EasySLAM

Alireza Fathi, Alex Cunningham, Balmanohar Paluri, Kai Ni, and Frank Dellaert

Abstract— **EasySLAM is a robust, accurate, efficient and easy-to-use visual SLAM framework which uses the unique properties of planar landmarks to navigate robots in societal settings. Due to the use of landmarks which can be associated with semantics, a hybrid symbolic-metric SLAM variant is obtained that makes the maps immediately usable for human-robot interaction, high-level monitoring, and semantic analysis.**

EasySLAM associates a set of landmarks to each part of the house (e.g. kitchen, living room, bathroom, bedroom, etc.) and takes navigation commands such as "go to kitchen". Loalization and mapping, planning and navigation results are presented with an inexpensive, commercially available robot and uniquely identifiable markers.

SLAM with planar landmarks is easy, robust, and fills the real need in both research and society, and we have a system that everyone can use.

I. Introduction

In this paper we present EasySLAM, a vision-based simultaneous localization and mapping method that uses planar markers to eliminate all major difficulties with deploying SLAM in real environments.

Simultaneous localization and mapping or SLAM is a crucial capability for mobile robots to successfully navigate in unknown environments. An overview of the recent state of the art in SLAM algorithms can be found in the articles by Durrant-Whyte and Bailey [1], [2] and in the excellent book by Thrun et. al.[3], but the field is still rapidly evolving and novel, exciting algorithms appear regularly [4], [5], [6].

However, there are still significant practical hurdles for anyone wishing to run a SLAM algorithm in a new environment. While the debate is out on whether SLAM is a "solved problem" in theory, the reality facing many researchers is that these theoretically-appealing algorithms are often hard to make work in practice, often requiring a dedicated graduate student to tune and/or tend to quirks of the chosen the system. A pleasant side-effect is that this then often leads to new ideas and new papers on SLAM, but one could argue that SLAM has become a distraction rather than a tool.

The difficulty of making SLAM work in practice is even more pronounced in the "real" world, i.e., when trying to use robots in societal settings such as residential homes, office buildings, industrial workplaces, and out in city streets.

We argue that there is a transitional need for a robust, reliable SLAM solution that makes these problems go away. We have no doubt that visual SLAM will eventually make its way out of the lab and into the real world, and many companies are hard at work to make this a reality.

This work was supported by the National Science Foundation

All authors are with the Georgia Institute of Technology, School of Interactive Computing, 801 Atlantic Drive, Atlanta, GA 30332, USA

Corresponding author is Frank Dellaert, frank@cc.gatech.edu

SLAM with planar markers is easy, robust, and fills this real need in both robotics research and society. Planar markers with known dimensions exhibit no scale ambiguity, eliminate data association, and provide robust 6DOF measurements from a single sighting.

There are many objections that can be raised against the approach we are proposing. Markers are unappealing visually, and many people might object to placing them in their homes or work-places. However, (1) one can imagine several ways in which planar targets can be made invisible or much less visually disruptive, e.g., by using images and/or artwork, or by working in a spectrum invisible to human eyes; and (2) even for the "ugly" black and white markers, one could argue that both researchers and consumers will make this choice based on a value proposition: if robotic technology is truly useful and a few markers around the environment is what it takes, many might be willing to live with this. Clearly, this is already happening in other domains such as Augmented Reality, but this choice has already been made on much larger scales, e.g., think of the bar-codes one finds on every conceivable product in a grocery store, or -more pertinent to mapping- the ubiquitous traffic signs indicating where you are and how to get to your destination.

We will soon release an open-source implementation of our code that will support robust mapping, localization, and navigation right out-of-the-box. We believe that such a capability can do for robotics and SLAM what Lowe's SIFT implementation [7] did for feature-based methods in computer vision: allow researchers to concentrate on the problems they want to study which might need SLAM as a tool, rather than on making SLAM work.

