2013 IEEE International Conference on Control Applications (CCA)
Part of 2013 IEEE Multi-Conference on Systems and Control
Hyderabad, India, August 28-30, 2013

Trajectory Planning for Monocular SLAM Systems

Laxit Gavshinde, Arun K Singh and K Madhava Krishna

Abstract— **This paper proposes a novel method of integrating planning with Monocular Simultaneous Localization and Mapping (SLAM) systems. Monocular SLAM, typically referred to as VSLAM systems in literature consists of recovering trajectory estimates of the camera and stationary world features from a single moving camera. Such VSLAM systems are significantly more difficult than SLAM performed with depth sensors, such as using an accurate Laser Range Finder (LRF). When the camera motion is subject to steep changes in orientation, tracked features over the previous instances are lost, making VSLAM estimates highly unreliable, erroneous that cannot be recovered. Most often a complete breakdown occurs, which entails a new sequence of images to be captured from a fresh camera trajectory. Herein we propose an optimization based path planning formulation for such VSLAM systems that reduces occurence of such errors through paths that are not subject to high orientation changes. Further we plan a velocity profile over the path that prevents features from getting significantly displaced over successive images, often considered a critical criteria for robust feature tracking. The velocity profile is computed using the novel concept of non linear time scaling proposed in our earlier work. The VSLAM system is also sufficiently innovated to provide for dense mapping over planar segments. The efficacy of the formulation is verified over real experiments on a camera mounted robot.**

I. Introduction

Simultaneous Localization and Mapping (SLAM) is the problem of simultaneously estimating the state of the robot and map as the robot moves around a terrain [1]. Early SLAM systems used range sensors such as Laser Range Finders (LRF) [5]. However in recent years single camera SLAM systems have become very popular [8][3]. Since monocular camera is a projective sensor, reconstruction of the map in terms of its 3D coordinates from such a moving camera is considered inherently more difficult than SLAM using range sensors. Most SLAM systems involve moving the robot by teleoperation or along a predetermined trajectory. An automated SLAM system where the robot decides its next location to move have been sparse. Called SPLAM (Simultaneous Planning Localization and Mapping), such systems typically plan trajectories for the robot such as the overall uncertainty in the estimates of the robot and map features are minimized [6]. However all such SPLAM systems deal with depth sensors wherein specific problems posed by projective sensors such as a monocular camera have not been considered. There has not been a method so far based on our survey that elaborates on how to compute paths and trajectories that aid VSLAM systems.

In this paper we present a novel path and trajectory planning methodology that enables automation of VSLAM. Key to this is the identification that most VSLAM systems tend to break down when features get displaced by large amounts between successive images or when there is a significant camera rotation between successive images. Over various practical experiments we have found that VSLAM estimates tend to be considerably erroneous when camera rotation is just more than nominal between successive views. Of equal importance is the need to observe the structures of the environment from viewpoints that are close enough to those structures. Such requirements are not imminently present with depth based SLAM systems. A LRF can accurately get ranges to structures/objects as far as 30m away. The planning formulation takes into account the above observations. It chooses viewpoints close enough to the structures in the environment and plans trajectories that approach them without appreciable changes in camera orientation. Further when moving from one viewpoint to another additional waypoints are generated to facilitate reversal along the path when conventional maneuvers do not satisfy the orientation constraints. Finally we invoke non-linear time scaling techniques, first introduced in [7] to embed velocity profiles onto the planned paths. The velocity profile provides for higher velocities when the camera is further away from objects while reduces as the camera approaches the object. This follows from well known principles that the disparity changes of features on an object tend to change slower when camera if further away and vice versa.

Experimental results portray the advantages of the proposed formulation. VSLAM estimates of camera trajectory are found to be closer to ground truth when integrated with the proposed planner vis-a-vis VSLAM estimates obtained from a camera executing the path of any state of the art planner [4]. We also show statistics of feature tracks that positively influence the VSLAM estimates when the camera executes trajectory obtained from the proposed planner.

The paper is organized as follows. In section II we detail our VSLAM framework. In section III the planning formulation for VSLAM is elaborated. Simulation and experimental results are presented in section V and the paper concludes with section VII.

