Feature-Rich Path Planning for Robust Navigation of MAVs with Mono-SLAM

Seyed Abbas Sadat, Kyle Chutskoff, Damir Jungic, Jens Wawerla and Richard Vaughan
Autonomy Lab, Simon Fraser University
{sas21, kchutsko, dja6, jwawerla, vaughan}@sfu.ca

***Abstract*— We present a path planning method for MAVs with vision-only MonoSLAM that generates safe paths to a goal according to the information richness of the environment. The planner runs on top of monocular SLAM and uses the available information about structure of the environment and features visibility to find trajectories that maintain visual contact with feature-rich areas. The MAV continuously re-plans as it explores and updates the feature-points in the map. In real-world experiments we show that our system is able to avoid paths that lead into visually-poor sections of the environment by considering the distribution of visual features. If the same system ignores the availability of visually-informative regions in the planning, it is unable to estimate its state accurately and fails to reach the its goal.**

I. INTRODUCTION

Micro Aerial Vehicles (MAVs) have drawn much attention in recent years and been used in many applications such as surveillance, search and rescue, exploration and mapping. Quadrotors, particularly, are small and able to fly in cluttered environments due to high maneuverability. However, autonomous operation of these vehicles require accurate pose information. External motion capture devices require modification of the environment and satellite-based GPS is not available when the view of the sky is obstructed.

Active sensors such as laser range finders or RGB-D sensors, though successfully used for Simultaneous Localization and Mapping (SLAM) in ground robots, pose a problem for quadrotors with very limited payload and energy reserves. On the other hand, a passive camera provides rich sensory information, consumes relatively little power and is very low mass which makes it suitable for localization and mapping with MAVs [1], [2].

Visual SLAM, however, is critically dependent on the availability of feature-rich areas in the environment. The accuracy of localization can be high when there are enough feature points in the camera view but, as the camera moves to a texture-less area, few measurements can be made and the uncertainty in pose estimate grows and SLAM fails. Therefore, for navigating to a goal, a trajectory along which the vehicle stays in feature-rich areas is a safer path. However, this requires a map of the environment with information about the distribution of feature points which is not always available. In this paper we present an approach to navigate safely to a goal location by iterative re-planning. The quadrotor starts with an initialized monocular SLAM system and continuously updates its estimation of the visual feature distribution and re-plans a path to the goal accordingly. We show that our new method is able to accomplish the navigation task, in an environment with irregular feature distribution, by generating paths that, though longer, are in visually rich part of the environment. The same planner is mostly unable to perform the task when feature richness is not considered.

(a) Experimental arena. With no initial map, and using only mono-SLAM, the MAV must navigate around obstacles to reach a goal point. The left-hand part of the room has few visual features and is unsafe for navigation.

(b) A sparse 3D reconstruction of the room. The left-hand side is a poor reconstruction.

Fig. 1: The target environment

The rest of the paper is organized as follows: section II reviews the related work. We describe how the surface mesh of the environment is generated in section III. Feature distribution estimation and plan generation is discussed in section IV followed by the experiments and results in section V. We conclude the paper in section VI.

II. RELATED WORK

There have been several approaches proposed for path planning considering exploration and SLAM performance. In [3], the poses of a Pose SLAM map are used for planning a path with minimum uncertainty in an already explored

Preprint submitted to 2014 IEEE International Conference on Robotics and Automation. Received September 15, 2013.

environment. By planning in a belief roadmap, the generated paths are safe in terms of the accuracy of the localization along the path. In contrast to ours, this method assumes the the environment is already explored. Inspired by [4], efficient EKF-based pose posterior estimation in a known environment is presented in [5], [6] to construct a Belief-RoadMap which can be used to generate plans with minimum uncertainty. However, these methods presuppose uniform availability of sensor measurements in the environment and cannot be directly used in keyframe-based mono-SLAM in an unknown environment. In [7] Rapidly-exploring Random Trees are used to produce information-rich plans to track an external target.In [8] a mechanism for attentional visual SLAM is presented to actively guide the exploration assuming the availability of uniform distribution of features.

