Perception-aware Receding Horizon Navigation for MAVs

Zichao Zhang, Davide Scaramuzza

***Abstract*— To reach a given destination safely and accurately, a micro aerial vehicle needs to be able to avoid obstacles and minimize its state estimation uncertainty at the same time. To achieve this goal, we propose a perception-aware receding horizon approach. In our method, a single forward-looking camera is used for state estimation and mapping. Using the information from the monocular state estimation and mapping system, we generate a library of candidate trajectories and evaluate them in terms of perception quality, collision probability, and distance to the goal. The best trajectory to execute is then selected as the one that maximizes a reward function based on these three metrics. To the best of our knowledge, this is the first work that integrates active vision within a receding horizon navigation framework for a goal reaching task. We demonstrate by simulation and real-world experiments on an actual quadrotor that our *active* approach leads to improved state estimation accuracy in a goal-reaching task when compared to a purely-reactive navigation system, especially in difficult scenes (e.g., weak texture).**

Supplementary Materials

A video showing the experiments can be found at `https://youtu.be/761zxZMeQNo`

I. Introduction

Being both agile and versatile, micro aerial vehicles (MAVs) are suitable for various tasks such as industrial inspection, agriculture, and goods delivery. To enable MAVs to operate autonomously in an unknown environment, reliable on-board state estimation is necessary. Among different sensors for state estimation, cameras are lightweight and power efficient and, therefore, ideal for MAVs due to their limited payload and battery life.

For vision-based state estimation, it is well-known that the motion of a camera has a significant impact on the estimation accuracy [1]. Therefore, the motion of MAVs should be planned considering both the task at hand and the perception quality.

In this work, we focus on the task of reaching a given goal with the highest accuracy while avoiding obstacles in the environment. When it comes to planning a trajectory in partially-unknown environments, the common approach is to couple a global planner with a receding-horizon method: while the global trajectory is being executed, a local planner is used to generate and search feasible, collision-free trajectories in a local robot-centric map of the environment. While this approach has been successfully implemented in several recent works [2], [3], [4], [5], [6], these do not take perception constraints into account (e.g., favour texture rich areas to minimize state estimation uncertainty).

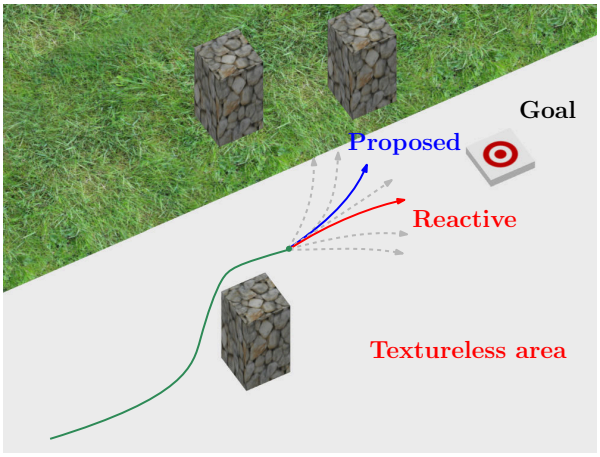

Fig. 1: Illustration of the proposed perception-aware receding horizon navigation system. Our method is able to select a suitable motion (blue) that can simultaneously avoid obstacles, reach a given destination and minimize state estimation uncertainty. By contrast, a purely-reactive navigation scheme (red) can enter textureless area, resulting in large state estimation error and the failure to reach the given destination.

Very little work has been done to consider the *quality of perception* and its influence on the accuracy of visual odometry in a receding horizon fashion. The problem of choosing the motion that maximizes the accuracy of state estimation is known as *active SLAM* [1]. This problem is often solved by optimizing the motion trajectory in a global map, which is usually expensive to compute. Instead, we propose to solve the active SLAM problem in a receding horizon fashion. In particular, we limit the scope of the problem to a local map and a short horizon and continuously recompute a suitable local trajectory for state estimation. This way, the active SLAM problem can be solved efficiently and can be integrated naturally into a receding horizon setting.