II. VSLAM Framework

In this section we describe VSLAM framework which coupled with path planning forms the proposed system. SLAM system aims at creating coherent maps of the areas that robot move through, in a single coordinate frame generally referred as the world frame. Monocular SLAM, which we have used, poses the most difficulty as estimation for robot location is in 7DOF; three for orientation, three for pose and scale of

All authors are with Robotics Research Lab, IIIT Hyderabad, India

the map. Motion is estimated at an arbitrary scale by the algorithm, incorrect estimations of pose lead to disjointed map, our path planner proposes a solution to improve pose estimation.

A. PTAM

Our Monocular SLAM system closely follows PTAM [3] developed by klein *et al*, which is one of the most widely used monocular 3D reconstruction system. We present a very short overview of the system. VSLAM models map points as $\mathbf{p} \in \mathbb{R}^4$ and camera pose as $\mathbf{A} \in \mathbf{SE}(3)$, whose corresponding Transformation matrix is denoted as $\mathbf{T(A)} \in \mathbb{R}^{4\times4}$.

1) Tracking: Tracking features in acquired frames is an essential part of the VSLAM system. Once an initial map is built tracking is performed using resection. To perform tracking PTAM estimates a prior pose for the current frame using a motion model. Points P_W are projected using a calibrated camera projection model CamProj(.)on to the image plane to get a prior pose measurement $\hat{p} \in \mathbb{R}^2$.

$$\hat{p} = CamProj(E_{CW}P_W) \tag{1}$$

Next, An 8×8 pixel patch search template is created from the source image. In the current frame this patch is compared to all the salient locations detected using FAST corners within a fixed radius from the predicted location. Zero-mean SSD is the metric denoting the degree of match. If the score for best match is below a specified threshold than the point P_W is considered as found in the current frame.

2) Mapping: PTAM saves information only for keyframes rather than frames, this is primarily done to reduce amount of data to be operated on, which enables real-time execution of slam. Two frames selected by user during map initialization are selected as the first two keyframes. Subsequent frames are added as keyframes if following conditions are satisfied:

- Tracking quality is good.
- Last keyframe was added n keyframes ago. For our case it's n=5.
- Camera has moved more than a minimum distance. This minimum distance is brought into scale by dividing it with mean depth of the map.

New points are added to the map when a new keyframe is added to map. To ensure quality tracking, subsequently good pose estimation, some of the salient features are triangulated and added to the map. Following conditions are must be satisfied for the point to be added:

- Non-maximal suppression and thresholding based on Shi-Tomasi.
- Points near already existing points are discarded.
- Point must lie on the epipolar line, which is computed using the previous Keyframe.
- Displacement of point on epipolar line should be more than a max threshold th_{max} and less than a min threshold th_{min}. These thresholds are computed using maximum and minimum depth in the map.
- Comparison of the best zero-mean SSD should be below a specified threshold.

B. Faliure Cases

Although PTAM says parallel tracking and mapping, VSLAM as a system has a serial pipeline which is dependent on previous step, which starts with input of image sequences and is as follows:

- Input: Sequence of images.
- Track features over images.
- Estimate pose for the images.
- Triangulate and create Map.
- Optimize using Bundle Adjustment.

We perform tracking to generate 2D feature correspondences. Quality of those correspondences determines quality of pose estimation. Quality of subsequent steps of triangulation and optimization are dependent on quality of both pose and tracking. It's safe to say quality of tracking has a major impact on quality of map created. VSLAM systems are preferred when operating online, which makes it an incremental system. That is, if an error is caused at any stage it will grow with time and eventually may cause the system to break.

One of the cases which results in low tracking quality is fast movement of camera. Almost all trackers which work in real time have an assumption of maximum search radius around the source point. High speed of camera can cause the points to go out of the specified search radius. High speed may also cause blurring to occur, in which case the signature of the point is lost and hence becomes untrackable. Steep transition in camera display the aforementioned behaviour when velocity is quite high compared to frame rate, but drop in tracking quality is more sensitive to steep rotation.

In case of steep rotation movement of points in the image is quite high, but compared to it baseline created is quite low. An in-place rotation is a degenerate case which makes triangulation impossible, similarly shorter baseline makes triangulation more erroneous. Steep rotation with short baseline will also result in closer values of th_{max} and th_{min} (section II-A.2), increasing the chances of track being missed. Fig 1(a) shows a state where trajectory was smooth and ample tracks were available. Fig 1(b) shows the state after robot has started to make a sharp turn, features are not being added to which leads to wrong pose estimation.

