6-DoF Camera Position and Posture Estimation Based on Local Patches of Image Sequence

Takuto Tsuji, Masayuki Ikebe, Shinya Takamaeda, Masato Motomura and Tetsuya Asai
Graduate school of Information Science and Technology, Hokkaido University
Kita 14, Nishi 9, Kita-ku, Sapporo, Hokkaido 060-0814, Japan
E-mail: {tsuji, ikebe}@lalsie.ist.hokudai.ac.jp

Abstract

We propose real-time camera position and posture estimation with six degrees of freedom (6-DoF) based on local patches of an image sequence. Our method alternately performs camera motion calculation and depth map reconstruction. Using the reconstructed depth map on only edge regions of images, our estimation method was 30% faster than that based on all pixels and achieved robust motion tracking compared with the feature-points-based method.

1. Introduction

Real-time camera position and posture estimation play an important role in the research of three dimensional (3D) reconstruction, AR (augmented reality) and VR (virtual reality). They have many applications in various fields such as robotics, movie industry and image processing. Our algorithm, which estimates camera poses and depth maps, is commonly called Monocular SLAM (simultaneous localization and mapping). In SLAM, since the system mostly uses only the feature points of images, the tracking of rapid camera motion is difficult. On the other hand, when the system uses all the pixels of images, the calculation cost greatly increases. In this study, we proposed a camera position and posture estimation method to enable the use of only image patches around edge regions. Our algorithm achieves a computational cost reduction and robust motion tracking.

Our contributions are as follows,
- Reconstructing the depth map of only edge regions using an image sequence
- Estimating the camera pose from a sparse depth map

2. Related Works

Real-time monocular SLAM is classified roughly into two types of methods, feature-points-based methods and density-based ones.

2.1 Feature-points-based method

This type of method extracts feature points (e.g., corners of domains) from a reference image. Next, the features are tracked in frame sequences is obtained until a sufficiently wide baseline. Then, the camera pose is estimated by triangulation using five related points in the previous/current frames. The pose is continuously estimated using the camera pose and the tracked feature points in point clouds in 3D space.

By applying bundle adjustment, predictive pose estimation and pyramid strategy to the basic flow of the feature-based method, reasonable accuracy of the method can be achieved. PTAM (parallel tracking and mapping) [1] is generally cited as a well-known feature-points-based.

2.2 Density-based method

In this type of method, the initialization is first carried out by a feature-points-based method. Next, a depth map of all the pixels is reconstructed and the camera pose is estimated using an energy-minimization procedure assuming constant brightness. Finally, the pose is continuously estimated using the optimized depth map obtained from the image sequences. DTAM (dense tracking and mapping) [2] has recently attracted a great deal of attention as a density-based method.

3. System Overview

The operation flow of our system is shown in Fig.1. Our system also performs initialization using the feature-points-based method SLAM. After the initialization, the system obtains a reference image and a set of frames which overlap the camera poses. Then, it estimates the depth map using the image sequence/camera poses and reconstructs the depth map on only the edge region of the reference image because a textureless region is cannot be used for pose estimation. In other words, texture regions have much information about a relative position for estimating the depth map. Thus, our system estimates live camera poses using the reconstructed edge depth map (Fig.2).
4. System Implementation

4.1 Feature-points-based SLAM

We apply the simple SLAM method [3] to the initialization omitting. Detail of the processing flow are described as following. First, the system captures a reference image and extracts feature points from reference image by the FAST (features from accelerated segment test) [4]. Next, it continues tracking the points the KLT (Kanade–Lucas–Tomasi) feature tracker [5] until a live image with a sufficiently wide baseline. Then, the relative camera is estimated pose between the poses of the reference image and live ones is estimated by feature-points tracking using the five-point algorithm [6], [7]. As the final step of this omitting, we construct the first 3D point map using triangulation.

In this initialization process, the accuracy of the first sparse depth map is important because the first error remains and the cumulative error can cause the subsequent process to fail collapses. Figure 3 shows the error of the initialization process. When the prediction of the camera pose is incorrect, each point of interest becomes considerably different; the depth estimation does not converge because of the unmatched triangulations. On the other hand, in the case of good prediction, all points of interests become equal; multiple triangulations work effectively and we can obtain a high-accuracy depth map.

4.2 Proposed method

We describe our proposed method in this section. It uses a similar algorithm to that in a density-based SLAM to update camera poses. In this SLAM, we estimate the depth map of the reference image using only the edge region of the initialized region. Here we explain the processing scheme of the density-based SLAM [2].

\[
T_{mr} = \begin{pmatrix} R_m & t_m \\ 0^T & 1 \end{pmatrix}
\]

(1)

\( T_{mr} \) denotes the matrix describing the camera motion from the reference position \( r \) to another position \( m \); \( R \) is the rotation and \( t \) is the optic center of the camera \( r \).

\[
\pi(p) = (p_x / p_y / p_z)
\]

(2)

A perspective projection is applied to Eq. (2). \( \pi(P) \) denotes the projection function. \( P \) is a pixel in each image.

