$ to be estimated. At the beginning of the iterative process, the matrix is initialized with the identity matrix.

The coordinates of the features from the first image in the camera coordinate system of the second image can be back-projected in the image coordinate system if the second image by using the inverse of eq. (3), as shown in (12).

$$\boldsymbol{x}_i^{'} = \frac{1}{z_C} \cdot (K_S \cdot K) \cdot \boldsymbol{x}_C^{'} \tag{12}$$

Note that the matrix K_s is the same for both images since the camera remains unchanged. The matrix K however needs to be adapted, since it depends on b_L, which is a function of the virtual depth (see eq. (5)). After transforming every feature of the first image into the second image (initially with the identity matrix), there will remain a reprojection error for each match, expressed as the difference between the position vector of a feature from the first image transformed to the second image and its match in the second image (eq. (13)). The position vectors are three-dimensional since they describe points in the virtual image. The third dimension in the virtual image is the virtual depth.

$$\boldsymbol{r}_j(\boldsymbol{a}) = \boldsymbol{x}_{i_j}^{'} - \boldsymbol{x}_{i_j} = \begin{pmatrix} x_{i_j}^{'} \\ y_{i_j}^{'} \\ v_{i_j}^{'} \\ 1 \end{pmatrix} - \begin{pmatrix} x_{i_j} \\ y_{i_j} \\ v_{i_j} \\ 1 \end{pmatrix} \tag{13}$$

The goal is to find the camera vector $\boldsymbol{a}$ (and thus implicitly the transformation matrix T) which minimizes the mean reprojection error over all matches:

$$\boldsymbol{a}^{'} = \min_{\boldsymbol{a}} \sum_{j=1}^{n} ||\boldsymbol{r}_j(\boldsymbol{a})|| \tag{14}$$

To solve the minimization problem, the Gauss-Newton algorithm is applied. To minimize the impact of outliers, Tukeys biweight cost function is used [8]. This function suppresses outliers by weighting them with zero, as shown in (15).

$$w_{Tb} = \begin{cases} \left(1 - \dfrac{\boldsymbol{r}_j^2}{\sigma^2}\right)^2 & |\boldsymbol{r}_j| \leq \sigma \\ 0 & |\boldsymbol{r}_j| > \sigma \end{cases} \tag{15}$$

By weighting large outliers with zero, wrong matches will not influence the estimation. For calculating the weight w_{Tb}, the empirical standard deviation σ of the errors is used.

C. Bundle adjustment

To optimize the camera position for more than just two images a local bundle adjustment [9] is performed. The bundle adjustment is called local since is not carried out over all images (frames) of the sequence, but only from a so called key-frame to the next key-frame. A new key-frame is set, if the number of matches in a feature chain (see Figure 4) falls below a threshold. In our experiments we have set a new key-frame when the number of matches has fallen below a threshold. However, even if the number of matches exceeds the threshold, every sixth frame is set as a key-frame to ensure that

the bundle adjustment is executed with a sufficient frequency to prevent a drift.

The m images contained between the two key-frames participate in the local bundle adjustment with their camera vectors $\boldsymbol{a}_k$. Consider that for a total of n world points $\boldsymbol{b}_j$ matches have been found. The world points are given by their homogeneous coordinate vector

$$\boldsymbol{b}_j = \begin{pmatrix} x_{w_j} & y_{w_j} & z_{w_j} & 1 \end{pmatrix}^T. \tag{16}$$

The residual error of the world points, projected in the corresponding frames depends of course on the camera vectors $\boldsymbol{a}_k$ as shown in eq. (17)

$$\boldsymbol{r}_{jk}\left(\boldsymbol{a}_k, \boldsymbol{b}_j\right) = \boldsymbol{x}^{'}_{i_{jk}}\left(\boldsymbol{a}_k, \boldsymbol{b}_j\right) - \boldsymbol{x}_{i_{jk}}. \tag{17}$$

The image coordinates used to calculate the error are obtained by projecting the world points in the corresponding frames using eq. (18), which is similar to eq. (12)

$$\boldsymbol{x}^{'}_{i_{jk}}\left(\boldsymbol{a}_k, \boldsymbol{b}_j\right) = T_{a_k} \cdot \frac{1}{z_{w_k}} \cdot \left(K_S \cdot K\right) \cdot \boldsymbol{b}_j. \tag{18}$$

The task of the bundle adjustment is to find the parameters which minimize the mean error over the n world points and m camera positions with

$$\left\{\boldsymbol{a}^{'}_k, \boldsymbol{b}^{'}_j\right\} = \min_{\boldsymbol{a}_k, \boldsymbol{b}_j} \sum_{k=1}^{m} \sum_{j=1}^{n} \left|\left|\boldsymbol{r}_{jk}\left(\boldsymbol{a}_k, \boldsymbol{b}_j\right)\right|\right|. \tag{19}$$

To minimize the effect of outliers, we use in this step also Tukeys biweighted function (15).