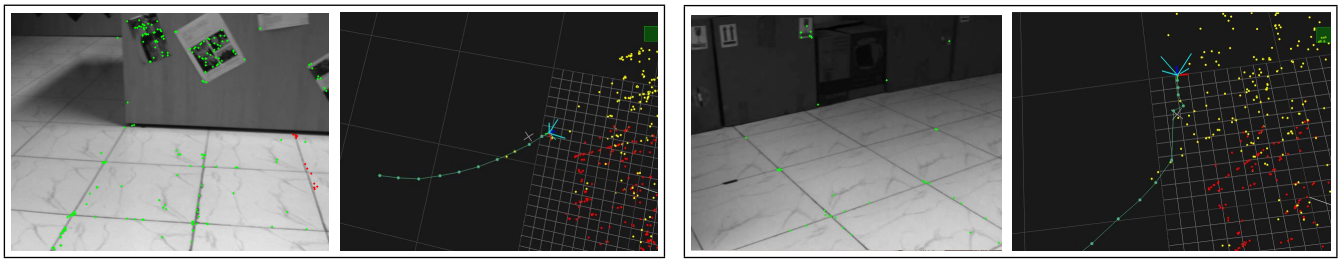
In the next section we propose framework to generate trajectories that ensure above mentioned cases do not occur, resulting in good tracking, hence quality map.

III. Path Planning Framework

As stated earlier the objective of the trajectory planning framework is to generate trajectories for non-holonomic robots between a given start and a goal location, such that robot's heading change along the path remains within a particular threshold. The robot is approximated as an unicycle represented by the following equations

$$\left\{ \begin{array}{l} \dot{x} = v\cos\theta, \dot{y} = v\sin\theta, u_1 = v, u_2 = \dot{\theta} \end{array} \right. \tag{2}$$

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



(a) (b)

Fig. 1. Images on the left in the boxes contain the actual scene with features marked with red and green color, Images on the right contain the trajectory at those instant. First box images are of a case where trajectory is smooth and the second row where trajectory brakes after rotation.

longitudinal axis. θ is the heading of the robot. u_1 and u_2 are the control inputs. System represented by (2) is differentially flat [[9]] 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.

$$x(a_i, q) = \sum_{i=1}^{12} a_i q^i, \tan\theta = \zeta(b_i q) = \sum_{i=1}^{5} b_i q^i \tag{3}$$

By (2) we have

$$\dot{y}(q) = \dot{x}(q)\zeta(q) \Rightarrow y(q) = \sum_{i=1}^{12} f_i(b_i, q)a_i \tag{4}$$

f_i are functions of only parameters b_i and q.In the above definition $q \in [q_o, q_f]$ is just arbitrary parameter describing the evolution of the path. With the above parametrization, the path planning framework can be framed as the following optimization problem.

$$L = \min\{(x(q_f) - x_f)^2 + (y(q_f) - y_f)^2\} \tag{5}$$

subject to the following equality constraints pertaining to initial boundary conditions

$$\begin{cases} x(q_0) = x_0, y(q_0) = y_0, \frac{dx}{dq}(q_0) = \dot{x}_i \\ \zeta(q_0) = \tan\theta_0, \frac{d\theta}{dq}(q_0) = \dot{\alpha}_i \\ \zeta(q_f) = \tan\theta_f \end{cases} \tag{6}$$

The inequality constraints limiting the heading change of robot along the path can be put as

$$-\phi \leq \frac{d\theta}{ds} \leq \phi \Rightarrow \dot{\theta}^2(q) - \phi^2\sqrt{\dot{x}^2(q) + \dot{y}^2(q)} \leq 0 \tag{7}$$

ϕ is the specified threshold and can be thought as physically similar to curvature of the path. The objective function (5) is a function of only the final state of the robot and its minimization signifies that the final state of the trajectory should be as close as possible to the desired goal location. The non-linear optimization described by (5)-(7) is in general a difficult problem to solve because of two reasons. Firstly, the nature of the inequality constraints (7) which involves the independent variable p as well. To overcome this bottleneck remove the dependency of the variable p, constraint (7) is evaluated symbolically at $n+2$ pre-specified points, $(q_o, q_1, q_2....q_n, q_f)$ resulting in $n+2$ algebraic constraints as a function of only the variables a_i,b_i. As

Secondly the initial guess for the non-linear optimization is very critical since it dictates the convergence and performance of the optimization. To get a fast estimate of a good initial guess, we propose a simpler approximate problem to the above optimization problem which consists of solving a quadratic programming problem followed by solving a set of linear equations. This is described in detail in the subsequent sub-sections.

