Trajectory Planning for Monocular SLAM based Exploration System

Sarthak Upadhyay
Robotics Research Center
IIIT Hyderabad, India

Ayush Dewan
Robotics Research Center
IIIT Hyderabad, India

Arun Kumar Singh
Robotics Research Center
IIIT Hyderabad, India

Madhava Krishna
Robotics Research Center
IIIT Hyderabad, India

ABSTRACT

In this paper, we propose a novel planning technique for monocular camera based Simultaneous Localization and Mapping(VSLAM). In VSLAM, the objective is to estimate the trajectory of camera and simultaneously identify 3D feature points and build a map, using camera as a depth sensor. Unlike a laser range finder based SLAM, VSLAM is known to be erroneous when camera motion includes an in-place rotation or feature displacement is large for successive frames. We propose a motion planning framework which combines motion primitives based planning and trajectory optimization approach to generate trajectories which exactly connects an initial and final state and also ensures that the change in camera's *field of view* between subsequent instances is less than some specified threshold. As a consequence of this motion planning framework we are able to automate SLAM and generate automated monocular SLAM maps of an indoor lab area. We also show when the robot follows the path of a generic planner, PTAM trajectory breaks more often than when it executes the path computed by the proposed planner. This performance improvement is further utilised to develop an autonomous vision based exploration system.

1. INTRODUCTION

In recent years, research on monocular camera based robotic system have spurred up, specially in area of aerial robotics [3], [15]. Since, very few Micro Aerial Vehicles(MAV) research platforms support heavy sensors like Laser Range Finder(LRF), camera remains a feasible and an inexpensive alternative. Mostly, camera has been primarily used as a sensor for body-velocity detection and as a sensor for state estimation. The latter technique generally uses map information for estimating pose of camera and is more relevant, considering the context of this paper.

AIR '15 July 02-04, 2015, Goa, India

DOI: http://dx.doi.org/10.1145/2783449.2783476.

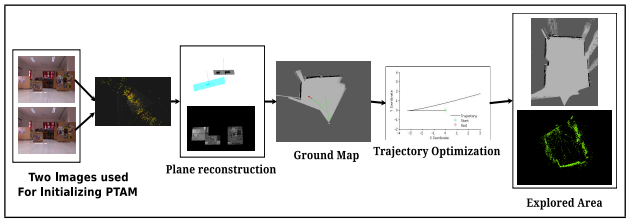

Figure 1: Steps involved in the proposed exploration system

In this paper, we primarily concentrate on using camera as a sensor for Simultaneous Localization and Mapping(SLAM). Monocular SLAM unlike range based mapping system is known to be less tractable, both in terms of accuracy of reconstruction and scale of map obtained. The main reasons include, (i) highly non-linear nature of projection operation,(ii), the entailment of good initial guess for effective convergence of non-linear optimization(Bundle Adjustment),(iii) presence of degenerate cases, such as highly planar scenes and in-place rotations, (iv) ambiguities in scale and in the decomposition of Essential Matrix to obtain camera pose estimation, (v) sparse nature of reconstruction.

Due to above reasons most monocular SLAM systems tend to show results over smaller scales[1],[14]. The best results for monocular SLAM have been obtained with hand held camera, where camera motion is unconstrained and repeated scanning of same area is feasible [7]. However, such motion is difficult to port onto non-holonomic systems like differential drive/ Ackerman steer robot. It is in this context, this paper describe a trajectory planning framework for automating monocular SLAM for such robots.

In this paper we propose a novel motion planning framework which connects exactly the initial and final state of the robot and ensures that change in camera's *field of view* (FOV) between subsequent instances is less than a specified threshold. We propose a hierarchical optimization framework which breaks a difficult non-linear optimization into a sequence of convex programmes with linear constraints and quadratic objective function. To ensure that the resulting trajectory lies within the known part of the map, we develop heuristics about the states which can be connected easily through the optimization and results in trajectories of shorter lengths. We use motion primitives based plan-

ning to guide the robot from the current state to a general vicinity of favourable states. We then used the trajectory optimization approach to make the final connection between the states. Although motion primitives based RRT [11] are themselves capable of producing trajectories satisfying differential constraints (like FOV constraints), their inability to converge exactly to the goal state is the main motivation behind using a mixture of motion primitives based planning and trajectory optimization techniques.