V. Evaluation

To evaluate the presented RGB-D SLAM algorithm, two experiment setups have been made. The first is a set of 40 images with known ground truth and the second is a set of 1974 images where the trajectory is compared with the one estimated by the Project Tango of Google. Example images for both experiments are shown in Fig. 5 and Fig. 6.

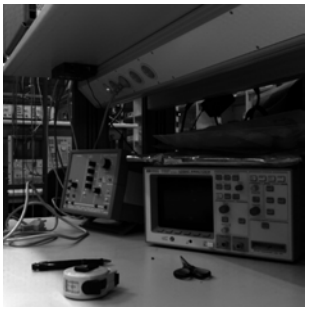

Fig. 5. Image 20 / 40 of the first experiment

Fig. 6. Image 800 / 1974 of the second experiment

A. Comparison with ground truth

The first experiment contains a small number of 40 images which have been recorded in a known grid. First, the camera was moved 19 cm to the right in one cm steps. After that the camera was moved 10 cm back, again in steps of 1 cm each. At last, the camera was moved 19 cm back to the left, in 2 cm steps. The last step amounted again to one cm. Thus the camera was at the end of the movement 10 cm behind its start position. The result of the RGB-D SLAM without a bundle adjustment is shown in Fig. 7. The blue cones mark the position and the orientation of the camera as estimated by our SLAM algorithm without bundle adjustment. The white cones show the ground truth.

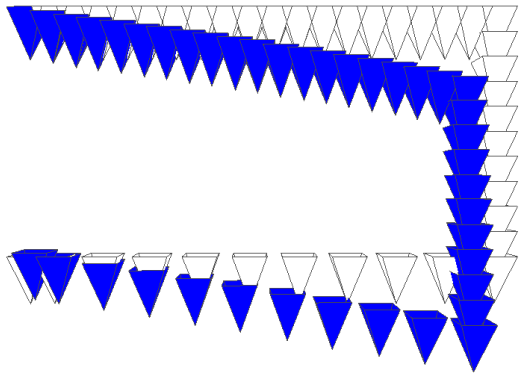

Fig. 7. Result for the first experiment, without bundle adjusment. Blue cones mark the estimated trajectory, the white ones the ground truth.

As one can see in Fig. 7 there is a big drift when not using local bundle adjustment. The trajectory for the same images with local bundle adjustment is shown in Fig. 8.

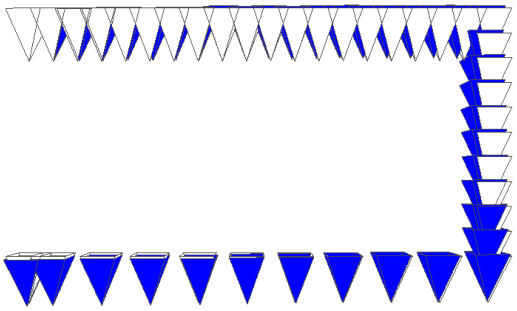

Fig. 8. Result for the first experiment, with bundle adjusment. Blue cones mark the estimated trajectory, the white ones the ground truth.

As one can see in Fig. 8 the estimated trajectory fits well to the known ground truth. The drift error is compensated by the local bundle adjustment.

B. Comparison with Google Project Tango

The second experiment was done on a longer trajectory, using 1974 images. To evaluate the result, the experiment was recorded in parallel with the plenoptic camera and with a tablet

fixed to the plenoptic camera. On the tablet Google Project Tango was running. The software of Project Tango estimates a 3D trajectory using a accelerometer and a monocular camera [10]. The trajectories resulting from our RGB-D SLAM and from Project Tango are compared qualitatively. Of course the result of Google Project Tango is not the actual ground truth, but also subject to errors. However, it is a quite good estimate.