A. Initial Guess for Non-Linear Optimization

To get an initial guess for the variables b_i the following quadratic programming problem is set up.

$$L_1 = \min \mu^2 \tag{8}$$

The equality constraints of this quadratic programme is given by

$$\zeta(q_0) = \theta_0, \frac{d\zeta}{dq}(q_0) = \dot{\zeta}_i, \zeta(q_f) = \zeta_f + \mu \tag{9}$$

There are $n+2$ linear inequality constraints given by

$$\begin{cases} -\tan(\phi) \leq \zeta(q_1) - \zeta(q_0) \leq \tan\phi \\ -\tan(\phi) \leq \zeta(q_2) - \zeta(q_1) \leq \tan\phi \\ ..... \\ -\tan(\phi) \leq \zeta(q_f) - \zeta(q_n) \leq \tan\phi \end{cases} \tag{10}$$

The solution for the quadratic programme (8)-(10) is used as the initial guess for the parameters b_i in the non-linear optimization described earlier in this section. Using the initial guess solution for b_i just obtained, initial guess for a_i are obtained as

$$[a_1, a_2, a_3...a_{12}]^T = A^+[x_0, y_0, x_f, y_f, \dot{x}_i]^T \tag{11}$$

A^+ represents the pseudo inverse of the matrix A. Matrix A is the coefficient matrix obtained by collecting the coefficient of the variables a_i in the position level boundary constraints.

Figure(2(a)) shows an output from the proposed optimization based path planning. framework. As can be seen from

the figure, the paths respecting constraint (7) is significantly longer than the unconstrained paths. However as it will be shown in section V that the quality of feature tracking is significantly better on the constrained path. Figure 2(b) shows the satisfaction of constraint (7). As it can be seen from the figure, constraint satisfaction is good at almost all the place except a marginal violation at the middle of the path.

In the next section we describe how velocity profiles are fitted to the paths just obtained.

IV. Velocity Profiles for the Heading Constrained Paths

In the previous section we obtained heading constrained paths in the form of parametric functions, $x(a_i, p)$,$y(a_i, b_i, p)$. To compute velocity profiles for these paths, the path variable p needs to be transformed to the time domain i.e we need relations in the following form

$$t = h(q) \tag{12}$$

We propose following two functional relations between the time variable t and the path variable q for the construction of the function $h(.)$

$$\frac{dq}{dt} = k_1 e^{-k_2 q} \tag{13}$$

A transformation in the independent variable q to time t in the functions $x(.)$ and $y(.)$ does not change the path of the function but brings the following change in it's derivative

$$[\dot{x}(t), \dot{y}(t), \dot{\theta}(t)]^T = \frac{dq}{dt}[\dot{x}(q), \dot{y}(q), \dot{\theta}(q)]^T \tag{14}$$

(14) shows that the velocities along the path are related to the path derivatives through $\frac{dq}{dt}$, called the scaling function. $\frac{dq}{dt}$ is constructed using the form presented in (13) to satisfy the following two objectives.

• The linear and angular velocities along the paths should not exceed the maximum permissible velocity of the robot.

• From the point of view of VSLAM The robot should approach/leave the obstacle with low velocity. This is because when the robot is closer to the obstacle, the features gets displaced faster from the subsequent image frames. This disparity of features is less rapid when the robot is away from the obstacle and hence during these portions of the paths, the robot can move faster.