The main contribution of this paper is that, it is one of the first such efforts to automate a monocular SLAM framework for a non holonomic robot. Secondly it proposes a novel trajectory planning framework that provides for stable VSLAM estimates without breaks. For instance it increases reversals to keep features in view and avoid large rotational displacements thereby arresting possibilities of break down of PTAM trajectories. For once the trajectory breaks it is indeed difficult to stitch together disjoin trajectories and the map as they are typically at different scales. Using the method proposed in [4], sparse pointcloud from PTAM is densified and the dense information is used for standard frontier based exploration. Results of a stable exploration in large indoor lab($12m \times 8m$) has been shown, without any breaks in trajectory estimate. We also show when PTAM is automated with a generic planner trajectory breaks more often resulting in loss of both map and trajectory information. To the best of our knowledge, there have not been any such reporting of autonomous exploration for monocular SLAM.

2. SYSTEM OVERVIEW

This section briefly describes the different modules involved in the autonomous exploration system. PTAM acts like the backbone of the whole approach and supports different modules. Details regarding working of PTAM, and the relevant modifications made by us are presented in section 3.1. The sparse map created by PTAM acts as an input for dense reconstruction module, details of which are mentioned in section 5.1. Using the dense information a 2D ground map is made. This map is used for estimating frontier positions, and using the exploration strategy described in section 5.2 an appropriate frontier is determined. Using the proposed planning algorithm (section 4), a motion plan is generated to move the robot towards the frontier and both, camera pose and map are constantly updated as the robot moves along the trajectory. Flowchart in Figure 1 explains the above mentioned approach.

3. VSLAM FRAMEWORK

A SLAM system aims at building a coherent map of the environment around the robot. Map is defined in a global coordinate frame, and to observe the features of the environment, it is necessary to know the position of the robot in the same global frame, in other words, robot needs to be localized. In a monocular camera based SLAM, estimation of camera pose is typically in 6DOF, at a scale governed by the assigned distance to the initial two camera frames. Since motion of camera between capturing of those two images may be unknown, the map is built on an arbitrary scale.

3.1 PTAM

PTAM has turned out to be the most popular choice of open-source implementation of monocular SLAM. The current work can be seen as that which enables automating a PTAM based mapping for lab scale environment, from a traditional desktop setting.

Novelty of PTAM lies in the decision to run Tracking and Mapping operations on parallel threads. For tracking, an initial pose of camera is estimated using a simple motion model. The 3D map points are then projected back to the current camera frame and re-projected map points are compared with the features detected in the current frame and accordingly current pose is updated.

For refining map points and camera pose, Mapper thread uses Batch Optimization technique called Bundle Adjustment(BA). Bundle adjustment runs on subset of frames called Keyframes, rather than every single image. Even though BA is computationally expensive, threading of tracking and mapping operations, makes it feasible. Complete details regarding motion model, Tracking, Mapping and other important modules can be seen in [2].

In our experiments, we automatically initialize PTAM. Robot is moved along a straight line for small distance, and using experimental heuristics, two frames for stereo initialization are selected. Using the odometry sensor reading, movement of camera is estimated between two Keyframes and the map is initialized on metric scale. As mentioned in [2], complexity of BA is $O(N + M)^3$, where N is number of KeyFrames and M is number of points, and with sparse implementation(used in PTAM), complexity drops down to $O(N^3)$. The change in complexity with sparse implementation is commendable, but still, computation becomes expensive with increase in number of Keyframes.