To demonstrate the usefulness of the idea, we implement a *perception-aware receding horizon navigation* system. Specifically, we use a monocular odometry and mapping system for state estimation and mapping. Instead of directly optimizing the motion parameters, we use an efficient trajectory generation method (minimum jerk [7]) to generate a library of candidate trajectories within a short horizon. For each of the trajectories, we evaluate its perception quality, the probability of collision, and its distance to the goal using the information from the monocular odometry and mapping system. The trajectory to execute is then selected as the

The authors are with the Robotics and Perception Group, Dep. of Informatics, University of Zurich , and Dep. of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland—`http://rpg.ifi.uzh.ch`. This research was supported by the National Centre of Competence in Research (NCCR) Robotics, through the Swiss National Science Foundation, the SNSF-ERC Starting Grant, and the DARPA FLA program.

one that maximizes a reward function based on these three metrics. The trajectory generation and evaluation process is repeated online.

A. Related Work

1) Receding Horizon Planning for MAVs: Receding horizon planning has been widely used to generate collision-free trajectories online. Liu *et al.* [5] developed a framework of receding horizon planning, which continuously plans trajectories within a safe flight corridor. Chen *et al.* [2] also optimized trajectories by using a similar corridor representation for the free space. Differently, Landry *et al.* [4] represented the free space as convex volumes, and enforced each trajectory segment to stay in the volumes to avoid collision. Mohta *et al.* [6] first planned a safe path consisting of straight line segments and then optimized safe trajectories within a finite horizon by enforcing the trajectories to be close to the safe path.

Instead of resorting to optimization, sampling-based methods have also been studied. Florence *et al.* [8] generated a library of candidate trajectories in a limited time horizon and selected the one to execute based on a cost function including the collision risk and the distance to a given destination. Matthies *et al.* [3] exploited a rapid-exploring random tree (RRT) planner and performed collision checking by projecting the sampled trajectories into the disparity space.

The aforementioned methods are mainly designed for collision free motion planning in a receding horizon fashion. None of them, however, took perception quality into consideration, which, as we later show, is extremely important in environments containing visually degraded areas (i.e., poor texture).

2) Active SLAM for MAVs: Different active SLAM approaches for MAVs have been proposed. Mostegel *et al.* [9] designed a set of heuristic metrics for evaluating localization quality and map generation likelihood. Depending the current state estimation quality, their control scheme decides whether to maximize the localization quality or map generation likelihood. Sabdat *et al.* [10] proposed a perception quality metric that combines the number of visible features and the viewing angle with respect to visible surfaces and incorporated this metric into Rapidly-Exploring Random Tree (RRT*) [11] for planning. Alzugaray *et al.* [12] first sampled positions near obstacles based on the intuition that pose estimation error is small when the camera is close to the features on obstacles. Then path planning was carried out based on the sampled positions. Different from previous methods, which focus on the geometric information, Costante *et al.* [13] additionally incorporated the photometric information of the scene to calculate the localization uncertainty of the camera. Some recent work also considered different sensor configurations, such as the global positioning system (GPS) [14] and inertial measurements units (IMU) [15]. As a complementary problem, researchers have also proposed methods to minimize reconstruction uncertainty [16] [17].

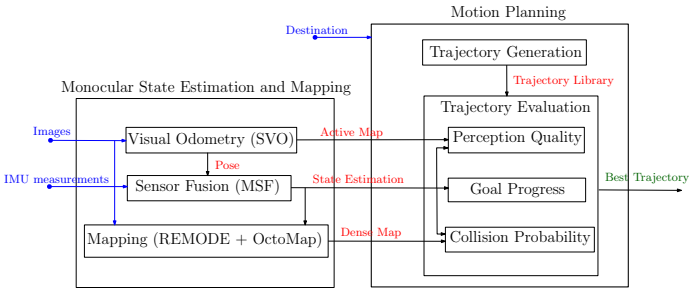

Fig. 2: Overview of the proposed system. The input of the system is marked as blue, communication among different modules red and the output green.