The permissible value for the velocity control input v_{max} is kept at $0.4m/s$. Further when the robot is closer to the obstacle the value of the v_{max} is further reduced to $0.2m/s$. As stated, the velocity of the robot along the path is derived from the path derivatives according to (14). Hence it is necessary to obtain first the minimum scale factor s_{min} values by which the path derivatives needs to be scaled to satisfy the above two conditions on velocities. Figure 3(a) shows the magnitude of the path derivatives $v(q)$ and $\dot{\theta}(q)$. Figure 3(b) shows the scale factor s_{min} values by which the derivatives needs to be scaled. The scaling function $\frac{dq}{dt}$ should be so constructed that the following conditions is satisfied throughout the path.

$$\frac{dq}{dt} \leq s_{min}, \Rightarrow k_1 e^{-k_2 q} \leq s_{min} \tag{15}$$

The above equation suggests that the scaling function profile should always be below the s_{min} variation profile. From (15), the values of k_1 and k_2 are computed by considering smaller sub-intervals of the path variable q. For example consider a subinterval of $q \in (q_1, q_2)$. Let the value of s_{min} at q_1 and q_2 be s_1 and s_2. So by using (15), we have

$$k_2 = \frac{ln\frac{s1}{s2}}{(q_2 - q_1)}, k_1 = s_1 e^{k_2 q_1} \tag{16}$$

Such values of s_1 and s_2 are chosen such that the scaling function profile is completely below the the minimum scale factor (s_{min}) profile. For example in the interval $q \in (1, 1.2)$, $s_2 = 0.035$. Hence value of $s_1 = 0.13$ was chosen. The resulting velocity profiles are shown in figure 3(c). As can be seen from the figure that the time taken to traverse the trajectory comes out to be 8.85 s.

TABLE I

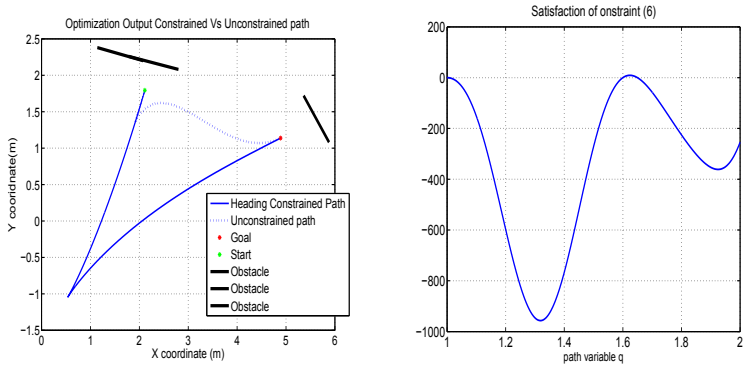
Comparison between two set of paths, C stands for constrained and U for unconstrained. Q_i's are statistics explained in section V

Dataset	Q_1		Q_2		Q_3	
	C	U	C	U	C	U
Set 1	0.708	0.473	350.1	324.7	420.9	411.1
Set 2	0.743	0.416	391.4	267.4	478.2	382.2

V. Results

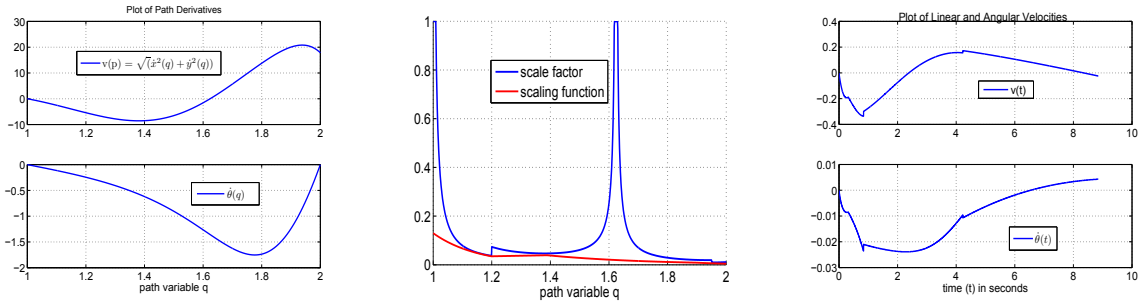
We compute some statistics to demonstrate features are being tracked better when path generated by our planner is followed. Table I shows three statistics for two sets of path to compare constrained and unconstrained path. First column(Q_1) is average number of points tracked for each frame. If Q_1 is high that means very few features did not satisfy the criteria mentioned in Section II-A.1. It's quite evident that features are better tracked by our planner. Second column Q_2 is average number of frames each feature was seen in. A higher value denotes a feature was tracked for longer time, which shows higher quality of the correspondences. Third column Q_3 shows avg. distance moved by a points in the complete run. This measure is similar to second column but this is independent of frame rate at which the data was captured.