Since the planning algorithm proposed by us, forbids any sharp change in field view of camera, the displacement of map points is gradual and smooth. This allows us to adopt a more conservative heuristic for adding Keyframe. If the robot is visiting an already explored area then, the frequency of adding Keyframe is reduced. Secondly, to manage the complexity for BA, we follow a approach similar to [3] and limit the number of Keyframes, to be considered for BA. We only choose n Keyframes near the current camera pose. As mentioned in [5], the effect of distant map points is negligible on optimization(BA), this modification does not impact the performance of PTAM.

3.2 Failure Case

As mentioned in our previous work [6], quality of map is highly dependent on performance of Tracker. The camera pose, estimated using motion model is updated by Tracker and the subsequent triangulation of 3D map points depends on pose of the camera. Hence, if quality of tracking drops, the error in camera pose estimation increases, which in turn causes erroneous triangulation of map points. Most of the trackers that work online, define a fix radius for searching image features. Thus, any sharp motion, like in-place rotation, could cause large displacement of features between consecutive frames and thereby hindering the quality of tracking. Similarly, fast movement of camera also has a negative impact on tracking. This movement causes blurring in images and due to the blur, signature of feature points are lost, which in turn makes them untrackable.

To attend to, the aforementioned failure cases and to generate a trajectory for camera, which improves the performance of PTAM, we propose our novel optimization formulation in next section.

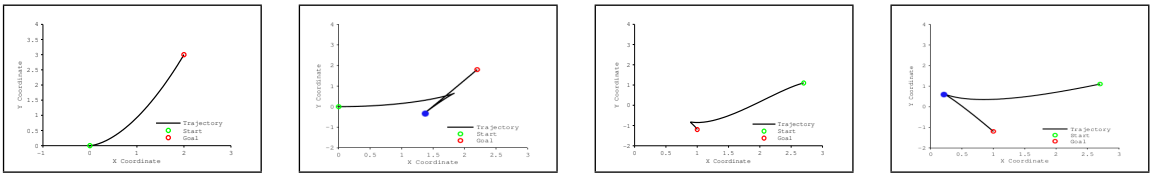

Figure 2: Figure shows snapshots of path plots((a)-(d)) map for unconstrained((a)&(c)) and constrained((b)&(d)) cases. Green, red and blue markers shows the initial, final and intermediate state. Green semi circles represent the already seen area, blue semi circle represent new points added to map during the motion and red semi circle represent the new points added at final state. Purple semi circle represent the point where trajectory breaks for unconstrained motion. Due to breaking, new points are not added to map for unconstrained motion.

4. MOTION PLANNING FOR IMPROVED VSLAM

In this section, we first describe the trajectory optimization framework which is the primary component of the proposed motion planning framework and then how it is combined with motion primitives based planning.

4.1 Trajectory Optimization

We start with the following unicycle model of the robot

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ u_1 = \dot{v} \\ u_2 = \dot{\theta} \end{cases} \tag{1}$$

where $v = \sqrt{\dot{x}^2 + \dot{y}^2}$ represents the linear velocity of the robot from the robot's local reference frame aligned with the longitudinal axis. θ is the heading of the robot. u_1 and u_2 are the control inputs. We will refer the triplet (x, y, θ) as states. System represented by (1) is differentially flat [12] which means that a subset of the states can be chosen as flat outputs and all other states and control variable can be expressed as an algebraic function of the flat outputs. We choose here x and θ as flat outputs and parametrize them as the following polynomial functions.

$$x(a_i, t) = \sum_{i=1}^{12} a_i t^i, \tan\theta = \zeta(b_i, t) = \sum_{i=1}^{5} b_i t^i \tag{2}$$

