A Fast Visual Odometry and Mapping System for RGB-D Cameras

Bruno M. F. Silva and Luiz M. G. Gonçalves
Department of Computer Engineering and Automation
Federal University of Rio Grande do Norte
Natal, Brazil
{brunomfs,lmarcos}@dca.ufrn.br

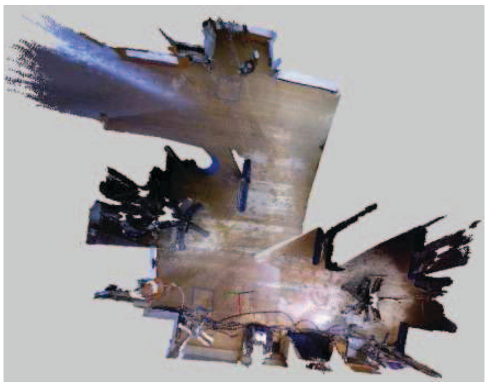
***Abstract*—The introduction of low cost range sensing devices such as RGB-D cameras allows applications for Robotics to exploit novel and real-time capabilities. One such application is Visual Odometry, a module responsible to use the synchronized color/depth streams captured by this class of sensors to estimate the position and orientation of a robot at the same time that a map representation of the environment is built. Aiming to localize robots in a fast and efficient way, we design a Visual Odometry system for RGB-D sensors that allows real-time (approximately 25 Hz) camera pose estimation despite the fact that no specialized hardware (such as modern GPUs) is employed. Experiments carried out on publicly available benchmark and datasets demonstrate the usefulness of the method, which achieved localization accuracy superior to the state-of-the-art RGB-D SLAM algorithm.**

I. Introduction

Recently, low cost depth sensing devices were introduced into the market designed for entertainment purposes. One such sensor is the RGB-D camera called Microsoft Kinect [1], which is capable to deliver synchronized color and depth data at 30 Hz and VGA resolution. Although initially designed for gesture based interfaces, RGB-D cameras are now being employed in scientific applications as for example object recognition [2] and 3D modeling [3]. Previously inaccessible data (range sensing) and hard to solve problems (color and depth registration) are now available and consolidated with RGB-D sensors, enabling sophisticated algorithms for Robotics and for one of its most studied research topics, SLAM (*Simultaneous Localization and Mapping*).

Due to the particularities involved in the process of estimating depth measurements from images, SLAM with pure visual sensors (Visual SLAM) can be considered a nontrivial problem. However, systems relying on Visual SLAM are very efficient in dealing with issues such as data association (by tracking visual features between different images) and loop closing (by detecting previously visited locations using image similarity). Since RGB-D cameras employ accurate depth estimation techniques (e.g. structured light stereo used in the Microsoft Kinect), SLAM can benefit from both visual data and range sensing.

The problem of map building and localization can also be solved by Visual Odometry [4], [5]. In contrast to Visual SLAM, that allows long term localization by associating visual data with a global map, Visual Odometry systems incrementally estimate frame to frame pose transformations. Consequently, faster computation times are prioritized over localization accuracy. In light of this, mapping and localization with RGB-D sensors can be designed to be both fast and accurate.


(a)

(b) (c) (d) (e) (f)

Figure 1. (a) Top view of the resulting 3D map of the sequence *FR1 floor*, obtained by concatenating the RGB-D data of each frame in a single point cloud. The resulting map can be used for robot navigation and obstacle avoidance, despite the fact that no SLAM techniques were used in the reconstruction process. (b-f) Some sample images used to build the map are also shown.

In this work, a Visual Odometry solution for RGB-D sensors is proposed for indoor mobile robots. By tracking and detecting visual salient features across consecutive frames, the pose (position and orientation) of a moving sensor can be computed in a fast and robust manner with RANSAC [6]. Also, by registering RGB-D data utilizing the respective estimated transformation, a map of the environment is built, as illustrated in Fig. 1. The system is evaluated through a set of experiments carried out on public RGB-D datasets [7] to demonstrate the accuracy and computation performance of the method.

In the remaining text, Section II lists the related works on Visual SLAM and Odometry based on RGB-D sensors. Section III describes the inner workings of the proposed system, while the experiments and results are discussed on Section IV. Finally, we close the paper in Section V.