II. Related Work

We work within the landmark-based SLAM framework. For SLAM, the earliest and most popular methods are based on Extended Kalman Filter (EKF) and are successfully implemented in different environments such as indoors [8], outdoors [9], sub-sea [10] and air-borne [11]. EKF linearizes about the current pose of the robot and position of all landmarks, and recursively estimates a Gaussian density around them. However, the EKF becomes computationally intractable fairly quickly. As a result, a large amount of effort has been devoted to extend EKF-based approaches to cope with larger-scale environments [12], [13]. In addition, EKF filtering methods can be shown to be inconsistent when applied to inherently non-linear SLAM problem [14].

Recently, there has been considerable interest in Smoothing And Mapping (SAM), where the entire robot trajectory is estimated in every time step. Dellaert [15] has shown that

smoothing can be a fast alternative to filtering-based methods, and in many cases smoothing improves performance rather than hurting it. Our localization and mapping approach is based on [15] which we will briefly describe in Section III.

We use ARToolkit markers [16], [17] as planar landmarks in our experiments, however, there has been a significant amount of research on automatic construction of planar landmarks from the features in the environment, that can replace the ARToolkit markers. Hayet et al.[18], [19] use edge grouping to find the quadrangle landmarks. Then they use Harris features to describe a landmark, and use partial Hausdorf distance to find the similarity between two landmarks.

Watkins et al.[20] use PCA and clustering to group the points into planar surfaces in a SLAM framework suitable for Micro Air Vehicles (MAV). Frese [21] and Zhang et al. [22] use artificial landmarks such as red squares or white circles on floors or walls. Berger and Lacroix[23] use homography to group a set of points as a planar landmark if they are consistent as a planar patch. They memorize the texture of planar patches, and compute their image from different views and use it for loop closing.

III. Approach

In this section, we describe the smoothing and mapping approach used by the EasySLAM problem to map the planar landmarks. EasySLAM makes initial estimation of the map by computing the relative poses between the camera and the planar landmarks using homography. Then it uses SAM framework to optimize the map. After map building, EasySLAM provides planning and navigation capability which will be described as well.

A. Initialization

To initialize the robot and marker poses, first we build a spanning tree that contains all the markers and robot poses as follows. We choose one of the markers to serve as a "world reference marker", and use that marker's local coordinate system as a global coordinate system, to which all other marker and robot poses are relative. In each frame, in which the robot observes one or more markers, we calculate their poses with respect to each other and to the robot.

To be able to compute the pose of the robot and markers observed in the current frame, we should find a path of poses that connects them to the reference marker. Among the markers observed in the current frame, we search for a marker that is observed in the past and added to the spanning tree. Since that marker is connected to the robot we will be able to estimate the pose of the robot as well. Having the robot pose added to the spanning tree, we verify if there are any new markers being observed in the current frame. If so, we simply connect them to the robot pose node and add those markers to the list of observed nodes. If we couldn't find a marker that is already added to the spanning tree and failed to connect the poses in the current frame to the previous ones, the robot is lost. We will move the robot and it makes new observations and will be able to find its pose again.

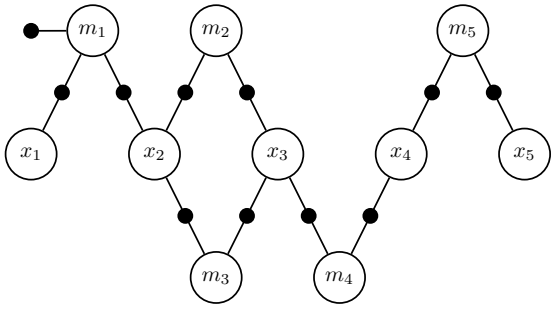


Fig. 1: The factor graph representation of the EasySLAM problem. Robot poses are marked as x's and landmarks as m's. Measurements bear factors (black circles) on edges.

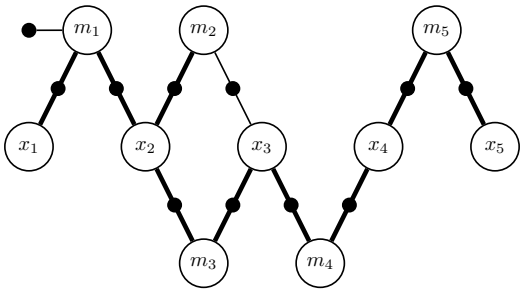


Fig. 2: The spanning tree of a factor graph, which we use to initialize all the poses based on the pose of the reference marker. The edges involved in the tree are shown darker.