\[
\pi^{-1}(u, d) = \frac{1}{d}K^{-1}(u^T, 1)^T
\]

(3)
From Eq. (2), the inverse projection is applied to Eq. (3) where $K$ is the camera calibration matrix, $d$ is depth of the image, and $u$ is the projected pixel coordinates [8], [9].

$$\rho(I_r, u, d) = I_r(u) - I_r(\pi(KT_u, \pi^{-1}(u, d)))$$

(4)

From Eq. (1) to (3), photometric error is applied to Eq. (4). Figure 4 shows the photometric error for a checker pattern. When there was a point of interest on a black region, local minimum also existed on the same one. Minimization of the difference between the reference image $I_r$ and the inverse calibrated motion image $I_i$ gives the 3D surface of the projected pixel position. In this way, the depth map can be calculated. Using the obtained depth map, we estimate the camera pose by applying an inverse operation. This operation has global minima which indicate the camera poses in Fig.5. Along with the shifted camera motion, each global minimum can be obtained and the estimation is successful [10]-[12].

In this scheme, calculations using all the pixels are carried out to obtain the depth map and the camera pose. Thus, the density-based SLAM requires a large numbers of calculations. However, regions outside the edges do not have high depth accuracy, because there are no features in the regions. Thus, we use a local patch around the edge to estimate the camera poses. In the edge detection, we apply a 5x5 window Laplacian filter and thresholding. Using the edge-region masking, conditional branching for number of the depth optimization is performed to reduce the calculations (Fig.6).

5. Real-Time 6-DoF Camera Pose Estimation

To evaluate our method, we used a Point Grey Flea2 camera operating at 30 Hz with 640×480 resolution and 24 bit RGB, which was run on a system consisting of an
NVIDIA GTX 480 GPU hosted by an Intel i7 quad-core CPU that operates at 3.07 GHz.

Figure 7 shows the execution time of 30 frames. Our method achieved 30% faster processing (21.56 fps) than for estimation using all the pixels (15.49 fps). Figure 8 shows real-time 6-DoF camera pose estimation. We confirmed successful estimation along with the camera motion.

![Figure 7: Execution time of 30 frames (w/ GPU)](image)

![Figure 8: Real-time 6DoF camera pose estimation](image)

### 6. Conclusions

We proposed 6-DoF camera pose estimation based on local patches. Using the images around edges, our method achieved about 30% faster processing than a density-based SLAM method and reasonable accuracy. We also achieved robust motion tracking compared with the feature-points based method.

### References

[1] G. Klein and D. Murray: Parallel tracking and mapping for small AR workspaces, Proc. International Symposium on Mixed and Augmented Reality, pp. 1-10, Nov. 2007.

[2] R. Newcombe, S. Lovegrove and A. Davison: DTAM: Dense tracking and mapping in real-time, Proc. 2011 International Conference on Computer Vision (ICCV), pp. 2320-2327, Nov. 2011.

[3] H. Uchiyama, T. Taketomi, S. Ikeda and J. P. S. M. Lima: Abecedary tracking and mapping: A toolkit for tracking competitions, Proc. 14th IEEE International Symposium on Mixed and Augmented Reality, pp. 198-199, Oct. 2015.

[4] E. Rosten and T. Drummond: Machine learning for high-speed corner detection, Proc. 9th European Conference on Computer Vision (ECCV), Vol. 1, pp. 430-443, May 2006.

[5] J.-Y. Bouguet: Pyramidal implementation of the Lucas Kanade feature tracker description of the algorithm, OpenCV Documents, Intel Corporation, Microprocessor Research Labs, pp. 1-9, Jan. 2000.

[6] D. Nister: An efficient solution to the five-point relative pose problem, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 2, pp. 195-202, June 2003.

[7] M.I.A. Lourakis and A.A. Argyros: SBA: A software package for generic sparse bundle adjustment, ACM Transactions on Mathematical Software, Vol. 36, No. 2, pp. 1-30, Mar. 2009.

[8] M.A. Fischler and R.C. Bolles: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM, Vol. 24, No. 24, pp. 381-395, June 1981.

[9] E. Rublee, V. Rabaud, K. Konolige and G. Bradski: ORB: An efficient alternative to SIFT or SURF, Proc. 2011 International Conference on Computer Vision (ICCV), pp. 2564–2571, Nov. 2011.

[10] B.D. Lucas and T. Kanade: An iterative image registration technique with an application to stereo vision, Proc. 7th International Joint Conference on Artificial Intelligence, Vol. 2, pp. 674-679, Aug. 1981.

[11] E. Malis: Improving vision-based control using efficient second-order minimization techniques, Proc. IEEE International Conference on Robotics and Automation (ICRA), Vol. 2, pp. 1843-1848, Apr. 2004.

[12] S.J. Lovegrove and A.J. Davison: Real-time spherical mosaicking using whole image alignment, Proc. 11th European Conference on Computer Vision (ECCV), pp. 73-86, Sept. 2010.