II. Related Work

Robot localization is a well studied problem with solutions emerging from research in Simultaneous Local-

978-1-4799-6711-7/14 $31.00 © 2014 IEEE
DOI 10.1109/SBR.LARS.Robocontrol.2014.35

ization and Mapping (SLAM). Successful strategies are being achieved with range sensors such as lasers [8] or Time of Flight cameras [9]. Solutions with perspective cameras are also employed in monocular [10] and stereo [11] configurations. The work of Nistér et al. [12] introduces the "Visual Odometry" terminology, which refers to systems that employ cameras to estimate the position and orientation of a moving agent. This class of systems is generally formed by incremental *Structure from Motion* [13] approaches that estimate frame to frame pose transformations instead of finding the position relative to a global map (which is the general definition of SLAM). A tutorial paper on the subject was recently pusblished [4], [5].

Once unaffordable, depth measurement devices are now being employed in solutions to the localization and mapping problem with the advent of consumer grade RGB-D sensors. These solutions are either dense [14], [15], [16], [17] or feature based [18], [19], [20], [21], [22] approaches.

Dense solutions [14], [15], [16], [17] directly use the input color/depth data i.e. they do not rely on the extraction and matching of visual features. Steinbruecker et al. [14] propose a method to compute camera motion by estimating the warping transformation between adjacent frames under the assumption of constant image brightness. This work is later complemented [15] to support probabilistic motion models and robust estimators in the transformation estimation, resulting in a system more resilient to moving objects in the scene. Whelan et al. [16] extend the solution of Steinbruecker et al. with a GPU implementation of the algorithm and the incorporation of feature based visual front-ends to the pose estimation process. Osteen et al. [17] take an innovative direction proposing a Visual Odometry algorithm that works by computing correlations between the frequency domain representations of the scene normals.

Alternatively, feature based solutions work by matching extracted features between sequential pair of images and then estimating the corresponding transformation through sets of three point matches and RANSAC [6] and/or *Iterative Closest Point* (ICP) [23]. Henry et al. [18] detect and match FAST [24] features using Calonder descriptors [25] and optimize an error function with RANSAC and a non-linear version of ICP. After detecting loop closures, Bundle Adjustment [26] or graph optimization [27] is executed to ensure global pose consistency. Paton and Kosecka [20] and Hu et al. [19] propose a similar system, although the latter has a strategy to switch between the algorithm of Henry et al. and a pure monocular Visual Odometry algorithm. This strategy is employed to deal with situations in which the available depth data is not sufficient. The detection and extraction of sparse visual features was also the direction taken by Endres et al. when designing RGB-D SLAM [21], which works by extracting SIFT features [28] on a GPU implementation and optimizing the pose graph with g2o [29]. Finally, Stückler and Benhke [22] employ a surfel representation of the scene containing depth and color statistics which are later matched and registered in a multi-resolution scheme.

Our work relies on sparse visual features and incremental pose estimation between adjacent frames. However, in contrast to all mentioned solutions of this class, we do not employ *tracking by detection* in frame to frame matching. Instead of extracting keypoints and matching them using their descriptors, we track features using a short baseline optical flow tracker. As evidenced by the experiments, the system can rapidly estimate accurate camera poses without the need of using state-of-the-art GPUs.

III. Proposed System

A. Overview

Our goal is to estimate the pose (position and orientation) of a moving sensor (a Kinect style RGB-D camera) using only the captured color and depth information. Camera poses are estimated relative to the first RGB-D frame (the origin reference frame). A map consisting of all registered RGB-D frames is also computed at each time step.

The proposed system tracks visually salient features across consecutive image pairs I_{t-1}, I_t, adds keypoints on an as-needed basis and then robustly estimates the camera pose $[R_t|\mathbf{t}_t]$ using the feature points. Each step of the algorithm, depicted on Fig. 2, is explained as follows.