Our algorithm basically creates a connected graph of robot and marker poses over the time, from which we extract a spanning tree that contains all the poses. The spanning tree of the factor graph of Figure 1 is depicted in Figure 2.

B. Relative Pose from Planar Landmarks

For planar landmarks of known size we can immediately recover the relative pose of the camera with respect to the landmark [24] and use it for initialization of robot and landmark poses. We define a local coordinate frame placed at the landmark center, with the X and Y axes spanning the landmark plane and the Z axis pointing out of the plane. For a planar marker located at a relative location t, a given point $(\hat{X}, \hat{Y}, 0)$ in the landmark coordinate frame is projected into the camera as

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = K \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix} \begin{bmatrix} \hat{X} \\ \hat{Y} \\ 0 \\ 1 \end{bmatrix}$$

$$= K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} \hat{X} \\ \hat{Y} \\ 1 \end{bmatrix}$$

where u, v, w are the image coordinates, K is the camera calibration, r_1, r_2, and r_3 are columns of a rotation matrix

R (from marker to camera frame). From the second line above it can be readily seen that $(u, v, w)^T$ and $(\hat{X}, \hat{Y}, 1)^T$ are related by a homography $H = K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix}$.

The homography H can be estimated from 4 or more 2D measurements of known points on the planar landmark. However, we can only recover H up to an unknown scale λ:

$$\hat{H} = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} = \lambda K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \quad (1)$$

If scale λ is known we can recover the relative pose as

$$\begin{aligned} r_1 &= \lambda^{-1} K^{-1} h_1 \\ r_2 &= \lambda^{-1} K^{-1} h_2 \\ r_3 &= r_1 \times r_2 \\ t &= \lambda^{-1} K^{-1} h_3 \end{aligned}$$

Luckily, we can recover the scale factor λ by imposing orthonormality on the recovered rotation R, e.g., $\|r_1\| = \|r_2\| = 1$, giving the scale factor either as $\lambda = \left\|K^{-1} h_1\right\| = \left\|K^{-1} h_2\right\|$. To account for inaccuracies in $\hat{H}$, one can take the average of both.

C. Smoothing and Mapping

In this section we review the smoothing and mapping (SAM) method that we use to optimize the robot and marker poses. In SAM, we aim to estimate the unknowns which are the entire robot trajectory X and the map M, given the measurements Z. In particular, we use a factor graph representation to model the SLAM problem as in [25]. A measurement at time i of landmark j introduces a binary factor between the robot's pose, x_i, and the measured landmark, m_j. In Figure 1, the robot poses (x's) and landmark poses (m's) are represented by nodes (unknowns) and there is a binary factor for each measurement. There is also a unary factor representing the prior on the pose of a single landmark that acts as the origin of the map. We ignore any odometry measurements, which would otherwise appear as factors between the robot poses.

The factor graph in Fig. 1 specifies the following joint density over the robot poses X and landmark poses M:

$$P(X, M) \propto f_r(m_r) \prod_k f_k(x_i, m_j)$$

where x_i and m_j are the robot and landmark poses, respectively, $f_r(m_r)$ encodes the prior on the reference landmark m_r, and $f_k(x_i, m_j)$'s denote the binary factors arising from the measurements between a robot pose and an observed landmark.

We obtain the maximum a posteriori (MAP) estimate by maximizing $P(X, M|Z)$ which leads to the following non-linear least squares problem:

$$\begin{aligned} \{X, M\} &= \underset{\{X,M\}}{argmin} \{||g(m_r) - w_r||^2_{\Sigma_r} \\ &+ \sum_k ||h_k(x_i, m_j) - z_k||^2_{\Sigma_k}\} \end{aligned} \quad (2)$$

In equation 2, $h_k(x_i, m_j)$ is the measurement function, given the robot pose x_i and the landmark pose m_j, predicting the observation z_k. We describe the details about the measurements h_k in Section III-D. The expressions $g(m_r)$ and w_r arise form the prior on the reference landmark, which forces it to stay at the origin.