By (1) we have

$$\dot{y}(t) = \dot{x}\zeta \Rightarrow y(t) = \sum_{i=1}^{12} f_i(b_i, t)a_i \tag{3}$$

f_i are functions of only parameters b_i and t. In the above definition $t \in [t_o, t_f]$ is an arbitrary time interval. The objective is to compute such a_i and b_i which satisfies the following set of constraints

$$C = \begin{cases} x(t_o) = x_0, x(t_f) = x_f, \dot{x}(t_o) = 0, \dot{x}(t_f) = 0 \\ \zeta(t_o) = \zeta_0, \zeta(t_f) = \zeta_f, \dot{\zeta}(t_o) = 0, \dot{\zeta}(t_f) = 0 \\ y(t_o) = y_0, y(t_f) = y_f \end{cases} \tag{4}$$

$$-\phi \leq \frac{d\theta}{ds} \leq \phi \Rightarrow -\phi \leq \frac{\frac{d\theta}{dt}}{\frac{ds}{dt}} \leq \phi \tag{5}$$

Equalities presented in (4) arises out of boundary constraints and ensures that the initial and final states are exactly connected. Inequality (5) restricts the heading change of the robot along the trajectory arc length and is equivalent to putting constraints on the field of view. This equivalence arises out of the fact that for a non-holonomic robot with camera fixed to the robot's base, changes in field of view between subsequent frames has a one to one correspondence with changes in the robot's heading along the trajectory arc length. It should be noted that although (5) mathematically resembles a typical curvature constraint on the robot's path, the numerical value of ϕ is quite different from minimum curvature value that the robot can follow.

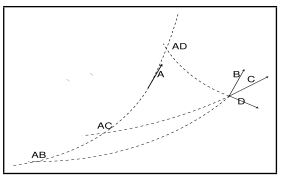

Figure 3: Heuristics for favourable states.

Using (3) constraint (5) can be further simplified in the following manner.

$$-\phi \leq \frac{\dot{\zeta}}{(1+\zeta^2)^{\frac{3}{2}}\dot{x}} \leq \phi \tag{6}$$

The coefficients a_i and b_i defining the trajectory can be obtained by using (4) and (6) as constraints in a typical parameter optimization setting. However, the last equality in (4) is quadratic in terms of a_i and b_i while (6) is highly non-linear which makes the parameter optimization difficult to solve. But it can be noted that if b_i are fixed (which in turn fixes ζ, $\dot{\zeta}$), the constraints (4) and (6) become linear. Hence we next propose a two level optimization structure where at the first level we obtain the parameters b_i and then they are used at the second level of optimization to solve (4) and (6).

4.2 Obtaining Parameters b_i

We start with generating n+2 grid points $(t_o, t_1, t_2....t_n, t_f)$ in the time interval $[t_o, t_f]$. The parametric functions ζ is then symbolically evaluated at these grid points and the resulting expressions are used in the following optimization problem

$$\min J = \sum_{i=0}^{n} (\zeta(t_{i+1}) - \zeta(t_i))^2 \tag{7}$$

subject to

$$C_\zeta = \{\ \zeta(t_o) = \zeta_0, \zeta(t_f) = \zeta_f, \dot{\zeta}(t_o) = 0, \dot{\zeta}(t_f) = 0 \tag{8}$$

The intuition behind objective function (7) is to minimize the heading change of the robot between consecutive grid points. But to create a convex quadratic objective function, we use (2) and instead minimize the tangent of the heading

change between consecutive grid points. The equality constraints (8) are linear in terms of parameters b_i and hence the optimization problem is a simple quadratic programming problem which can be solved efficiently. The parameters b_i thus obtained are used in the second level of optimization which is described next.

4.3 Obtaining Parameters a_i

To obtain parameters a_i, we frame the following optimization problem

$$\min J = \sum_{i=0}^{n} (y(t_{i+1}) - y(t_i))^2 + (x(t_{i+1}) - x(t_i))^2 \tag{9}$$

subject to

$$C_{xy} = \begin{cases} x(t_o) = x_0, x(t_f) = x_f, \dot{x}(t_o) = 0, \dot{x}(t_f) = 0 \\ y(t_o) = y_0, y(t_f) = y_f \end{cases} \tag{10}$$

$$C_{in} = \begin{cases} \dot{\zeta}(t_i) - (1 + \zeta^2(t_i))^{\frac{3}{2}} \dot{x}(t_i)\phi \leq 0 \\ -\dot{\zeta}(t_i) - (1 + \zeta^2(t_i))^{\frac{3}{2}} \dot{x}(t_i)\phi \leq 0, \forall i = 1, 2, 3..n \end{cases} \tag{11}$$