B. Tracking Visual Features

At the initial time step (frame I_0), Shi-Tomasi corners [30] are extracted and invalid keypoints (points without a depth measurement) are removed. The remaining points form the initial set of features $\{\hat{\mathbf{x}}\}_0$, with $\hat{\mathbf{x}} = [x, y, d]^t$, where d is the depth of the point at image coordinates x, y. Then, for each frame I_t with $t > 0$, the points of the set $\{\hat{\mathbf{x}}\}_t$ are tracked through sparse multiscale optical flow [31], assuming a small capture interval between I_{t-1} and I_t and image brightness held constant. Since a point can become lost (or invalid) after tracking, invalid points are removed once again. Then, the resulting set of tracked points $\{\mathbf{x}\}_t$ and the set with the corresponding points on the previous time step $\{\mathbf{x}\}_{t-1}$ are used to estimate the current camera pose $[R_t|\mathbf{t}_t]$, as explained in Section III-C.

We proceed with the following strategy for the management of keypoints being currently tracked. Instead of detecting and adding keypoints only when the number of tracked keypoints is below a threshold, we extract Shi-Tomasi corners at every frame, although each keypoint is only added to the tracker if two conditions are met: the current number of active keypoints is below a threshold τ and the keypoint lies in a region in the image without any tracked point.

To accomplish this, every point $\hat{\mathbf{p}}$ of the set of newly extracted points $\{\hat{\mathbf{p}}\}_t$ is checked to test whether it lies inside any of the rectangular perimeter W_j centered around each tracked point $\mathbf{x}_j$ of the set $\{\mathbf{x}\}_t$. If $\hat{\mathbf{p}}$ is not inside any rectangular region, it is added to the tracker in the set $\{\mathbf{p}\}_t$. Note that it can only be used to estimate the camera pose on the next frame, since it does not have a correspondence

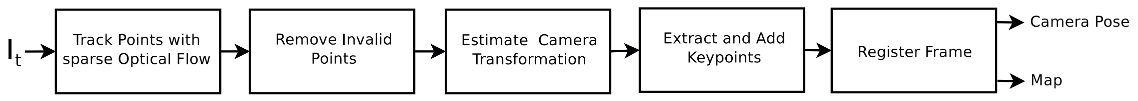

I_t
Track Points with sparse Optical Flow
Remove Invalid Points
Estimate Camera Transformation
Extract and Add Keypoints
Register Frame
Camera Pose
Map

Figure 2. The proposed Visual Odometry method.

at $t-1$ yet. Points with invalid depth measurements once again discarded. This process is illustrated in Fig. 3.

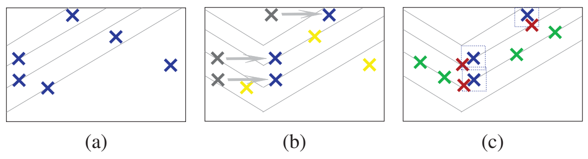


Figure 3. Feature tracking strategy: (a) At I_0, the initial set of features $\{\hat{\mathbf{x}}\}_0$ (blue crosses) is extracted. (b) At I_t with $t > 0$, $\{\hat{\mathbf{x}}\}_t$ (gray crosses) is tracked into $\{\mathbf{x}\}_t$ (blue crosses). Keypoints that could not be tracked are shown in yellow. (c) The set of added keypoints $\{\mathbf{p}\}_t$ is shown with green crosses, whereas the rejected points, which are within the rectangular perimeter (dashed rectangles) of the tracked points, are shown with red crosses.

By varying the maximum number of active points τ and the size of the rectangular windows W_j, different configurations of the algorithm can be achieved, allowing tuning based on application requirements (such as accuracy or performance).

C. Frame to Frame Motion Estimation

The sets containing all tracked points in the current $\{\mathbf{x}\}_t$ and in the previous frame $\{\mathbf{x}\}_{t-1}$ are used to estimate the camera pose $[R_t|\mathbf{t}_t]$. For this, points are reprojected to 3D as

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} d(x-c_x)/f_x \\ d(y-c_y)/f_y \\ d \end{bmatrix},$$

resulting in the sets $\{\mathbf{X}\}_t$ and $\{\mathbf{X}\}_{t-1}$ of 3D points referenced in the camera frames of I_t and I_{t-1} respectively. The terms f_x, f_y, c_x, c_y are the intrinsic parameters of the Kinect RGB sensor. The current camera pose can be estimated by finding the rigid transformation $[R^*|\mathbf{t}^*]$ that minimizes in the least squares sense the alignment error in 3D space between the two point sets, as shows Equation 1. We proceed with the method proposed by Umeyama [32] which uses Singular Value Decomposition to solve for the transformation parameters.