The above equation is non-linear, but we can always relinearize around the current pose estimates. We can linearize each binary factor as below:

$$\begin{aligned} h_k(x_i, m_j) &= h_k(x_i^0, m_j^0) + H_k^i \delta x_i + J_k^j \delta m_j \\ h_k(x_i, m_j) - z_k &= H_k^i \delta x_i + J_k^j \delta m_j - c_k \end{aligned}$$

where H_k^i and J_k^j are the Jacobians of $h_k(.)$ with respect to a change in x_i and m_j respectively, evaluated at the linearization point (x_i^0, m_j^0) , and $c_k = z_k - h_k(x_i^0, m_j^0)$ is the measurement prediction error. The unary factor of the reference landmark is treated similar to the binary factors. As shown in [25] by collecting all the Jacobians into a matrix A, and the measurement errors into a right hand side (RHS) vector b, we can obtain the following standard least square problem,

$$\delta^* = \underset{\delta}{argmin} ||A\delta - b||_2^2 \quad (3)$$

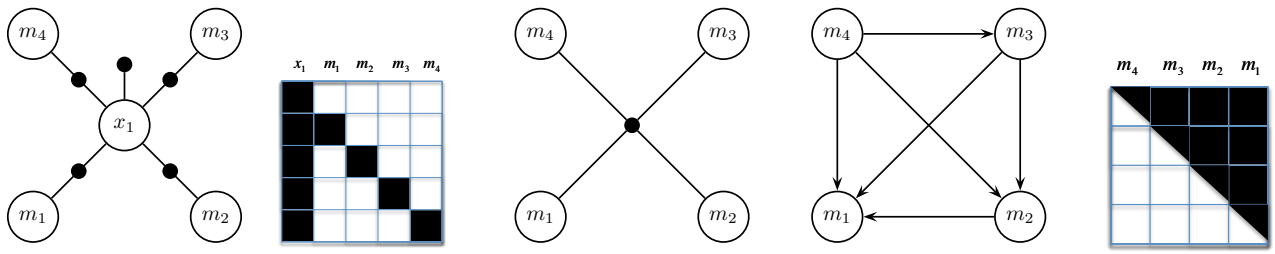
which can be solved using QR factorization or other linear algebraic methods. We first linearize equation 2 using the initial robot and marker poses. After we compute the new pose estimations, we iteratively relinearize around the new estimations and update the estimations, until convergence. We describe the method we use for estimating the initial poses in Section III-B.

D. Measurement Function

The measurement function $h_k(x_i, m_j)$ receives the pose of the robot (camera) and the planar landmark as inputs and returns the pixel locations where the points on the landmark should appear. We can use homography to compute where each of the L points of the planar landmark (given that we know the relative location of the points with respect to the local coordinate frame of the planar landmark) should appear in the image. Since we know the relative pose of marker with respect to the robot's camera ($m_j \ominus x_i$), and we know the relative location of point l on the landmark with respect to its local coordinate frame, we can compute the relative location of each point with respect to the camera. Measurement function h returns a vector of x, y pixel locations of the points on the planar landmark in the image.

In our experiments, we use ARToolKitPlus code to detect the fiducial planar markers in the images and generate the ground truth measurements (z_k) for each marker [16], [17]. A measurement z_k of a single marker is a set of its 4 corners in the image (in pixels). An ID is assigned to each marker based on its unique appearance which takes care of data association.

To linearize the equation 2, we need to compute the derivative of the h_k function at the current estimate of the x_i and m_j. We use an automatic differentiation framework [26] which allows us to efficiently calculate a Jacobian for a given point, free of numerical instabilities.



(a) (b) (c)

Fig. 3: Construction and optimization of the factor graph: (a) The robot (x_1) observes some landmarks. The unary factor connected to x_1 represents its location w.r.t. the reference landmark. (b) By eliminating x_1, we arrive at a factor which is connected to all landmarks seen in the first frame. (c) Elimination of all the variables results in a chordal Bayes net, and is equivalent to performing QR factorization on the matrix.