While the aforementioned methods maintain a global map for planning, some recent work focuses on planning based on short term accuracy. Rong *et al.* [18] evaluated short-term perception quality using empirical observability gramian. They demonstrated their metric can successfully reflect the change of perception quality, e.g., in visually degraded areas. Papachristos *et al.* [19] proposed a two-step planning strategy for perception-aware exploration. They first planned a view that maximizes the information gain in terms of explored space, then a second planner sampled views locally and selected the one with the least state estimation uncertainty. The proposed method is similar to [19] in that we also evaluate the perception quality of candidate motion within a limited time horizon. The differences are twofold. First, we focus on the task of reaching a given destination instead of exploration. Second, in their method, the obstacle avoidance is done by planning in free space, while in our method it is considered in a unified receding horizon navigation framework.

B. Contributions and Outline

To the best of our knowledge, this is the first work that integrates active vision in a receding horizon navigation framework for a goal reaching task. We demonstrate by simulation and real-world experiments that our *active* approach leads to improved state estimation accuracy when compared to a purely-reactive navigation system, especially in difficult scenes (e.g., weak texture).

The rest of the paper is structured as follows. In Section II, we give an overview of our perception-aware receding horizon navigation system. In Section III, we describe our monocular state estimation and mapping system. Then in Section IV, we detail how to plan the motion of a quadrotor using the information from the system in Section III. To demonstrate the effectiveness of the proposed method, we show simulation and real-world experiments in Section V.

II. System Overview

Fig. 2 gives an overview of the proposed system, which consists of a monocular state estimation and mapping system and a motion planning system.

The monocular state estimation and mapping system is responsible to provide the state estimation of the MAV and the map for different purposes. We first use SVO [20]

to estimate the 6-Degrees-of-Freedom (DOF) pose of the camera. The pose estimation is further fused with the IMU measurement using the Multi-Sensor Fusion (MSF) software [21] to get the correct scale and the extra velocity estimation, which is necessary for trajectory generation and control. Then, the state estimate and the images are fed into a variant of REMODE [22] to get a dense 3D map of the frontal view. Before using the dense map for motion planning, we utilize OctoMap [23] to reduce the noise in the map. In addition to the state estimation and the dense map, we also get an *active map* from SVO, which will be detailed in Section III.

The motion planning system consists of a trajectory generation module and a trajectory evaluation module. We use an efficient trajectory generation method [7] to generate a library of candidate trajectories based on the current state estimate. We then evaluate each trajectory based on three metrics:

- the *collision probability*, based on dense 3D map from REMODE and Octomap (Section IV-C),
- the *perception quality*, based on the *active map* from SVO (Section IV-D),
- and the *goal progress* (which is a function of the distance to the goal), based on the current state estimate and given goal (Section IV-E).

Based on the evaluation, we select the best trajectory to execute and send the desired state to the controller.

III. Monocular State Estimation and Mapping

Our monocular state estimation and mapping is similar to the one proposed in [24]. We use SVO plus MSF for state estimation, and REMODE to generate a dense pointcloud for obstacle avoidance. SVO is an extremely efficient VO algorithm that is suitable for resource-constrained systems, such as MAVs. REMODE, on the other hand, is originally designed to run on a Graphic Processing Unit (GPU). In [24], the authors proposed several modifications to enable REMODE to execute on a smartphone processor. We refer the reader to [24] for more details.

Compared to [24], our system is different in the following aspects. First, to evaluate the perception quality, we also extract an *active map* from SVO. Internally SVO maintains a set of sparse points, which can be divided into two categories: *landmarks* and *seeds*. Landmarks are 3D points that have been observed multiple times from different frames, and their positions are already well estimated. In contrast, seeds are 3D points, whose positions are not accurately estimated yet.