$$R^*, \mathbf{t}^* = \underset{R,\mathbf{t}}{\operatorname{argmin}} \sum_i ||\mathbf{X}_{t-1}^i - \left(R\mathbf{X}_t^i + \mathbf{t}\right)||^2 \quad (1)$$

Since point correspondences $\mathbf{X}_t^i, \mathbf{X}_{t-1}^i$ can be falsely estimated (due to occlusions, bad lighting conditions, the aperture problem inherent to optical flow, etc.), a robust pose estimator should be employed to compute the desired transformation. The RANSAC [6] algorithm accomplishes this function. RANSAC works by sampling a minimal set of data to estimate a model, which in our case are point correspondences and a camera transformation respectively. For a given number of iterations, minimal sets of three point matches are randomly chosen to estimate a hypothesis pose by minimizing Equation 1. All other point correspondences are then used to count the number of inliers (correspondences with alignment error below a specified threshold ϵ). The hypothesis with the largest number of inliers is elected as the winner transformation. Using all the inliers corresponding to the winner solution, a new least squares transformation is computed, resulting in the sought $[R_t|\mathbf{t}_t]$. This last step results in camera poses significantly better than simply taking the R and $\mathbf{t}$ related to the winner transformation and the performance penalty is negligible.

The RANSAC algorithm can run faster by using an adaptive termination of the procedure that takes into account an estimate of the fraction of inlier correspondences [6]. Hence, by updating this fraction with the largest ratio found after each iteration of the algorithm, a significant speedup is achieved.

The last step in our algorithm involves registering the RGB-D frame I_t with that of I_{t-1}. The estimated pose is used to transform the 3D points in the current camera frame to the reference frame relative to the previous camera by applying the $[R_t|\mathbf{t}_t]$ to each point of $\{\mathbf{X}\}_t$. By doing so, the current transformation is always estimated relative to the origin reference frame and a map formed by all registered RGB-D frames until time step t is computed.

D. Algorithm Parameterization

In the current implementation, all rectangular regions W_j of the tracked points have a fixed size. All experiments of Section IV are executed with maximum number of tracked points $\tau = 1000$ and window W_j with size 30×30.

RANSAC parameters are set with a correspondence error threshold ϵ of 8 mm (which automatically rejects the imprecise measurements for features with large depth) and a maximum of 10000 iterations. The reported values for all parameters were found after running the algorithm through several initial experiments.

IV. Experiments and Results

A. Methodology

We evaluate the proposed Visual Odometry system with respect to its localization accuracy and computational performance. All experiments were carried out on a MacBook Pro with an Intel Core i5 3210M 2.5 Ghz processor and 8 GB of RAM. The system is implemented in C++ using Point Cloud Library[1] and OpenCV [2]. The OpenCV library provides a parallelized implementation of the optical flow algorithm based on Intel TBB [33] that was key to achieve the reported performance of the algorithm.

[1]http://pointclouds.org
[2]http://opencv.org

We resort to the datasets and benchmark proposed by Sturm et al. [7] to evaluate our system. Using the provided RGB-D data (with synchronized ground truth collected by a motion capture system), we assess the localization accuracy of the proposed system by estimating the translational and rotational *Relative Positioning Error* (RPE). The RPE is a performance metric suitable to Visual Odometry systems since it gives a measure of the drift (accumulated error) per unit of time. Accordingly, the Root Mean Square Error (RMSE) computed over all possible frame pairs I_i, I_{i+1} (i.e. adjacent frames) is provided as a robust error statistic of a given estimated trajectory. The same error statistics are also computed for the RGB-D SLAM system [21] to clarify how the proposed system compares against state-of-the-art algorithms in RGB-D localization. The results from RGB-D SLAM can also be downloaded from the datasets of Sturm et al. [7].

B. Accuracy Results

The results for the used datasets after executing the algorithm on each image sequence are presented on Table I, which shows the RMSE and maximum translational and rotational drifts per frame for the proposed system and also for RGB-D SLAM.