E. Planning and Navigation

Indoor navigation with planar markers is not as easy as it appears in the first glance. Not only does the robot need to use the markers to localize itself in real time, plan a path to the destination, and follow it with limited commands available for an inexpensive robot, but also it must avoid obstecles. When navigating between waypoints, the robot has knowledge only of the planar markers, but not of obstacles such as walls and furniture. We keep a history of robot poses to solve this problem. At the beginning the robot will be manually operated in the house, and while building a map of envirnoment, it will record its pose at each frame as well. When the robot navigates autonomously between commanded waypoints, it uses, its history of poses to plan its path. The history consists of poses that the robot has occupied, and thus are free of obstacles. The robot can be made aware of changes to the environment later by being driven manually through an area again.

To plan a path between two robot poses, we build a graph in which the nodes are the poses from the history, and the edges encode the distance between poses. We will connect two poses if their distance is less than a threshold. To compute the distance between the poses we consider both their relative translation and rotation. Since the robot is always on the ground, and it can only rotate around camera's y axis, we can present the rotation by an angle. We compute the distance (D) using the following formula:

$$D \quad = \quad \alpha \cdot ||t||_2 + \beta \cdot |\theta|$$

where α and β are coefficients, t is the length of the translation vector between the two poses, and θ is the angle between the two poses in radians. α and β can be set based on the characteristics of the robot and preference of the user. To make the robot gradually rotate while moving, β should be increased. We use $Dijkstra$'s algorithm to compute the shortest path between the current pose and the destination pose in the graph of recorded poses. Figure 5 shows an example of the plan for going to the bedroom from the kitchen.

The navigation module contains a stack of objectives, such as $idle$, $traverse$, $charge$. When this module is initiated an idle objective is pushed into the stack. When the robot receives a command, the planning module will compute a path of robot poses, and pushes each of them as a traverse objective to the stack. The planner pushes the poses into the stack in reverse order. The navigation model always loads the objective at the top of the stack and tries to accomplish it. In each frame the navigation modul performs a sequence of sense-think-act. It receives an image from the robot, localizes the robot in the environment, verifies if it has accomplished its current objective or not, and based on that decides the next action.

IV. Experimental Results

In this section, we describe our experimental results along with the details on the experimental setting.

We have conducted two experiments, one at home with markers placed in all rooms (in kitchen, dininig room, living room, bedroom, bathroom, etc.) and the other one in the corridors of our department, around a cubical in our lab. In both experiments we use ROVIO. Rovio is a small robot which has a camera on it, communicates via http and can receive movement commands. Rovio has three wheels and can rotate around itself, and move in different angles. Images transmitted from Rovio's camera are 640×480 and are not very high quality. In each of the experiments, we have placed markers onto walls and drawers. Here we describe each of these experiments and validate the results of EasySLAM for each one.

House: While controlling the Rovio with a laptop we drove it around a house.

We manually controlled the robot for 15 minutes, while it was capturing images every 1 second. A total of 964 images were collected. EasySLAM built the map of the environment. In Figure 4 we present the computed map on top of the house plan.

After the robot built the map, we put it in the navigation mode, and asked it to move to different parts of the house, and it passed all the tests successfully. In Figure 5 we have