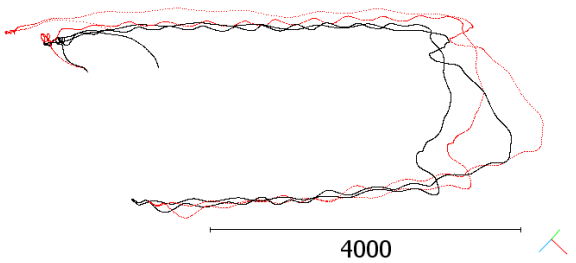

Fig. 9. Result for the second experiment. Red trajectory is from RGB-D SLAM and the black one is from Google Project Tango. Scale is in mm.

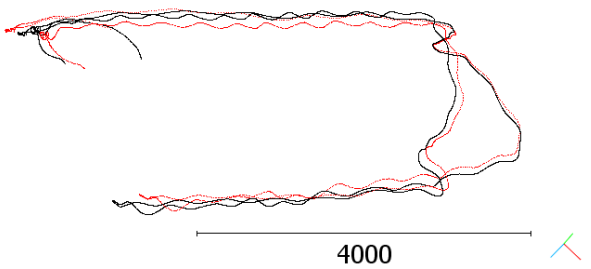

Fig. 10. Result for the second experiment. Red trajectory is from RGB-D SLAM with corrected scale and the black one is from Google Project Tango. Scale is in mm.

The result of the second experiment is shown in Fig. 9. The estimated trajectory matches qualitatively with the trajectory from Google Project Tango, but as one can see there is a scaling error in the data. For better comparison we estimated the scale error using CloudCompare [11]. After correcting our trajectory with the estimated scale, which amounted to 1.125, the comparison can be performed easier. It is shown in Fig. 10.

As one can see in Fig. 10, the two trajectories fit quite well. There is still a very small offset, observable in the top left of the figure. To minimize also this offset a global bundle adjustment could be performed.

VI. Conclusion

The presented method works quite well for typical videos recorded with the plenoptic camera. The matching method supplies good and enough matches for the trajectory estimation to work stable, even for a long scene as shown in the second experiment. Nevertheless there remain some unsolved problems like the scaling error shown in Figure 9. This error has been corrected up to now only interactively and an automated procedure has to be implemented. Furthermore would a global bundle adjustment using the key-frames only help to achieve higher stability against drift.

VII. Acknowledgment

This research was done in the Bachelor Thesis of Andreas Kühefuß. It is part of the project Mosyko3D, which is funded by the Baden-Wrttemberg-Stiftung in its program Photonics, Microelectronics, Information Technology. We gratefully acknowledge the support.

References

[1] A. Davison, I. Reid, N. Molton and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM" IEEE Transactions on Pattern Analysis and Machine Intelligence Vol. 29, nr. 6, p. 1-16, 2007.
[2] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces", Proc. International Symposium on Mixed and Augmented Reality, ISMAR'07, Nara, 2007.
[3] N. Zeller, F. Quint and M. Stterlin, "Investigating Mathematical Models for Focused Plenoptic Cameras", Proc. of International Symposium on Electronics and Telecommunications ISETC2016, p. 285-288, Timisoara, 2016.
[4] N. Zeller, F. Quint, U. Stilla, "Establishing a Probabilisitc Depth Map from Focused Plenoptic Cameras", Proc. International Conference on 3D Vision (3DV), p. 91-99, Lyon, 2015.
[5] "Converting Depth Data," *www.raytrix.de*, Raytrix GmbH, 2013.
[6] H. Bay, T. Tuytelaars, L. Van Gool, "SURF: Speeded Up Robust Features", Journal on Computer vision and image understanding, vol. 110, nr. 3, p.346-359.
[7] M. Muja, D. Lowe, "Fast Approximate Nearest Neighbors With Automatic Algorithm Configuration", International Conference on Computer Vision Theory and Application VISSAPP'09, INSTICC Press, p. 331-340, 2009.
[8] R. Maronna, D. Martin, V. Yohai: Robust statistics: Theory and methods, Wiley Series in Probability and Statistics, Chichester: John Wiley & Sons, 2006.
[9] R. Hartley, A. Zisserman: Multiple View Geometry in Computer Vision. Cambridge University press, 2004.
[10] "Google Project Tango", *www.google.com/tango/*, 2016.
[11] "CloudCompare", *www.cloudcompare.org*, 2016.