The proposed Visual Odometry system outperforms RGB-D SLAM regarding the RMSE translational and rotational drifts per frame in all of the used datasets. Improvements in the translational RMSE can be as high as 14 mm, which represents an error reduction of more than 70% (*FR1 plant*), while the rotational error can be decreased by 0.8° or ∼60% (*FR1 plant* and *FR1 teddy*). On average, the RMSE translational and rotational drifts are 9.9 mm/frame and 0.440°/frame for the proposed system and 15.3 mm/frame and 0.929°/frame for RGB-D SLAM, although loop closures and drift minimization are not employed as does RGB-D SLAM.

Figure 4 shows the resulting camera trajectory computed by the proposed system along with the ground truth trajectory. The resemblance between the ground truth and the computed camera trajectories can be noticed from the referred plots.

C. Performance Results

The performance of the proposed algorithm is assessed by collecting the total time spent on the execution of each RGB-D frame. The average, standard deviation and maximum time per frame are shown in Table II.

Table II
PERFORMANCE OF THE PROPOSED RGB-D VISUAL ODOMETRY SYSTEM. SHOWN ARE THE AVERAGE, STANDARD DEVIATION AND MAXIMUM RUNNING TIMES COMPUTED OVER ALL FRAMES OF EACH SEQUENCE.

Sequence	Avg.	Std. Dev.	Max
FR1 desk	27.643 ms	6.383 ms	110.413 ms
FR1 floor	44.284 ms	14.668 ms	165.115 ms
FR1 plant	29.673 ms	7.676 ms	145.000 ms
FR1 teddy	42.486 ms	24.816 ms	201.628 ms

Despite the fact that no specialized hardware (such as modern GPUs) is used by our system, localization estimates can be computed at rates of approximately 27Hz, as evidenced by the computed average over all sequences (36.021 ms per frame). The slowest time spent on a single frame (∼200 ms) is still faster than the reported average of RGB-D SLAM (330/350 ms without/with global optimization respectively) [21]. The varying quantity of valid tracked features and oscillations in the RANSAC runtime explain the variations in the estimated timings collected among different sequences. Specifically on the sequence *FR1 floor*, a large number of keypoints is tracked in each frame due to the large amount of texture in the scene, which results in an almost uniform sampling of points for tracking and thus in a higher execution time.

We note that the proposed algorithm is restricted to operations in texture rich and indoor environments due to failure situations which might occur in textureless environments or in situations with lack of depth measurements.

V. CONCLUSION

In this work, we propose a Visual Odometry system based on RGB-D sensors that estimates camera poses using visual/depth measurments. At each RGB-D frame, visual features are detected and tracked through sparse optical flow with new features being added if they do not fall within regions related to already tracked features. Camera poses computed as the transformation between adjacent frames using RANSAC can then be estimated in a fast and accurate manner.

The accuracy and computation performance of the proposed system is demonstrated by experiments carried out on public available RGB-D datasets and benchmark [7]. The average RMSE translational and rotational drifts are 9.9 mm/frame and 0.440°/frame for the proposed system and 15.3 mm/frame and 0.929°/frame for the state-of-the-art RGB-D SLAM algorithm [21], at the same time that the running times for the proposed algorithm are almost one order of magnitude faster on average. This results are achieved without the use of expensive and specialized hardware such as modern GPUs.

Future works will focus on loop closing and global trajectory optimization algorithms using the Visual Odomety module to play an important role in a full SLAM solution.

ACKNOWLEDGMENT

This work is supported by the Coordination for the Improvement of Higher Education Personnel (CAPES) and the Funding Agency for Studies and Projects (FINEP).