The objective function (9) is convex quadratic and seeks to minimize the arc length of the trajectory. The equality constraints (10) and the inequality constraints (11) are linear in terms of parameters a_i. Hence the optimization again is a simple quadratic programming problem. A careful review of the trajectory optimization would reveal that it does not include any linear and angular velocity or acceleration bound constraints. This is consciously done since these constraints are highly non-linear and non-convex. These constraints are enforced on the resulting trajectory through the concept of time scaling [13].

4.4 Integrating Trajectory Optimization with Motion Primitives

In cases when some potions of the trajectory resulting from the optimization above falls outside the map, we resort to motion primitives based RRT [11] to guide the robot from the current state to the vicinity of favourable states. Favourable states refer to those which can be easily connected through the trajectory optimization and results in trajectories of shorter lengths. To understand the heuristics behind characterizing favourable states, consider figure 3 which shows an initial state A and multiple final states B, C and D. The final states have the same position and differ only in their orientation. A circle of radius R is drawn such that the state A is tangential to the circle. Similarly circle of same radius is drawn with states B, C and D as tangents. Intersection of these circles are shown in the figure as AB, AC and AD respectively. The approximate arc length distance of the point of intersection from state A can be seen to be in order $AB > AC > AD$. Since constraint (6) can be seen as a constraint on arc curvature, we characterize favourable state as one which has a larger heading difference with the final state.

Since, in place rotation is not permissible, it is not possible to change the heading of initial state, while maintaining the position. Hence we sample a new state (x, y, θ) and use motion primitives based planning to guide the robot from the current state to the newly sampled state. It is enough that the robot is guided to the vicinity of the sampled state. The trajectory optimization is then used to make the final connection to the goal state. The sampled intermediate state serves two purpose. First it increases the heading difference with the final state. Moreover, it is further inside the map thereby resulting in a trajectory which lies within the map.

5. VISION BASED EXPLORATION

As an application to the proposed planning algorithm, we present an autonomous vision based exploration technique. The objective of an exploration mission is to explore an unknown area and gather maximum information about the environment. One of the most commonly used strategy for exploration is known as the "Next Best View" approach. Using the current environment information(a map), a set of plausible goal positions, known as frontiers, are identified. These frontiers are quantified using a cost function and among them a suitable frontier is chosen according to an objective function.

Most of the existing exploration algorithms have primarily employed depth sensors like laser or more recently Kinect for interacting with surrounding environment. Output from both of these sensors can be used for making a dense and information rich map of the environment. The challenge arises in case of a monocular camera, where the map information remains sparse and determining a frontier position from such sparse map becomes non-trivial.

To densify the sparse pointcloud, and make it legible for exploration we use algorithm proposed in [4].

5.1 Dense Reconstruction

Dense reconstruction using a monocular camera has been an active research problem for quite a few years. Davison et al proposed Dense Tracking and Mapping Algorithm in [7]. Building over a sparse pointcloud information, using optical flow they were able to achieve remarkable results in real time. More recently, a Signed Distance Function based function 3D dense reconstruction algorithm was proposed in [8]. Even though, both these methods present commendable results, they heavily rely on GPUs for processing. Therefore, using them for system which relies on on-board processing is currently not feasible.

In this paper we follow the reconstruction algorithm, proposed by Argiles etal in [4]. Using the algorithm mentioned, we detect and reconstruct dominating planes from sparse PTAM map. For detecting planes in the scene we use Jlinkage. Jlinkage uses random sampling for estimating multiple plane hypothesis and then clusters the hypothesis and extracts planes from it. The plane detected using Jlinkage is divided into smaller grids, and each grid is checked for photo-consistency. Initially grids, with 3D map points are chosen for checking photo-consistency and then the same check is done for there neighbours. This way of searching allows to retrieve texture less plane surface.

