Applied Mechanics and Materials Vols 336-338 (2013) pp 348-354 *Online: 2013-07-15*

doi:10.4028/www.scientific.net/AMM.336-338.348

Visual Odometry and 3D Mapping in Indoor Environments

Qinfan Wu[1, a], Qing Li[1, b] and Nong Cheng[1, c]

[1]Research Center of Navigation and Control, Department of Automation, Tsinghua University, Beijing 100084, China

[a]wuqinfan@gmail.com, [b]liqing@tsinghua.edu.cn, [c]ncheng@mail.tsinghua.edu.cn

Keywords: Autonomous Flight; Micro Aerial Vehicle; Kinect; Visual Odometry; 3D Reconstruction.

Abstract. This paper presents a robust state estimation and 3D environment modeling approach that enables Micro Aerial Vehicle (MAV) operating in challenging GPS-denied indoor environments. A fast, accurate and robust approach to visual odometry is developed based on Microsoft Kinect. Discriminative features are extracted from RGB images and matched across consecutive frames. A robust least-square estimator is applied to get relative motion estimation. All computation is performed in real-time, which provides high frequency of 6 degree-of-freedom state estimation. A detailed 3D map of an indoor environment is also constructed.

Introduction

Micro Aerial Vehicle (MAV) is going to play an important role in detection, searching and rescue. Considering some extreme environments, such as a building partially collapsed, sending rescue personnel into these places puts them in great danger. In such circumstances, MAV is able to enter the building and search for survivors and the risk taken by people is greatly reduced. Besides, there are many situations where are dangerous and difficult for humans to approach and where MAV can be employed.

Unfortunately, building a fully autonomous MAV is not an easy task. Autonomous flight of MAV faces great challenges. In typical outdoor environments, autonomous MAVs rely on navigation systems based on combination of Global Positioning System (GPS) and Inertial Measurement Unit (IMU). However, in indoor environments, the following problems should be solved before a fully autonomous MAV is constructed:

1. In GPS-denied environments, no global positioning information is provided. The only sources of information are the sensors equipped on the MAV. Besides, the onboard IMU cannot provide pose and position information with sufficient sensing range and accuracy. Hence fast, accurate and robust navigation system is of great importance.

2. The scale of indoor environments is much smaller than outdoor environments. Therefore, the accuracy required by autonomous flight in indoor environments is much stricter than in outdoor environments. Small error in positioning easily leads to crashes.

3. The payload of small MAV is so limited that sensing device is always limited. The computation resource is also limited. However, many SLAM algorithms are computationally demanding even for powerful desktop computers.

In order to develop a fast and accurate navigation system, we employ vision techniques to retrieve pose and position for the MAV. Recent RGB-D cameras provide both RGB images and per-pixel depth information at video rate. Compared with traditional stereo vision techniques, these sensors generate depth even with poor texture. In our work we use Microsoft Kinect, which captures 640x480 registered RGB image and 3D point clouds at 30Hz. However, such sensors suffer from several drawbacks: low-resolution, limited sensing range, high level of noise, and limited field of view. Although such problems exist, RGB-D cameras offer an attractive approach to visual odometry, since variant low-cost, easy to use sensors are now available.

This paper proposes an approach to real-time, accurate and robust visual odometry. We use Kinect as the prime sensor to get RGB-D images, then extract features from images and match features across

frames to get relative motion estimation of full 6DOF. We take advantage of modern general purpose GPU hardware to achieve real-time performance. Finally, a detailed 3D map of an indoor environment is constructed.

Related Work

Visual odometry refers to the process of incrementally estimating the MAV's relative motion using measurements from vision sensors. A number of researchers have developed various visual odometry approaches using laser range-finders, monocular/stereo vision or RGB-D sensors.