Intuitively, it is the landmarks that contribute the most to the accuracy of the pose estimate of a frame. Therefore, we extract the visible landmarks from keyframes that overlap with the current frame. We denote these landmarks as the *active map* (cf. Fig. 3). Later, we will show how to use the active map to evaluate the *perception quality* (Section IV-D). Second, we use Octomap to further reduce the noise in the output of REMODE. If the dense pointcloud contains too many outliers, the trajectory evaluation module will wrongly estimate the collision probability, resulting in unnecessary collision avoidance maneuvers. Fig. 4 shows an example of the active map and dense map, which will be denoted as M_A and M_D, respectively, in the rest of the paper.

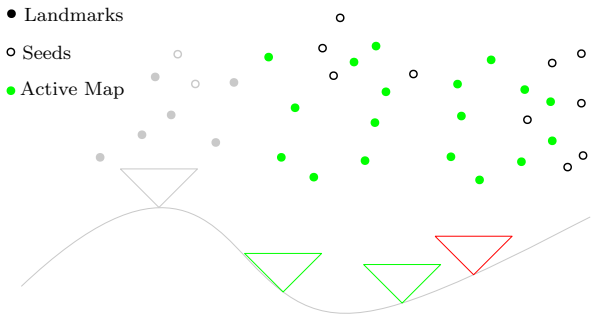

Fig. 3: The extraction of the *active map*. For the current frame (red triangle), we find the keyframes that have overlap with it (green triangles). Then the landmarks that are visible in these keyframes are extracted as the active map (green solid dots). Keyframes that have no overlap (gray triangles) and seeds (circles) are not considered in the active map.

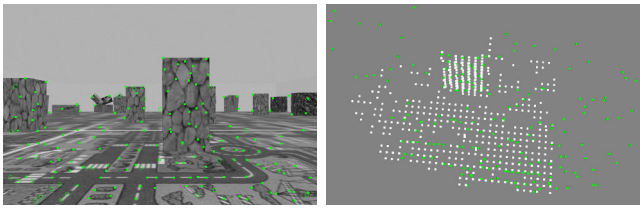

Fig. 4: An example of the *active map* and *dense map*. The left one is the image from the camera, where the solid green dots are *landmarks* and the circles are *seeds*. The right one shows the corresponding active map (green) and the dense map (white).

IV. Trajectory Generation and Evaluation

In this section, we describe how we generate a library of candidate trajectories and select the best one to execute.

A. Notations

A pre-subscript denotes the frame where the quantity is expressed. We use $\mathrm{T}_{ab} \in SE(3)$ to represent the rigid body transformation of frame b in frame a, which transforms a point in frame b to frame a as ${}_a\mathbf{p} = \mathrm{T}_{ab} \cdot {}_b\mathbf{p}$. If a quantity/transformation is expressed in the world frame, we omit the (pre-)subscript for simplicity.

B. Trajectory Generation

We use [7] to generate our candidate trajectories. Conceptually, the trajectory generation process, denoted as $g(\cdot)$, is

$$f(t) \leftarrow g(\mathbf{p}_0, \mathbf{v}_0, \mathbf{p}_f, t_f). \tag{1}$$

$\mathbf{p}_0$ and $\mathbf{v}_0$ are the initial position and velocity of the trajectory in the world frame, which come from the current state estimation. $\mathbf{p}_f$ is the desired end point of the trajectory, and t_f is the time it takes for the quadrotor to follow the trajectory and reach $\mathbf{p}_f$. The output is a function $f(t)$. For $t \in [0, t_f]$, $f(t)$ gives the state (position, orientation, velocity) on the trajectory at time t.