We have also showed Figures which demonstrate smooth trajectories for constrained paths, where as unconstrained path breaks at the point where high rotation is involved. Fig 4(a) shows results for constrained path for Set 1, Fig 4(b) shows unconstrained path for same set. Similarly, Fig 4(c) and Fig 4(d) respectively show the constraint and unconstrained paths for set 2. Please read the caption for further explanation.



(a) (b)

Fig. 2. (a): Optimization Output for constrained and unconstrained paths. (b) Satisfaction of constraint (7)



(a) (b) (c)

Fig. 3. (a):plot of path derivatives. (b): Scale factor s_{min} variation and scaling function construction. (c):plot of linear and angular velocities.

VI. Acknowledgment

This work was supported by Department of Science and Technology (DST) India through the grant number: SR/SE/EECE/0114/2010.

VII. Conclusion

This paper presents a novel path and trajectory planning framework for monocular SLAM (VSLAM) systems. Unlike previous planning methodologies for SLAM which dealt with range sensors, this is the first such method that accomplishes such a planner for single camera SLAM. By incorporating issues pertaining to monocular SLAM as appropriate constraints in the planner superior results are reported. These results show VSLAM's camera trajectory estimates being closer to ground truth when the camera executes the path computed by the planner than otherwise. We also similar performance gain when the camera moves with the planned velocity profile. The future scope of the work likes in testing it for larger scale indoor environments and the development of an exploration module for VSLAM that integrates the current planner.

References

[1] T. Bailey. Mobile robot localisation and mapping in extensive outdoor environments, 2002.

[2] J.-L. Blanco. A tutorial on se(3) transformation parameterizations and on-manifold optimization. Technical report, University of Malaga, Sept. 2010.

[3] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225 –234, nov. 2007.

[4] S. M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

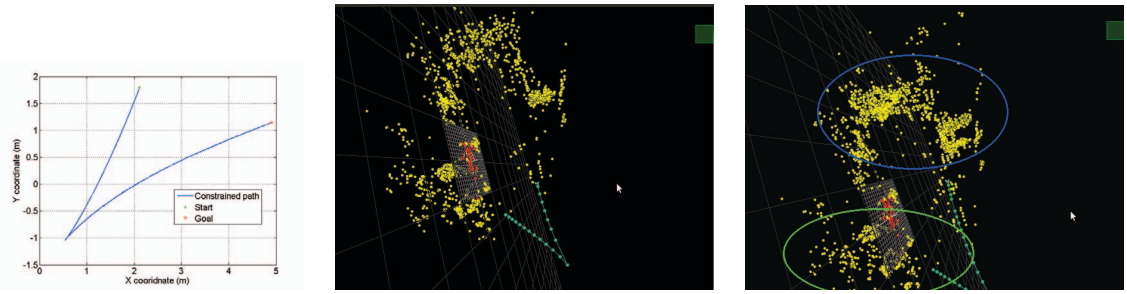
[5] M. Montemerlo. *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2003.

[6] A. Parulkar, P. Shukla, and K. M. Krishna. Fast randomized planner for slam automation. In *CASE*, pages 765–770. IEEE, 2012.

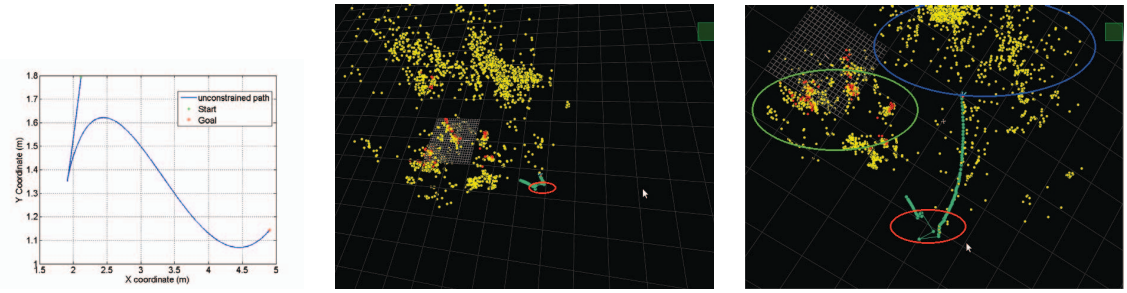
[7] A. Singh, K. Krishna, and S. Saripalli. Planning trajectories on uneven terrain using optimization and non-linear time scaling techniques. In *To appear in Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012.