Laser range-finder based approaches [1, 2] typically estimate relative motion by matching laser scans from different directions along the sensing plane. Previous approaches using scan matching algorithms based on Iterative Closest Point (ICP) [3] and Monte-Carlo localization [4] have been implemented on ground moving robots and MAVs. However, laser-based odometry typically impose strong assumptions about the environments in that they require the environment contains vertical structures, and thus the scan matching may fail in homogeneous structures such as long corridors. In addition, since laser range-finders can only obtain distance measurements in a single 2D sensing plane, they cannot make use of the 3D structure features outside sensing plane. As a result, the laser range-finder based approach is not feasible for complex 3D environments

There is also a significant amount of research on camera based approaches using monocular [5] or stereo vision [2] techniques. Vision-based odometry approaches commonly calculate ego-motion estimations by detecting and matching features across frames captured by monocular and stereo cameras. Although vision-based odometry have been proven to be effective in several applications of MAVs, these methods sometimes may fail due to lack of visual texture or extreme motion blur. In addition, extraction of 3D depth information using the stereo vision is typically computationally demanding, which is a significant drawback for its real-time applications.

Building accurate and consistent 3D models and maps of an environment is one of the essential tasks for the indoor Simultaneous Localization and Mapping (SLAM) problem, and there exists a large amount of techniques for 3D modeling and mapping of indoor environments using stereo sensors in various vision-SLAM frameworks. In general, most of these SLAM techniques rely on spatial alignment of consecutive depth data provided by stereo sensors.

Previous researches have demonstrated laser-based environment modeling approaches using 3D depth data collected by active laser devices. Most of these approaches employ ICP [3] and its variants [6, 7] (such as KinectFusion project [8]) for the alignment of depth data as ICP-based methods can take advantage of the full 3-D depth information. In each step of ICP, correspondences between point clouds of two consecutive frames are computed first, and then the transformation which minimizes the distance between the corresponding points is calculated to reconstruct the relative 3D position of these point clouds. However, ICP is prone to convergence at local minima and thereby fails to get correct transformation. In addition, ICP-based methods are not feasible for large-scale environments due to the slow convergence rate and the difficulty in parameter selections.

In contrast, vision-SLAM approaches utilize monocular or stereo technique to recover 3D depth data of the environment. For stereo systems, visual features are extracted from images and the corresponding depth is calculated using the relative position of the two cameras. In contrast, the full 3D depth information cannot be directly obtained by using monocular RGB camera. To solve this problem, many monocular approaches [9] use Structure from Motion (SFM) techniques to recover depth information and achieve real time performance. Although effective in many applications [10], vision-based method may fail to obtain depth information due to lack of textures.

Visual Odometry

Visual odometry can be described as the problem of estimating full 3D motion of the MAV navigating in an unknown indoor environment, using consecutive RGB-D images captured by the RGB-D sensor. In this paper, the visual odometry system is based on an improved framework from stereo

vision algorithms. Discriminative features are extracted from RGB images and these features are matched across frames. Then these matched 2D points are projected into 3D world to establish 3D correspondences from different viewpoints, and finally least-square techniques are employed to robustly estimate camera motion. This section presents the description of each step of the algorithm and compares its implementation to some state-of-art approaches.

Feature Detection. First, color images acquired from the RGB-D camera are converted into grayscale and preprocessed for feature detection. There exists several widely used keypoint detectors, including Harris Corner [11], SIFT [12], FAST [13], SURF [14] and so forth. In order to make a proper choice for our system from these detectors, we compare computation time of different detectors by applying detectors to a 640x480 sample image captured by Kinect, as shown in Fig. 1.

Detector	Computation Time (ms)
Harris Corner	16.7
SIFT	91.4
SURF	112.3
FAST	2.8
ORB	15.6

Fig. 1. a) Sample RGB image captured by Kinect. b) Comparison of computation time of different feature detectors.

These experiments are conducted on a laptop with Intel Core i5 CPU (2.4GHz). As shown in Fig. 1 b), FAST corner detector achieves best real-time performance. However, FAST corner detector does not provide sufficient sub-pixel accuracy and is susceptive to change in scale and viewpoint. While SIFT and SURF can achieve best robustness to scale, rotation, variations in 3D viewpoint and illumination, as well as affine distortion and image noises compared with other feature detectors, it is comparatively time-consuming to detect features and compute corresponding descriptors. However, this drawback can be tackled by employing GPU based parallel computing for the implementation of feature detection.

In order to achieve a decent trade-off between speed and robustness, we employ SURF detector and implement it on an off-board computer using GPU-based parallel framework. Using SURF detector, features are extracted from images and descriptors are computed for each keypoint, as shown in Fig. 2. In addition, experiments demonstrate that GPU-based parallel can significantly speed up the feature detection and achieve real-time performance.