Perception-driven Navigation (PDN) presented in [9], [10], [11] is similar to our work in that it also finds regions that have high visual saliency. This approach is used on an underwater vehicle to create a map of a ship hull for inspection by moving along a pre-planned path. If the uncertainty in the pose estimate becomes high, PDN plans to visit already-explored highly-salient areas to reduce the uncertainty in the state estimations. In contrast to our approach, they use other sensor types, apart from camera, to add constraints in their Pose-Graph SLAM which makes the SLAM tolerant of the absence of visual features. Additionally, their goal is to explore and obtain a map of the whole targeted area, whereas our goal is to navigate to a given location as fast and as safely as possible, with only enough exploration to find such a path.

A visual appearance mapping system is explained in [12] which is used to generate plans in image space for unicycle-like vehicles. It uses feature trajectories to select image sequences (as waypoints) that have high number of features in common. However this process is done after a complete exploration of the environment and the creation of a topological map, which make it different to our case. In [13], a planning method is presented to find a safe path in which some landmarks are always observed to reduce uncertainty in localization. A different approach but similar story can be found in [14] as well. Both these methods use known artificial landmarks with given positions for pose estimation which makes them different from our method. A similar biologically-inspired approach is proposed in [15] which generates a navigation field based on the landmark observability and confidence. Keeping a known object within the camera field-of-view has been studied in visual servoing of manipulator arms as well. Examples of such studies are [16], [17].

III. Sparse 3D Reconstruction

We use Parallel Tracking and Mapping (PTAM) [18] as our monocular visual SLAM system. PTAM has been successfully used in aerial vehicles navigation [2], [19]. It divides SLAM into tracking and mapping, each running in a separate thread. The tracking component tracks salient features (FAST corners [20]) in the camera image by matching them with the feature points already stored in the map. For matching features, first a motion model is applied to the camera pose which results in a prediction of the camera pose. Then the map points are projected back onto the camera frame and compared with the extracted salient features in the camera frame by local search in the image. After feature matching, the new pose of the camera is calculated by minimizing the error between the position of the observed features in the image and the projection of the map points onto the camera frame. Tracking is performed in every frame and, based on some heuristics, some images are selected and inserted in the map as keyframes. The mapping component performs local and global bundle adjustments to refine the pose for all keyframes (except the first keyframe) and all the points in the map.

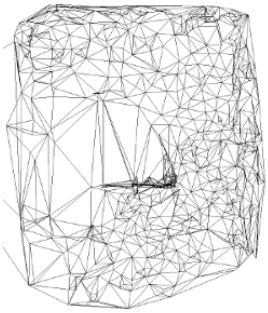
PTAM represents the map of the environment by a cloud of feature points that are measured in two or more keyframes. To produce a mesh of the environment, we perform a sparse 3D reconstruction of the explored area by using the method introduced in [21]. This approach is suitable for scene reconstruction using features generated by *structure from motion* (like PTAM), and produces acceptable results even when using sparse point clouds. It first creates a 3D Delaunay triangulation of the map points[1]. Then using graph-cuts, it extracts the surface mesh embedded in the Delaunay triangulation that minimizes an energy function. The energy function is based on the visibility constraints imposed by the line of sight of each feature. To reduce the computational cost, we do not include the photo-consistency part of the energy function (see [21] for details).