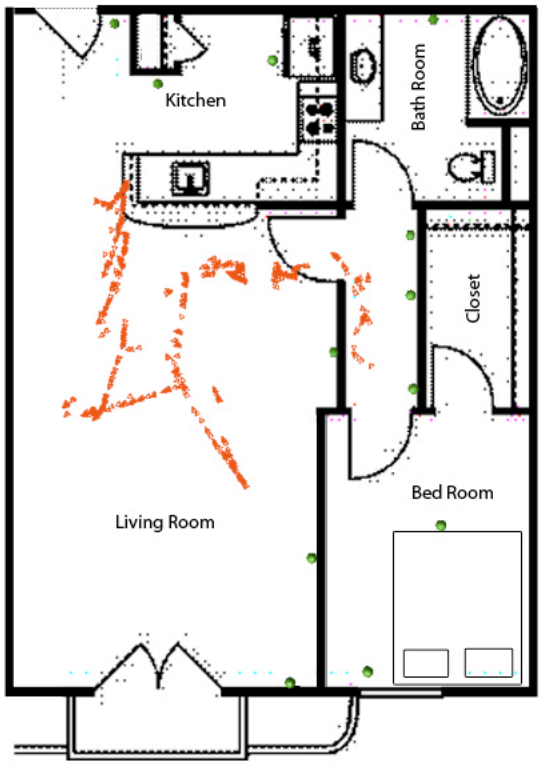


Fig. 4: The results of marker and robot pose computations for the house experiment are shown in the top view. Markers are shown as green spheres and robot poses are shown in red.

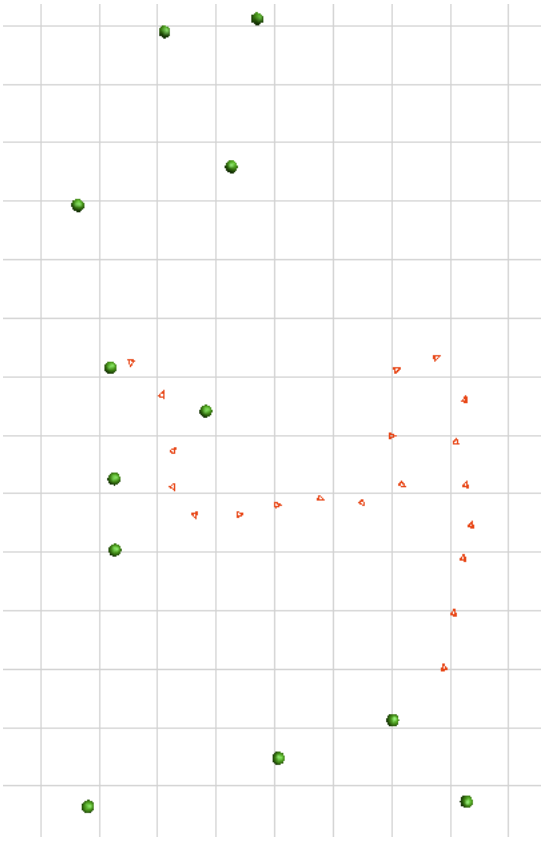


Fig. 5: The path planned by the robot to go to the bedroom from its current location (kitchen) is shown. As it is seen the robot never plans to go trhough the obstecles since it is using the history of poses for planning.

shown the path robot planned to go to the bedroom from the kitchen.

One might argue whether we need to optimize poses beyond the initial pose estimations. In Figure 6 we have presented the initial pose estimation results, which is not looking as good as the optimized version shown in Figure 4 and it won't be useful for reliable planning and navigation.

Lab: We drove our ROVIO robot in a corridor, around a cubical in our lab. We used a laptop to control the robot. We manually controlled the robot for 10 minutes, while it was capturing images every 1 second. A total of 684 images were collected. This experiment has two main properties: (1) we did not close the loop, to test if EasySLAM can build an accurate map without loop closing, and (2) we put many markers densly together to test if the robot can efficiently build the map with more markers that one will typically put in his house. The robot was able to build an accurate map of the environment without closing the loop as presented in Figure 7.

V. Conclusion

We introduced EasySLAM, a fast, robust, and easy to use framework for mapping and navigation in societal settings. One can introduce different markers at different locations in a house or a company, and robots can be controlled through the environment to create the map and navigate robustly which allows them to operate within the environment. It has been shown that EasySLAM can reliably localize itself, without using odometry and navigate reliably in real time and successfully avoid obstecles.

Since the data association is solved by unique marker appearances, we are able to solve the kidnapped robot situation. The other usage of using landmarks with IDs is that they can be easily associate with semantics (e.g. landmark 3 is associated with kitchen), however, we also plan to use the existing planar landmarks such as paintings, traffic signs, patterns on the walls, etc. within our framework in our future work.

References

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (SLAM): Part I the essential algorithms," *Robotics & Automation Magazine*, Jun 2006.