Fig. 2. SURF features extracted from RGB images.

Feature Matching. After the extraction of features from images, these features are then matched across consecutive frames by comparing their SURF descriptor values through a procedure of knn-search of k=2. Denote d_1 as the Euclidean distance of SURF descriptors of the best match, and d_2

as that of the second best match, a feature correspondence is declared if the following constraint is satisfied:

$$d_1 < 0.5,\ d_1 / d_2 < 0.7 \quad (1)$$

After this step, there are still incorrect feature matches between frames, which may lead to incorrect 3D correspondences. In order to prune out incorrect feature matches, a RANSAC [15] procedure is employed to select correct matching result from all the matching pairs above. Examples of feature correspondences are demonstrated in Fig. 3.

Fig. 3. Key points matching between two frames after RANSAC procedure.

Motion Estimation. Given the feature correspondences in consecutive frames before and after the MAV has undergone a rotation R and translation t, the relative motion of the MAV between these consecutive frames can be computed. The least-square approach [16] is employed to determine 6DOF rigid motion between two sets of corresponding points by minimizing the mean square errors. In addition, the motion estimation integrates a novel RANSAC framework in order to achieve better robustness to short-scale errors.

Algorithm. 1: Robust Motion Estimation using RANSAC

Data: $\mathbf{X} = \{\mathbf{x}_i\}$ and $\mathbf{Y} = \{\mathbf{y}_i\}$, $i = 1, 2, \cdots, n$

Result: rotation R and translation t which minimize ε

Begin

  $k \longleftarrow 0$

  while $k <$ *max iterations* do

    choose sunset $S^{(k)} = \{(x_1^{(k)}, y_1^{(k)}), (x_2^{(k)}, y_2^{(k)}), (x_3^{(k)}, y_3^{(k)})\}$ from $\mathbf{X}$ and $\mathbf{Y}$ randomly

    compute $R_0^{(k)}$ and $t_0^{(k)}$ using $S^{(k)}$

    consensus set $S_c^{(k)} \longleftarrow S^{(k)}$

    for each $p_i^{(k)} = (x_i^{(k)}, y_i^{(k)}) \in \{(x_4^{(k)}, y_4^{(k)}), \cdots, (x_n^{(k)}, y_n^{(k)})\}$ do

      add $p_i^{(k)}$ to $S^{(k)}$, denote $S^{(k)} \cup \{p_i^{(k)}\}$ as $S_t^{(k)}$

      compute $R_i^{(k)}, t_i^{(k)}$ using $S_t^{(k)}$

      compare $R_i^{(k)}, t_i^{(k)}$ with $R_0^{(k)}, t_0^{(k)}$: if $R_i^{(k)}, t_i^{(k)}$ is almost same as $R_0^{(k)}, t_0^{(k)}$ then

        add $p_i^{(k)}$ to $S_c^{(k)}$

      end

      remove $p_i^{(k)}$ from $S^{(k)}$

    end

    $k \longleftarrow k + 1$

  end

  $k \longleftarrow \arg\max_k \ \text{sizeof}(S_c^{(k)})$

  Compute final R, t using $S_c^{(K)}$

end

After establishing the correspondences of features from 2D color images, these matching pairs are firstly projected back into 3D space using corresponding depth data of each point provided by the RGB-D camera. After that, given these previously acquired 3D correspondences (two sets of points), denoted as $\{x_i\}$ and $\{y_i\}$, the rotation R and translation t is determined by minimizing the following squared error:

$$e^2 = \Sigma_i \| y_i - (Rx_i + t) \|^2 \quad (2)$$

The motion estimation approach describing here uses singular value decomposition (SVD) based method to calculate R and t that minimize e. In order to achieve better robustness to noises, the motion estimation is integrated in a RNASAC-based framework, as shown in Algorithm. 1.

3D Environment Mapping and Modeling

In this section, we present a method to build 3D maps of large-scale indoor environments using the 3D information of acquired by the RGB-D sensor. Unlike conventional SLAM frameworks which utilizes computationally-demanding optimization for global mapping, our approach decouples the motion estimation and modeling, and fused motion states are used in 3D modeling to improve the accuracy of environment models. An optimization process runs opportunistically to detect loop closures and provide correct for global drifts.