We use the resulting surface mesh in two ways: (i) to avoid obstacles; and (ii) to estimate the feature richness of different regions in the environment (described in the next section). For both these purposes, we use ray-tracing to find out if a line segment $\overline{p_s p_e}$ collides with the surface. We efficiently perform ray-tracing using the original Delaunay triangulation which embeds the surface mesh as follows: The tetrahedron that contains the start point p_s is looked up (already an efficient operation). Then, the next tetrahedron that collides with the line is found by checking all neighbour cells (two tetrahedra are neighbours if they have a common triangle). This process is done iteratively and terminates upon reaching the tetrahedron that contains the line end point. If during this process a the common triangle between two consecutive tetrahedron belongs to the surface mesh, it implicates a collision. For line segments where either of the end points is outside the convex hull of the point cloud (and is thus not contained by any tetrahedron), we perform ray tracing using the line segments formed by the middle point of the line and either one of the end points. The same process is used if both the end points are outside the convex hull unless the length of the line segment is smaller than some threshold. Note that the ray tracing queries are performed very efficiently by moving along a line segment in non-uniform steps because most of the queries encountered during planning are almost

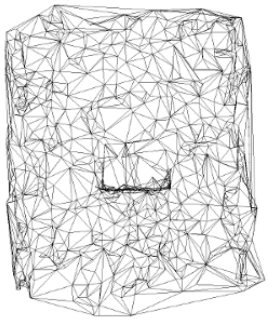
[1]We used the CGAL library for this purpose

(a) Feature-poor

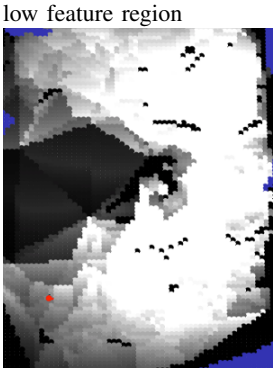
(b) Feature-rich



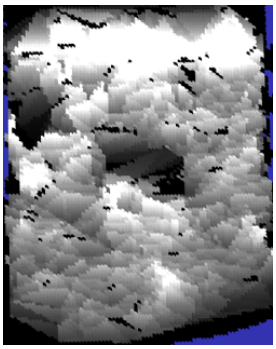
(c) Feature-poor. Note the low triangle density in the low feature region



(d) Feature-rich. The triangle density is high and uniform in the environment.



(e) The viewpoint score of feature-poor environment.



(f) The viewpoint score of feature-rich environment.

Fig. 2: Two configurations of the room. (a,b): 3D reconstruction of the room. (c,d): The generated mesh of the room. (e,f): a map of the feature-richness score of a sample of viewpoints. The direction of all the samples are towards the top of the image.

completely contained in the convex hull of the feature cloud.

IV. Planning and Navigation

We propose a novel approach in which the MAV navigates to its goal safely by iterative evaluation of availability of feature-rich regions and re-planning. The drone starts with a mesh produced using the initial feature cloud of PTAM. Then a set of candidate plans is generated using optimal version of Rapidly-Exploring Random Tree (RRT*) [22]. We eventually select the plan that optimizes a cost function that includes both path length and the expected amount of features along the path (explained in Sections IV-A and IV-B). A plan P consists of a list of waypoints $p_i = [x_i, y_i, z_i, \theta_i]$ which specifies a 3D point at location $[x_i, y_i, z_i]$ in the world coordinate system and a rotation of θ_i along the world Z axis. The drone visits the waypoints one by one using a PID controller. At each waypoint the whole scene is reconstructed and the 3D mesh is updated. The drone re-plans after a fixed part of the plan has been executed (visited 4 waypoints in our experiments). This process is repeated until the drone reaches the goal (within a radius of 1 m). The reason behind re-planning is that when the drone moves along the plan, new points are added to the map and the 3D mesh of the scene is changed. Therefore, re-planning might generate a better path than the current plan since it uses the most up-to-date information about the world.

Detecting a low number of useful features or predicting collisions will result in re-planning as follows: as the drone is following the waypoints, we track the number of features that are successfully matched to map points and used towards pose estimation in PTAM. If it is lower than a threshold (50 features), the drone changes its current target waypoint to the previous waypoint and re-plans when it reaches that waypoint. Detecting a future collision along the plan (after updating the mesh) triggers the same behaviour. In both cases, moving to the last visited waypoint provides more free space for the drone to manoeuvre and get out of unsafe situations encountered during exploration of unknown environments.