Figure 4 shows reconstruction of planes for scene in Figure 4(a). Figure 4(b) shows the dominating plane estimated by Jlinkage using PTAM pointcloud. Using photo-consistency matching, the gap between two planes is detected. Complete details regarding photo-consistency matching and the complete implementation can be seen in [4].

5.2 Exploration

The dense pointcloud is sampled at a particular height and is used as a virtual laser scan for creating ground maps. The resultant map can be seen in Figure 5.2. After building the ground map, we continue with frontier based exploration.

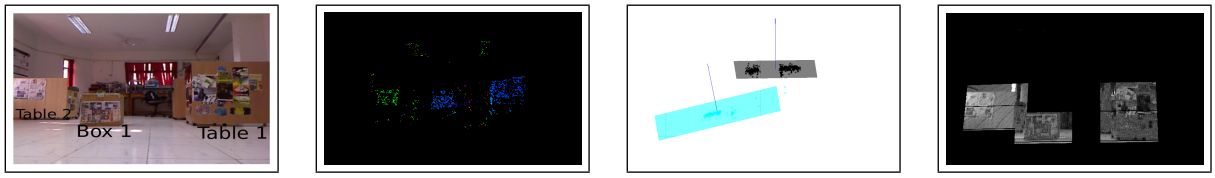

Figure 4: Various steps involved in reconstruction of the scene in (a). Using Jlinkage dominated planes are estimated(c), for sparse pointcloud of the scene, shown in (b). Photoconsistency matching helps in detecting gap between Box1 and Table1 and the final 3D reconstruction is shown in (d).

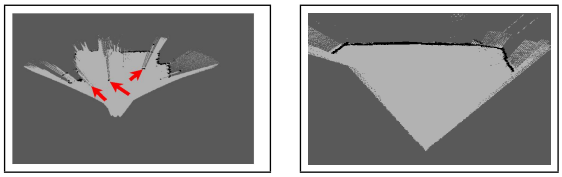

Figure 5: Ground Map created (a) using sparse PTAM pointcloud. (b) After dense reconstruction

Dataset	U_1		U_2	
	Constrained	Unconstrained	Constrained	Unconstrained
Dataset 1	0.9053	0.210	242.994	68.34
Dataset 2	0.7053	0.3713	116.0683	41.1478

Table 1: Comparison between Constrained and Unconstrained Trajectories

Each frontier is quantified with following cost function:

$$U_i = \frac{v_i}{d_i} \tag{12}$$

Here, v_i is the approximated gain of information for frontier i and d_i is the distance robot has to travel to reach the frontier. The objective is to choose a frontier with maximum utility U_i. As shown in our previous work [10], this choice of cost function provides superior results in comparisons to other proposed methods. After choosing a frontier of maximum utility, using the proposed planning technique, robot moves to the goal and continues the exploration.

6. HARDWARE IMPLEMENTATION AND RESULTS

For our experiments, we use Kinect mounted on a iCreate TurtleBot. Kinect is used only as a monocular camera and not as depth sensor. The whole algorithm is processed on a laptop connected to robot. Laptop runs on a 2.3 GHz Quadcore i7 processor. The complete exploration system was implemented using ROS(Robotics Operating System).

As shown in the Figure 1, the first step of the exploration is PTAM. We initialize the PTAM using the technique mentioned in section 3.1. In the next step, dominant planes are estimated from initial pointcloud and using the procedure mentioned in section 5.1 a ground map is built(Figure 5.b).

For reconstruction we added another thread to the existing PTAM framework. The motivation for dense reconstruction is to improve the map for aiding accurate navigation of robot. Figure 5 shows the comparison between the ground map created using sparse pointcloud and the map created after dense reconstruction. In sparse map, due to sparse cloud information, obstacle boundaries have gaps, whereas obstacle boundaries are coherently mapped after densifying the sparse information. Secondly, the error in map due to PTAM map points belonging to floor is also removed. This improvement in map verifies our motivation to use dense reconstruction for navigation.