[8] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Real-time monocular slam: Why filter? In *IEEE International Conference on Robotics and Automation, ICRA 2010, Anchorage, Alaska, USA, 3-7 May 2010*, pages 2657–2664. IEEE, 2010.

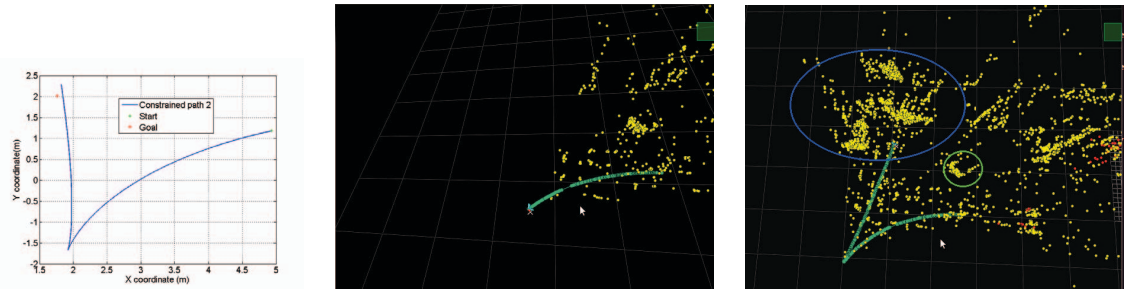
[9] C. P. Tang, P. T. Miller, V. N. Krovi, J.-C. Ryu, and S. K. Agrawal. Kinematic control of a nonholonomic wheeled mobile manipulatora differential flatness approach. In *ASME Dyn. Syst. Control Conf., Ann Arbor, Michigan*, 2008.



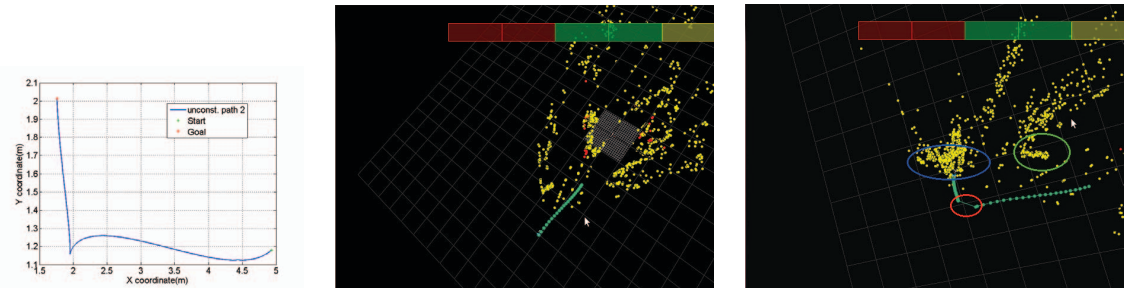
(a) Constrained path for Set 1. Left to right: Path generated by the planner, map and trajectory at intermediate stage, map and trajectory at the end.



(b) Unconstrained path for Set 1. Left to right: Path generated by the planner, map and trajectory at intermediate stage, map and trajectory at the end.



(c) Constrained path for Set 2. Left to right: Path generated by the planner, map and trajectory at intermediate stage, map and trajectory at the end.



(d) Unconstrained path for Set 2. Left to right: Path generated by the planner, map and trajectory at intermediate stage, map and trajectory at the end.

Fig. 4. Figure comprises of screen-shots of map viewer and path plot. Red ovals mark trajectory break. Green ovals mark points belonging to same section of environment, these part were seen before robot takes the big turn. Similarly blue also maps same section, but it's seen by the robot after the turn. For unconstrained path red and blue ovals are not in coherence.