Next, we describe how the expected amount of visual features is estimated and used in the planning cost function.

A. Viewpoint Score

Generally in key-frame based visual SLAM systems, the view points that are very close to key-frames in terms of distance and viewing angle, with a high chance can extract and match many features to the map points. But this criteria is very restrictive and does not provide a way to predict if a view point far from the key-frames is texture rich enough for SLAM to work. Therefore we devised a more general way using local density of the reconstruction mesh triangles to predict the expected amount of features in an image taken from a viewpoint.

The scoring function contains two parts:

$$s(P) = D(P)V(P)$$

where $P \in SE(3)$ is a 6-DoF pose containing translation $P_t \in \Re^3$ and rotation $P_r \in SO(3)$ that specify the viewpoint. We define $D(P)$ as a measure of the texture quality and $V(P)$ as an estimate of how well the target area is visible from the view point. The calculation of these functions is as follows: a ray is cast from P along the viewing angle to find the point $c \in \Re^3$ on the mesh that will be in the center of the camera view. Then the set T of all the triangles on the mesh with centres within a radius r(1 m) of c are found. Next, the normal of the local surface $\overrightarrow{n} \in \Re^3$ is estimated by fitting a plane to the triangles (Figure 4). Now we define $D(P)$ as:

$$D(P) = qm$$

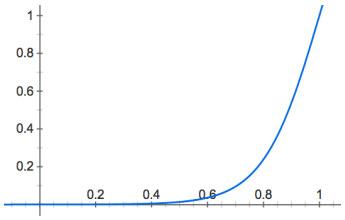


Fig. 3: The Kernel function used in scoring viewpoints.

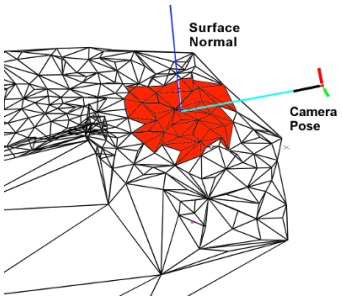


Fig. 4: Finding the local density of the mesh.

Here m is the density of the triangles in T, that is number of triangles divided by the sum of their area. q with $0 \leq q \leq 1$ is the quality of the plane fitting, where 0 means that the variance is the same along any plane going through the best fitting line, and 1 means that the variance is null orthogonally to the best fitting plane. The second component of the scoring function, $V(P)$ penalizes the view points as follows:

$$V(P) = \cos(\alpha) \exp(-|d_{pref} - \|\overrightarrow{cP_t}\||).$$

Here α is the angle between the local surface normal $\overrightarrow{n}$ and $\overrightarrow{cP_t}$ and d_{pref} is the preferred viewing distance. Therefore, view points at the preferred distance and fronto-parallel to the target surface receive high scores. On the contrary, viewing angles away from surface normal or very close to or far from the surface are assigned low scores.

With the above $s(P)$ function a very dense patch of triangles might lead to a very high score, therefore we define an upper bound for $s(P)$ and normalize the the score as $s_{norm}(P)$. We use a kernel function $f(x) = \frac{a}{1+e^{bx+c}}$, shown in Figure 3, to increase the contrast between high and low scores which leads to the final form of the scoring function:

$$Score(P) = \begin{cases} \frac{a}{1+e^{bs_{norm}(P)+c}} & P \text{ far from key-frames} \\ s_{default} & \text{Otherwise} \end{cases}$$

Note that, since the viewpoints that are far from the key-frames (translational distance more than $4m$ or yaw distance more than $90°$) cannot be reliably scored using the above method, they are assigned a default score $s_{default}$.

B. Planning

Planning is based on RRT*. The main idea is to generate a set of candidate paths and then select the one that has minimum cost. The cost function includes both traveling distance and viewpoint score of the waypoints along the path. The Feature-Rich RRT* is described in Algorithm 1.