Table 1 shows comparisons between unconstrained and constrained trajectories for cases shown in Figure 2. Comparisons are done on two metrics, (i)average number features tracked in every frame (U_1) and (ii) average number of frames in which every point is tracked (U_2). For every frame, PTAM estimates total number of map points to be searched in current frame. U_1 is ratio of points tracked to the total points to be searched, averaged over total number of frames. U_2 is the average number of frames in which every point is tracked. In both cases, a high value will point towards efficient tracking. In both cases our proposed approached outperformed the unconstrained method by large margin. In Table 1, Dataset 1 refers to cases in Figure 3.2 and 3.2, and dataset 2 refers to cases in Figure 3.2 and 3.2.

Figure 6 shows comparisons between planner trajectory (blue) and PTAM trajectory(green) for both Dataset 1 and 2. In case of constrained trajectory, PTAM trajectory closely follows the planner trajectory, whereas in case of unconstrained, planner trajectory is closely followed until PTAM trajectory breaks(purple marker) after which both trajectory and map information is lost vindicating our current formulation. This results vindicates the success of our formulation.

Figure 7 shows the explored area at different time frames. Blue points belongs to the patches or grid which satisfied the photo-consistency matching (discussed in Section 5.1) and the trajectory of robot is shown in green color for both cases. We would like to emphasize, that these maps are built using a monocular camera and we do not expect our results to be on the same level as the maps built using generic depth sensors. Exploration was done for an indoor lab area of dimension $12m \times 8m$. The exploration shown was completely autonomous and no human intervention was required at any step. As per our knowledge, results of this extent, for a monocular camera based exploration system has not been shown before.

In this paper we shown results to verify the superior performance of our planning technique, in comparison to an existing framework(Figure 2,6 & Table 1). We have also presented a fully autonomous frontier based exploration system which verifies the importance of such a planning framework in VSLAM and shows how PTAM can be utilized for mapping larger environments.

7. CONCLUSION

In this paper we proposed a novel trajectory planning

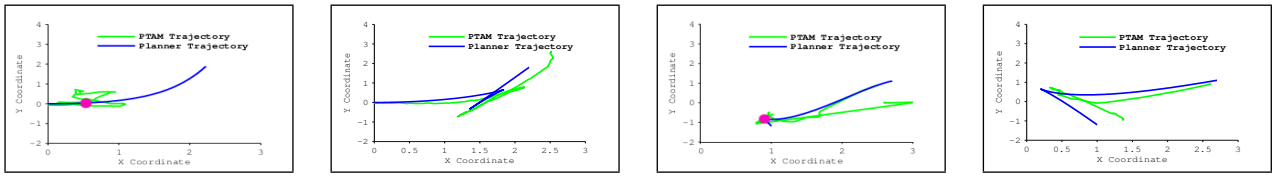

Figure 6: Comparison between planner trajectory(blue) and PTAM trajectory(green). In case of constrained trajectories((b)&(d)) PTAM trajectory follows planner trajectory closely. In case of unconstrained motion((a)&(c)) PTAM trajectory is close to planner trajectory, until it breaks(Purple marker)

.