Therefore, the 3D mapping algorithm consists of two steps:

1. Find the correspondence between every two consecutive 3D point clouds to get spatial alignment;
2. Perform global optimization to achieve global consistency.

3D Pointclouds Reconstruction. Typically, the 3D reconstruction algorithm builds environment models using 3D point clouds acquired from stereo sensors. In our approach, we use Microsoft Kinect RGB-D cameras to obtain 3D point clouds, as Kinect combines both RGB and depth images by capturing 640x480 registered RGB and depth image at 30Hz. One example of RGB-D image captured by Kinect is shown in Fig. 4, with a RGB image (left) and a depth image (right).

Fig. 4. Example frame from Microsoft Kinect.

A point cloud typically consists of many discrete points, and each of the point can be denoted as $p_i=(x_i, y_i, z_i)$, with x_i, y_i and z_i representing the relative distance in a coordinate which has its origin at the sensing device. Since each individual frame captured by the sensor contains partial 3D data from a particular viewpoint, building global 3D maps requires registration of all frames. This is achieved by determining relative motion between all the consecutive point clouds and transforming all the frames into one single global coordinate.

As described in previous sections, we have proposed a robust RGB-D odometry to estimate the transformation of the Kinect's position and pose. Thereby this information can be used to recover the Kinect's 3D trajectory and obtain spatial alignment of all the frames. First we extract discriminative features from RGB image and match these features across frames. These matched 2D points are then projected into 3D space to obtain 3D correspondence. Finally a least-square technique is used to

estimate relative motion. The motion estimates are then fused with IMU measurement to form more accurate state estimates. Unlike conventional SLAM framework which directly use the motion estimates from visual odometry, we instead use the fused state estimates to determine the relative transformations between 3D point clouds. After that, we can transform all the frames into the one global coordinate and construct 3D models using 3D point clouds. The 3D model built by our algorithm is shown in Fig. 5. The red line in Fig. 5 a) represents the trajectory of Kinect estimated by visual odometry system.

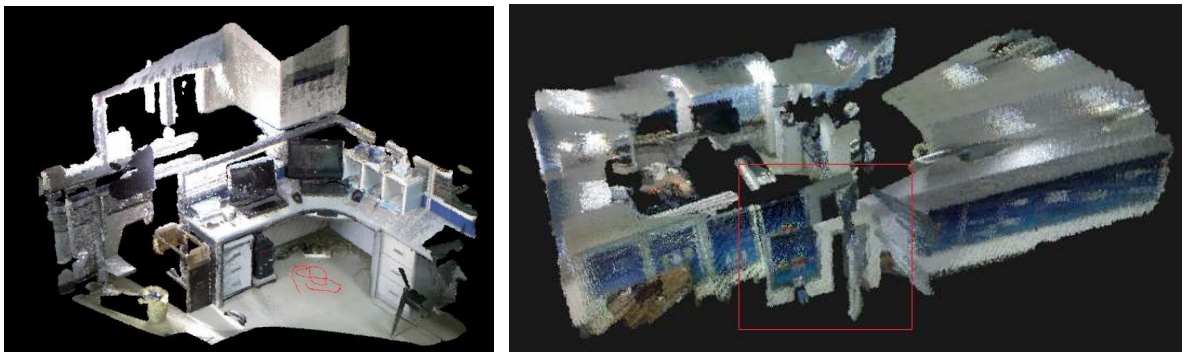

Fig. 5. a) 3D map of part of the environment. b) Global inconsistency in 3D maps.

Global Optimization. After frame-to-frame alignment, we can get the rigid transform from all the frames relative to the reference frame. Then we concatenate all the point clouds and build the overall 3D model of the environment, as described in above sections. However, accumulated error during the frame alignment process may cause significant drift that leads to global inconsistency. This typically occurs when the vehicle returns to a previously visited location after moving through a long path, which is typically known as the loop closure problem in SLAM, as shown in Fig. 5 b).

We solve this problem using sparse pose adjustment approach proposed by [17]. First, a graph of pose is constructed. Each node in the graph contains the corresponded camera pose. Every time two nodes are connected by an edge, a motion constraint is added to the pose graph. Once loop closure is detected, an additional constraint is added to the graph between nodes that are not temporally adjacent. Finally a Levenberg-Marquardt solver is employed to optimize the pose graph. After global optimization, the 3D model of the environment can be improved, as shown in Fig. 6. In our approach, the global optimization runs opportunistically as a back ground process, providing corrections to global drift when desired without affecting the real time performance of the state estimation and mapping process.