Algorithm 1 Feature-Rich RRT*

Input: Reconstructed mesh M, Workspace X_{all}, Motion constraints K, Start pose $\mathbf{x}_{start}$, Goal position $\mathbf{p}_{goal}$, Number of Iterations N.
$V \leftarrow \{\mathbf{x}_{start}\}$, $E \leftarrow \emptyset$, $Tree \leftarrow (V, E)$
$C \leftarrow \emptyset$ *The set of candidate nodes.*
for $i = 1...N$ **do**
  $\mathbf{x}_{sample} \leftarrow Sample(X_{all})$
  $\mathbf{x}_{near} \leftarrow Nearest(\mathbf{x}_{sample}, V \backslash C)$
  $\mathbf{x}_{new} \leftarrow Steer(\mathbf{x}_{near}, \mathbf{x}_{sample}, K)$
  if $NoCollision(\mathbf{x}_{near}, \mathbf{x}_{new}, M)$ AND $AcceptableScore(\mathbf{x}_{near}, \mathbf{x}_{new})$ **then**
    $V \leftarrow V \bigcup \{\mathbf{x}_{new}\}$
    $E \leftarrow E \bigcup \{(\mathbf{x}_{near}, \mathbf{x}_{new})\}$
    $Tree \leftarrow Rewire(\mathbf{x}_{new}, Tree)$
    if $IsCandidate(\mathbf{x}_{new}, \mathbf{p}_{goal})$ **then**
      $C \leftarrow C \bigcup \{\mathbf{x}_{new}\}$
    end if
  else
    $Delete(\mathbf{x}_{new})$
  end if
end for
return $ExtractPlansToNodes(C, Tree)$

The algorithm extends the tree incrementally as follows: It samples a point from the configuration space and generates a new node using the $Steer()$ function to extend the nearest node to the sample. The $Steer()$ function satisfies constraints defined for the drone's motion. Specifically, the drone is allowed to move only in the direction of the camera (within an angle of $45°$) unless it is moving to a place that it has been before. This is to ensure that as the drone can detect obstacles that are on its way. The tree is extended with the new node if the resulting edge does not collide with the mesh ($NoCollision()$ function) and the viewpoint score along the edge is above a threshold ($AcceptableScore()$ function). When a new node is added to the tree, we maintain the optimality of the tree in traveling cost using the $Rewire()$ function. This function implements the approach used in [22] for RRT*.

A newly added node enters the $Candidate$ set if either it is within some radius of the goal or it satisfies all of the following conditions: a) it has a default viewpoint score b) the line connecting the node to the goal is collision free and the angle it makes with the direction of the node is less than $90°$ (i.e. the node is facing the goal). The nodes in the $Candidate$ set are not expanded during the subsequent samplings. Therefore the candidate plans might not go all the way to the goal but stop at waypoints from where a straight line will result in a collision-free path. Note that these decisions are based on the currently available and usually incomplete information about the world. In case new

obstacles are encountered during the execution of the plan, the drone re-plans as explained in Section IV.

The above algorithm produces a set Q of plans. Each plan $P_i \in Q$ is a list of N_i waypoints p_{ij} (with current pose at the start of the list). We select the plan that minimizes the following cost function as the best plan:

$$Cost(P_i) = \sum_{j=1}^{N_i-1} \frac{Dist(p_{ij}, p_{ij+1})}{MinScore(p_{ij}, p_{ij+1})} + EstimateDist(p_{iN_i}, \mathbf{p}_{goal})$$

In the above equation, $Dist(.,.)$ calculates the distance between two waypoints considering both the translation and yaw (4 DoF). $MinScore$ finds the smallest viewpoint score along the edge connecting two points by some sampling viewpoints in fixed intervals along the edge. $EstimateDist(.,.)$ estimates the traveling distance from the last waypoint to the goal considering the motion constraints mentioned before.