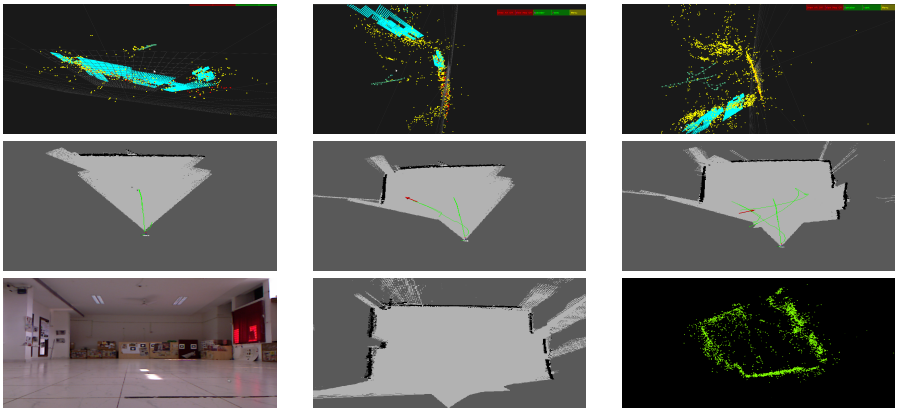

Figure 7: Figure (a)-(c) and (d)-(f), show PTAM map and ground map at different stages of exploration of scene in (g).(h) Complete ground map,(i) complete PTAM map.

framework for automating Monocular SLAM. Monocular SLAM systems have typically known to work when a small area is repetitively scanned with a freely moving camera, which are difficult to port onto non holonomic robots. Through an apt combination of motion primitives and constrained trajectory optimization techniques we show stable reconstruction of indoor lab scale environments. We show through the proposed framework feature tracks are obtained for longer sequences as well as more features are seen in every image when compared with a generic planner. This directly contributes to stable trajectories when compared with a generic planner where trajectory and maps are lost more frequently. In future we would like to show exploration for a larger area.

8. REFERENCES

[1] A. Davison, I. Reid, N. D. Molton, and O. Stasse.“ MonoSLAM: Real- time single camera SLAM.” *IEEE Trans. PAMI*, 2007.

[2] Klein, G., Murray, D.: “Parallel tracking and mapping for small ar workspaces”. *In: Proc. ISMAR'07*.Nara, Japan (2007)

[3] S Weiss, M W. Achtelik, S Lynen, M Chli and R Siegwart. “Real-time Onboard Visual-Inertial State Estimation and Self-Calibration of MAVs in Unknown Environments”,ICRA,2012

[4] A Argiles, J Civera and L Montesano “Dense Multi-Planar Scene Estimation from a Sparse Set of Images”, IROS,2011.

[5] G. Sibley, C. Mei, I. Reid, and P. Newman “Adaptive relative bundle adjustment”,RSS,pages 976-982,2009.

[6] Gavshinde, Laxit, Arun K. Singh, and K. Madhava Krishna. ”Trajectory planning for monocular SLAM systems.” Control Applications (CCA), 2013 IEEE International Conference on. IEEE, 2013.

[7] R A. Newcombe, S J Lovegrove and A J. Davison “DTAM: Dense Tracking and Mapping in Real-Time” ICCV, 2011.

[8] Bylow, Erik, J Sturm, C Kerl, F Kahl, and D Cremers. “Real-time camera tracking and 3d reconstruction using signed distance functions.” RSS, 2013.

[9] S. Weiss, M. Achtelik, L. Kneip, D. Scaramuzza, R. Siegwart, “Intuitive 3D Maps for MAV Terrain Exploration and Obstacle Avoidance”, *Journal of Intelligent and Robotic Systems*, Vol. 61, 2011,

[10] A.Dewan, A.Mahendran, N.Soni and M.Krishna , “Heterogeneous UGV-MAV Exploration Using Integer Programming” IROS,2013.

[11] LaValle, Steven M., and James J. Kuffner. ”Randomized kinodynamic planning.” The International Journal of Robotics Research 20.5 (2001): 378-400.

[12] Tang, Chin Pei, et al. ”Differential-flatness-based planning and control of a wheeled mobile manipulator Theory and experiment.” Mechatronics, IEEE/ASME Transactions on 16.4 (2011): 768-773.

[13] Hollerbach, John M. ”Dynamic scaling of manipulator trajectories.” American Control Conference, 1983. IEEE, 1983.

[14] AP Gee, D Chekhlov, A Calway, W Mayol-Cuevas “Discovering higher level structure in visual SLAM”, *Robotics, IEEE Transactions on 24 (5), 980-990*

[15] J Engel, J Sturm, D Cremers, “Camera-Based Navigation of a Low-Cost Quadrocopter”,IROS,2012.