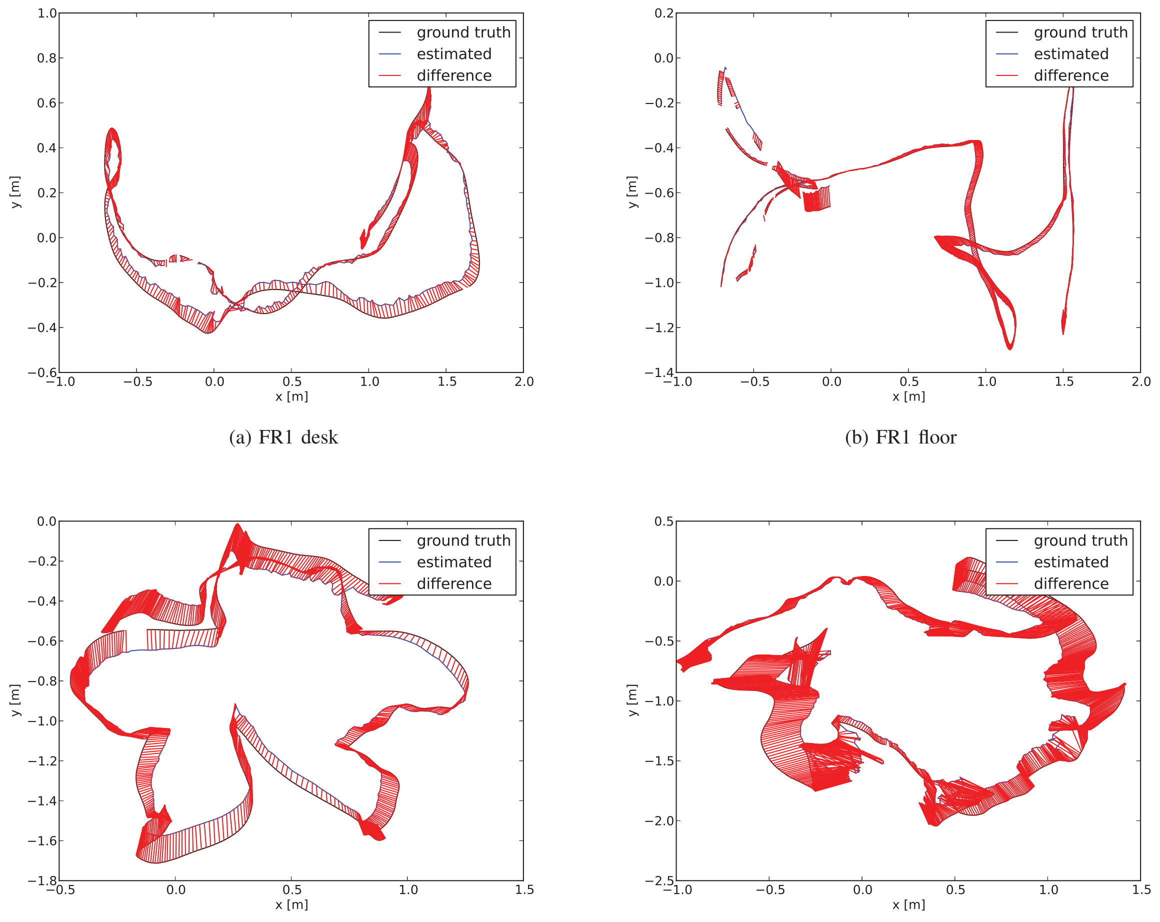
REFERENCES

[1] Microsoft, “Microsoft kinect,” http://www.xbox.com/en-US/kinect, 2014, accessed on April 1st, 2014.

[2] A. Aldoma, Z. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. Rusu, S. Gedikli, and M. Vincze, “Tutorial: Point cloud library: Three-dimensional object recognition and 6 DOF pose estimation,” *Robotics Automation Magazine, IEEE*, vol. 19, no. 3, pp. 80 –91, sept. 2012.

Table I
COMPARISON BETWEEN THE PROPOSED SYSTEM AND THE STATE-OF-THE-ART IMPLEMENTATION OF RGB-D SLAM. SHOWN ARE THE RMSE AND MAXIMUM TRANSLATIONAL AND ROTATIONAL RELATIVE POSITIONING ERROR, WHICH GIVES A MEASURE OF THE RESULTING DRIFT PER FRAME. BEST RESULTS ARE IN BOLDFACE.