V. Experiments and Results

We tested our proposed method with a real MAV in a $8m \times 6m$ room. The commercially available Parrot AR.Drone 2.0 quadrotor was used as our platform. The only sensor that is used in our experiments is the forward camera which has a resolution of 640x320. We changed the camera angle so that it has a pitch of $\approx 45^\circ$ towards the ground to capture images from both forward and bottom areas. The robot communicates with an off-board computer over Wi-fi and all the sensors data are read using *ardrone_autonomy*[2] , a ROS driver for AR.Drone quadrotors. Translational control commands in XYZ and yaw rotational speed are sent to the drone using the driver as well. Also after the initialization of PTAM, we use a fiducial marker tracker[3] to fix the scale of the map and transform it to a predefined coordinate system. The task was to navigate from a predefined starting position to a fixed 3 DoF goal position. The 3D reconstruction of the room is shown in Figure 2. The start and goal positions are also indicated with a square and a star shape respectively. A wall of cardboard boxes was placed in the middle of the room as an obstacle. This world configuration provides two paths from the start to the goal location. A short, direct path on the left of the obstacle and a larger path on the right. it is worthwhile pointing out that the right path requires a lot more turning manoeuvres and hence is a lot more challenging for the drone. We performed experiments with two different visual feature distributions in the room (see Figure 2a, 2b): The first one has low amount of texture on the left side of the obstacle whereas the second one is approximately uniformly textured. For each room configuration we performed 10 trials using FR-RRT* and RRT*. If the drone reached the goal (within a radius of $1m$), the trial was *successful*. A trial was marked a *failure* if the drone lost track of the features, therefore failed to estimate its pose and subsequently was unable to navigate.

[2]https://github.com/AutonomyLab/ardrone_autonomy

[3]http://www.ros.org/wiki/ar_track_alvar

	Success	Failure
FR-RRT*	8	2
RRT*	3	7

(a) Experiment 1: Irregularly textured room

	Success	Failure
FR-RRT*	9	1
RRT*	10	0

(b) Experiment 2: Uniformly textured room

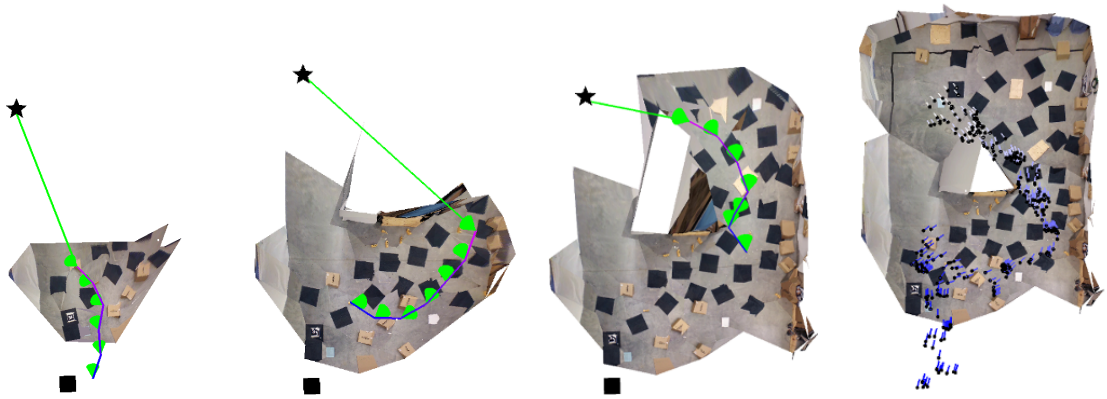
TABLE I: Results

The results are shown in Table I. In the first environment configuration the shortest path found by RRT is always passing through the left side of the room which is a textureless area and therefore 70% of the time the vehicle fails to reach the goal (Table Ia). However, RRT sometimes generated a sub-optimal path on which some of the texture patches behind the wall were captured in camera images and therefore PTAM was able to continue pose estimation. By contrast, using FR-RRT* the drone only failed 20% of the times to reach the goal. This was achieved by choosing a safer, texture-rich path on the right side of the room. In the fully-textured environment, FR-RRT* performs almost the same as RRT* (Table Ib).