Fig. 6. 3D model befor (left) and after (right) global optimization.

Conclusions

In this paper, we developed a visual odometry system that measures Kinect's position and pose when the Kinect is moving in typical indoor environments. Full 6DOF status of the MAV is robustly estimated. This SLAM algorithm can be adapted to unknown and unstructured environments with enough texture. A detailed 3D model of an indoor environment is also constructed. All the computation is performed on General Purpose GPU and achieves real-time performance.

References

[1] S. Grzonka, G. Grisetti and W. Burgard: Towards a navigation system for autonomous indoor flying. Robotics and Automation, 2009. ICRA'09. IEEE International Conference on. IEEE, 2009: 2878-2883.

[2] M. Achtelik, A. Bachrach, R. He, et al: Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments. SPIE, 2009.

[3] P.J. Besl and N.D.McKay: A method for registration of 3-D shapes. IEEE Transactions on pattern analysis and machine intelligence, 1992, 14(2): 239-256.

[4] S. Thrun, D. Fox, W. Burgard, et al: Robust Monte Carlo localization for mobile robots. Artificial intelligence, 2001, 128(1): 99-141.

[5] A.J. Davison, I.D. Reid, N.D. Molton, et al: MonoSLAM: Real-time single camera SLAM. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2007, 29(6): 1052-1067.

[6] S. Rusinkiewicz and M. Levoy: Efficient variants of the ICP algorithm. 3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on. IEEE, 2001: 145-152.

[7] A. Segal, D. Haehnel and S. Thrun: Generalized-icp. Proc. of Robotics: Science and Systems (RSS). 2009, 25: 26-27.

[8] R.A. Newcombe, A.J. Davison, S. Izadi, et al: KinectFusion: Real-time dense surface mapping and tracking. Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on. IEEE, 2011: 127-136.

[9] R.A. Newcombe, A.J. Davison: Live dense reconstruction with a single moving camera. Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, 2010: 1498-1505.

[10] A.S. Huang, A. Bachrach, P. Henry, et al: Visual odometry and mapping for autonomous flight using an RGB-D camera. International Symposium on Robotics Research (ISRR). 2011.

[11] C. Harris and M. Stephens: A combined corner and edge detector. Alvey vision conference. 1988, 15: 50.

[12] D.G. Lowe: Distinctive image features from scale-invariant keypoints. International journal of computer vision, 2004, 60(2): 91-110.

[13] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. Computer Vision–ECCV 2006, 2006: 430-443.

[14] H. Bay, T. Tuytelaars and L. Van Gool. Surf: Speeded up robust features. Computer Vision–ECCV 2006, 2006: 404-417.

[15] 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, 1981, 24(6): 381-395.

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

[17] K. Konolige and G. Grisetti: Efficient sparse pose adjustment for 2d mapping. Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on. IEEE, 2010: 22-29.

Industrial Instrumentation and Control Systems II
10.4028/www.scientific.net/AMM.336-338

Visual Odometry and 3D Mapping in Indoor Environments
10.4028/www.scientific.net/AMM.336-338.348

DOI References
[3] P.J. Besl and N.D. McKay: A method for registration of 3-D shapes. IEEE Transactions on pattern analysis and machine intelligence, 1992, 14(2): 239-256.
http://dx.doi.org/10.1109/34.121791
[4] S. Thrun, D. Fox, W. Burgard, et al: Robust Monte Carlo localization for mobile robots. Artificial intelligence, 2001, 128(1): 99-141.
http://dx.doi.org/10.1016/S0004-3702(01)00069-8
[5] A.J. Davison, I.D. Reid, N.D. Molton, et al: MonoSLAM: Real-time single camera SLAM. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2007, 29(6): 1052-1067.
http://dx.doi.org/10.1109/TPAMI.2007.1049
[15] 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, 1981, 24(6): 381-395.
http://dx.doi.org/10.1145/358669.358692
[16] S. Umeyama: Least-squares estimation of transformation parameters between two point patterns. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1991, 13(4): 376-380.
http://dx.doi.org/10.1109/34.88573