Sequence	Prop. System				RGB-D SLAM			
	Transl. RMSE	Transl. Max.	Rot. RMSE	Rot. Max.	Transl. RMSE	Transl. Max.	Rot. RMSE	Rot. Max.
FR1 desk	**0.0106 m**	**0.0506 m**	**0.5471°**	**2.3216°**	0.0117 m	0.0630 m	0.7309°	6.8548°
FR1 floor	**0.0030 m**	**0.0167 m**	**0.2441°**	2.0214°	0.0037 m	0.0271 m	0.2833°	**1.9288°**
FR1 plant	**0.0066 m**	**0.0700 m**	**0.3657°**	**2.1504°**	0.0207 m	0.1210 m	1.2550°	4.1202°
FR1 teddy	**0.0197 m**	0.1818 m	**0.6065°**	**4.7872°**	0.0254 m	**0.1094** m	1.4490°	7.2330°



(a) FR1 desk

(b) FR1 floor

(c) FR1 plant

(d) FR1 teddy

Figure 4. Camera path computed by the proposed Visual Odometry algorithm (shown in blue) in comparison to the ground truth trajectory (shown in black). The red lines are the distance between the computed and ground truth positions.

[3] H. Du, P. Henry, X. Ren, M. Cheng, D. Goldman, S. Seitz, and D. Fox, "Interactive 3D modeling of indoor environments with a consumer depth camera," in *Proc. of the Int. Conf. on Ubiquitous Computing (UbiComp)*. New York, NY, USA: ACM, 2011.

[4] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[5] F. Fraundorfer and D. Scaramuzza, "Visual odometry : Part ii: Matching, robustness, optimization, and applications," *IEEE Robotics Automation Magazine*, vol. 19, no. 2, pp. 78–90, 2012.

[6] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image

analysis and automated cartography," *Communications of the ACM*, pp. 381–395, 1981.

[7] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. of IEEE/RSJ International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[8] P. Newman, D. Cole, and K. Ho, "Outdoor SLAM using visual appearance and laser ranging," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2006, pp. 1180–1187.

[9] S. May, D. Droeschel, D. Holz, S. Fuchs, E. Malis, A. Nüchter, and J. Hertzberg, "Three-dimensional mapping with time-of-flight cameras," *Journal of Field Robotics*, vol. 26, no. 11-12, pp. 934–965, nov 2009.

[10] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proc. of IEEE Int. Conf. on Computer Vision (ICCV)*, 2003.

[11] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

[12] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2004, pp. 652–659.

[13] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[14] F. Steinbruecker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *Workshop on Live Dense Reconstruction with Moving Cameras at the Int. Conf. on Computer Vision (ICCV)*, 2011.

[15] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2013.

[16] T. Whelan, H. Johannsson, M. Kaess, J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense RGB-D mapping," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[17] P. Osteen, J. Owens, and C. Kessens, "Online egomotion estimation of RGB-D sensors using spherical harmonics," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012, pp. 1679–1684.

[18] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments," *International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, April 2012.

[19] G. Hu, S. Huang, L. Zhao, A. Alempijevic, and G. Dissanayake, "A robust RGB-D SLAM algorithm," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012, pp. 1714–1719.

[20] M. Paton and J. Kosecka, "Adaptive RGB-D localization," in *Conf. on Computer and Robot Vision (CRV)*, 2012, pp. 24–31.

[21] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, St. Paul, MA, USA, May 2012.

[22] J. Stückler and S. Behnke, "Integrating depth and color cues for dense multi-resolution scene mapping using RGB-D cameras," in *Proc. of IEEE Int. Conf. on Multisensor Fusion and Information Integration (MFI)*, September 2012.

[23] P. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[24] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. of European Conference on Computer Vision (ECCV)*, Berlin, Heidelberg, 2006, pp. 430–443.

[25] M. Calonder, V. Lepetit, and P. Fua, "Keypoint signatures for fast learning and recognition," in *Proc. of European Conference on Computer Vision (ECCV)*, 2008, pp. 58–71.

[26] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *Vision Algorithms: Theory and Practice, LNCS*. Springer Verlag, 2000, pp. 298–375.

[27] G. Grisetti, C. Stachniss, and W. Burgard, "Nonlinear constraint network optimization for efficient map learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 428–439, 2009.

[28] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[29] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2O: A general framework for graph optimization," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 3607–3613.

[30] J. Shi and C. Tomasi, "Good features to track," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1994, pp. 593–600.

[31] J.-Y. Bouguet, "Pyramidal implementation of the lucas kanade feature tracker," *Intel Corporation, Microprocessor Research Labs*, 2000.

[32] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, Apr. 1991.

[33] Intel, "Threading Building Blocks (TBB)," https://www.threadingbuildingblocks.org/home, 2014, accessed April 1st, 2014.