To show the behaviour of the drone, sample plans generated by FR-RRT* are illustrated in Figure 5. They indicate that the initial plans are close to the shortest path (Figure 5a). But, as the vehicle visits some waypoints close to the textureless area and updates the 3D reconstruction, the next plans deviate from the shortest path and are more inclined to pass through areas with high texture due to high cost of waypoints in texture-poor areas (Figure 5b, 5c). In contrast, RRT paths are uninformed about the saliency of the environment texture and hence only optimize for travelled distance. As shown in Figure 6, RRT paths are unsafe and unable to direct the vehicle to the goal.

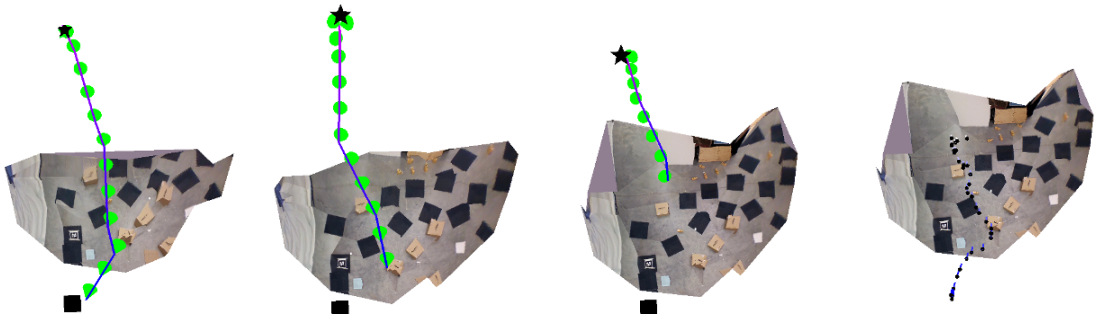
Another observation is that when images captured by the camera are not feature-rich enough, PTAM needs more keyframes to be able to provide enough map-points for pose estimation. This leads to a high density of keyframes in low-textured area and an increase in memory consumption. However, if the number of features found in key-frames is high (i.e. moving in texture-rich region) there is no need for a high density of key-frames and PTAM is able to measure features in the local space without inserting new key-frames. This means fewer key-frames and lower memory use.

During the experiments we observed that in some cases the pose estimate is not accurate enough to avoid collisions. This could be due to inaccurate scale estimation during initialization. Since the safety distance for the drone is in absolute scale, there is a possibility of collision with obstacles in these situations. We terminated the trials in which the drone was too close to obstacles (9 cases in all 54 experiments). Also in 3 cases the drone ended up too close to obstacles (e.g by turning and seeing a new nearby obstacle) such that no valid paths could be found since it was



(a) 1st Plan (b) 3rd Plan (c) 5th Plan (d) Final Trajectory

Fig. 5: Sample plans generated by FR-RRT* in a trial. The MAV reaches the goal, after replanning to avoid a feature-poor region.



(a) 1st Plan (b) 2nd Plan (c) 3rd Plan (d) Final Trajectory

Fig. 6: Plans generated by RRT in a sample trial. The MAV does not reach the goal, as it is unable to localize when it reaches a feature-poor area.

already in a collision state. All these trials were excluded from the analysis as they are not caused by our method but the limitations of the vision-only SLAM and navigations.

VI. Conclusions and Future Work

In this paper we presented a novel method for reliable navigation in unknown environments. By estimating feature-richness and considering it during path planning we direct the drone's sensor towards feature rich regions. Iterative re-planning ensures that the latest information is taken into account and a safe path (collision-free and feature-rich) to the goal is found. Our approach does not use any prior knowledge about the distribution of visual features or obstacles in the environment and only relies on the information sensed by the drone as it navigates towards the goal.