[2] T. Bailey and H. Durrant-Whyte, "Simultaneous localisation and mapping (SLAM): Part II state of the art," *Robotics & Automation Magazine*, Sep 2006.

[3] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT press, Cambridge, MA, 2005.

[4] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 29, no. 6, pp. 1052–1067, Jun 2007.

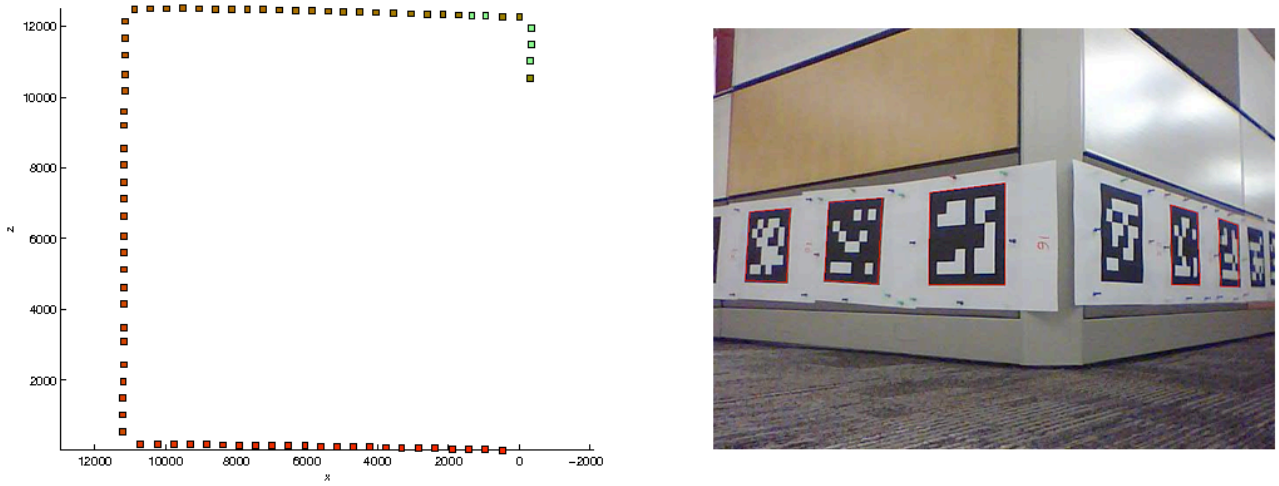


Fig. 7: EasySLAM builds an accurate map even without closing the loop. The image saved by the robot at frame 684 is shown on the right. Markers with a red square around them are the detected ones. The computed map is shown on the left. The units are in mm. The cubic area around which we put the markers is about $11m$ by $12m$. The markers are shown in different colors based on their ARToolkit id.

[5] K. Konolige and M. Agrawal, "FrameSLAM: From bundle adjustment to real-time visual mapping," *IEEE Trans. Robotics*, vol. 24, pp. 1066–1077, 2008.

[6] M. Cummins and P. Newman, "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance," *Intl. J. of Robotics Research*, vol. 27, no. 6, pp. 647–665, June 2008.

[7] D. Lowe, "Object recognition from local scale-invariant features," in *Intl. Conf. on Computer Vision (ICCV)*, 1999, pp. 1150–1157.

[8] J. Castellanos, J. Montiel, J. Neira, and J. Tardós, "The SPmap: A probabilistic framework for simultaneous localization and map building," *IEEE Trans. Robot. Automat.*, vol. 15, no. 5, pp. 948–953, 1999.

[9] J. Guivant and E. Nebot, "Optimization of the simultaneous localization and map building algorithm for real time implementation," *IEEE Trans. Robot. Automat.*, vol. 17, no. 3, pp. 242–257, June 2001.

[10] J. J. Leonard, R. Carpenter, and H. J. S. Feder, "Stochatic mapping using forward look sonar," *Robotica*, 2001.

[11] J. H. Kim and S. Sukkarieh, "Airborne simultaneous localisation and map building," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2003.

[12] M. Dissanayake, P. Newman, H. Durrant-Whyte, S. Clark, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Trans. Robot. Automat.*, vol. 17, no. 3, pp. 229–241, 2001.