However, for viewpoint evaluation we only relied on the availability of feature-rich patches in the camera view. An interesting avenue for future work is to include the saliency of the feature patches in the scoring so that regions with distinctive patterns are favoured over repetitive patterns. This is in particular useful in natural outdoors environments in which regions with highly-similar grass textures might lead to feature mis-matching.

References

[1] F. Fraundorfer, L. Heng, D. Honegger, G. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a quadrotor mav," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 4557–4564, Oct.

[2] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[3] R. Valencia, J. Andrade-Cetto, and J. M. Porta, "Path planning in belief space with pose SLAM," *2011 IEEE International Conference on Robotics and Automation*, pp. 78–83, May 2011.

[4] N. Roy, W. Burgard, D. Fox, and S. Thrun, "Coastal navigation-mobile robot navigation with uncertainty in dynamic environments," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 1, pp. 35–40 vol.1, 1999.

[5] A. Bry and N. Roy, "Rapidly-exploring Random Belief Trees for motion planning under uncertainty," in *2011 IEEE International Conference on Robotics and Automation*, pp. 723–730, IEEE, May 2011.

[6] S. Prentice and N. Roy, "Planning in information space for a quadrotor helicopter in a GPS-denied environment," in *2008 IEEE International Conference on Robotics and Automation*, pp. 1814–1820, IEEE, May 2008.

[7] D. Levine, B. Luders, and J. P. How, "Information-rich path planning with general constraints using rapidly-exploring random trees," 2010.

[8] S. Frintrop and P. Jensfelt, "Active gaze control for attentional visual SLAM," *2008 IEEE International Conference on Robotics and Automation*, pp. 3690–3697, May 2008.

[9] A. Kim and R. M. Eustice, "Combined visually and geometrically informative link hypothesis for pose-graph visual slam using bag-of-words," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 1647–1654, IEEE, 2011.

[10] A. Kim and R. M. Eustice, "Real-time visual slam for autonomous underwater hull inspection using visual saliency."

[11] A. Kim and R. M. Eustice, "Perception-driven navigation: Active visual slam for robotic area coverage," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Karlsruhe, Germany), May 2013. Accepted, To Appear.

[12] D. Fontanelli, P. Salaris, F. A. W. Belo, and A. Bicchi, "Visual Appearance Mapping for Optimal Vision Based Servoing," pp. 353–362, 2009.

[13] I. Moon, J. Miura, and Y. Shirai, "On-line viewpoint and motion planning for efficient visual navigation under uncertainty," *Robotics and Autonomous Systems*, vol. 28, pp. 237–248, Aug. 1999.

[14] a. Yamashita, K. Fujita, T. Kaneko, and H. Asama, "Path and viewpoint planning of mobile robots with multiple observation strategies," *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 4, pp. 3195–3200, 2004.

[15] G. Bianco and A. Zelinsky, "Biologically-inspired visual landmark learning and navigation for mobile robots," in *Intelligent Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ International Conference on*, vol. 2, pp. 671–676, IEEE, 1999.

[16] S. Léonard, E. A. Croft, and J. J. Little, "Dynamic visibility checking for vision-based motion planning," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 2283–2288, IEEE, 2008.

[17] D.-H. Park, J.-H. Kwon, and I.-J. Ha, "Novel position-based visual servoing approach to robust global stability under field-of-view constraint," *Industrial Electronics, IEEE Transactions on*, vol. 59, no. 12, pp. 4735–4752, Dec.

[18] 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*, pp. 225–234, IEEE, 2007.

[19] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter.," in *IROS*, pp. 2815–2821, IEEE, 2012.

[20] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *In European Conference on Computer Vision*, pp. 430–443, 2006.

[21] P. Labatut, J.-P. Pons, and R. Keriven, "Efficient Multi-View Reconstruction of Large-Scale Scenes using Interest Points, Delaunay Triangulation and Graph Cuts," in *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, IEEE, 2007.

[22] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, pp. 846–894, June 2011.