[13] M. Deans and M. Hebert, "Experimental comparison of techniques for localization and mapping using a bearings only sensor," in *Intl. Sym. on Experimental Robotics (ISER)*, December 2000.

[14] S. Julier and J. Uhlmann, "A counter example to the theory of simultaneous localization and map building," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 4, 2001, pp. 4238–4243.

[15] F. Dellaert, "Square Root SAM: Simultaneous location and mapping via square root information smoothing," in *Robotics: Science and Systems (RSS)*, 2005.

[16] D. Wagner and D. Schmalsteig, "Artoolkitplus for pose tracking on mobile devices," in *Computer Vision Winter Workshop*, 2007.

[17] M. Fiala, "Artag, a fiducial marker system using digital techniques," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[18] J. Hayet, F. Lerasle, and M. Devy, "A visual landmark framework for indoor mobile robot navigation," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2002.

[19] J. B. Hayet, F. Lerasle, and M. Davy, "Visual landmark detection and recognition for mobile robot navigation," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[20] A. Watkins, J. Kehoe, and R. Lind, "Slam for flight through urban environments using dimensionality reduction," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2006.

[21] U. Frese, "Treemap: An $O(\log n)$ algorithm for indoor simultaneous localization and mapping," *Autonomous Robots*, vol. 21, no. 2, pp. 103–122, 2006.

[22] P. Zhang, E. E. Milios, and J. Gu, "Underwater robot localization using artificial visual landmarks," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, 2004.

[23] C. Berger and S. Lacroix, "Using planar facets for stereovision SLAM," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

[24] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *Intl. Conf. on Computer Vision (ICCV)*, 1999.

[25] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *Intl. J. of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, Dec 2006.

[26] A. Griewank, "On Automatic Differentiation," in *Mathematical Programming: Recent Developments and Applications*, M. Iri and K. Tanabe, Eds. Kluwer Academic Publishers, 1989, pp. 83–108.

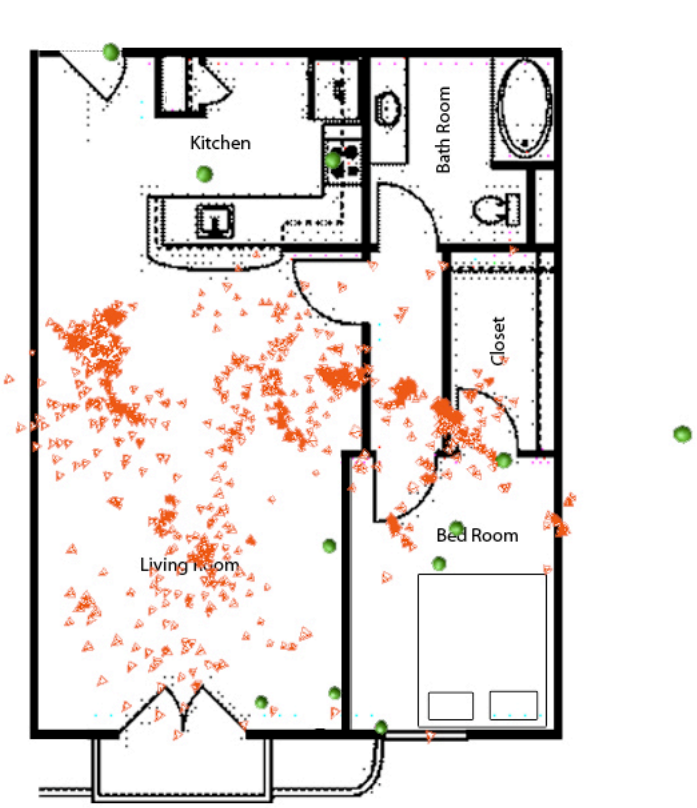


Fig. 6: The pre-optimized results of marker and robot pose for the house experiment are shown in the top view. The initial results are built by computing the relative poses between the robot and landmarks using homography in each frame. As it is seen pre-optimized results are not as accurate, and markers on the right are completely off the ground truth pose.