Now we detail how to select $\mathbf{p}_f$ and t_f for generating a library of trajectories. We plan the trajectories by selecting the end points on an arc in front of the quadrotor, since a

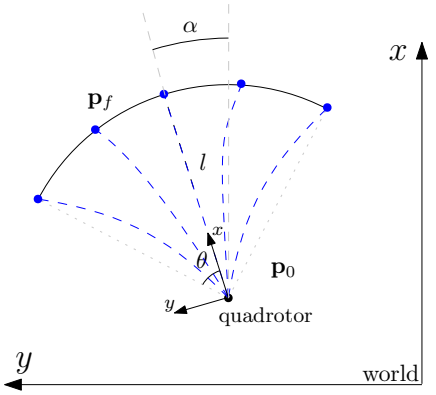

Fig. 5: Trajectory generation. We uniformly sample N points on an arc, defined by θ and l, in front of the quadrotor. The candidate trajectories are shown in blue. α is the yaw angle of the quadrotor.

forward-looking camera is used and we want move in visible directions. To this end, we need to know the radius l and angle θ of the arc, as illustrated in Fig. 5. While l is simply a design parameter, θ is calculated as

$$\theta = \max(k_\theta \|\mathbf{v}_0\|, \theta_{\max}), \tag{2}$$

where k_θ is a constant. Intuitively, θ increases with the velocity until a maximum value $\theta_{\max}$. Using θ and l, we uniformly sample the end points on the arc.

As for t_f, we use the following formula:

$$t_f = \min(\frac{l}{\|\mathbf{v}_0\| + \Delta v}, \frac{l}{v_{\max}}), \tag{3}$$

where Δv in a constant value, and $v_{\max}$ is the maximum velocity allowed. (3) means that we want to increase the velocity until the maximum value is reached.

After generating N candidate trajectories, we sample J poses from each trajectory by a constant time interval. Finally, we have a set of sampled poses for each candidate trajectory, which can be formulated as

$$C = \{c_i\}_{i=1}^{N}, \tag{4}$$

$$c_i = \{\mathrm{T}_j\}_{j=1}^{J}, \tag{5}$$

where $\mathrm{T}_j \in SE(3)$ is the jth pose on the trajectory expressed in the world frame. Since the yaw of the quadrotor is not constrained, we simply set the yaw to be the same as the direction of velocity to facilitate obstacle avoidance (i.e., the frontlooking camera will always look in the moving direction).

Next, we need to select the best trajectory c_{best} for the given task. We choose the following criterion:

$$c_{best} = \arg\max_{c \in C} (1 - p_{col})(R_{\text{perc}} + R_{\text{goal}}) + p_{\text{col}} R_{\text{col}}, \tag{6}$$

where p_{col} is the probability to collide with obstacles (i.e., the dense map). R_{col}, R_{perc} and R_{goal} are the rewards related to collision risk, perception quality and goal progress respectively, which we will describe now. For simplicity, we will drop the subscript in (5) and refer to the trajectory to be evaluated as c.

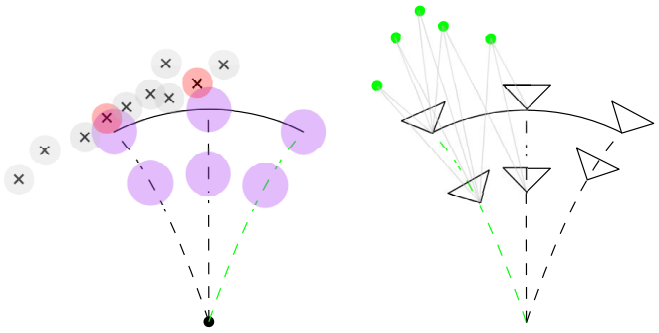

Fig. 6: The evaluation of collision probability (left) and perception quality (right). Left: Crosses are points from the dense map, and dashed curves are candidate trajectories. To calculate the collision probability of a certain position on a trajectory, we find the nearest point (crosses with red circles) in the dense map and calculate the collision probability using a multivariate Gaussian distribution. Right: Solid green dots are the landmarks from the active map. For all the sampled poses along a trajectory, we collect the visible landmarks and construct the information matrix to evaluate the perception quality. The best trajectories in both cases are colored as green.

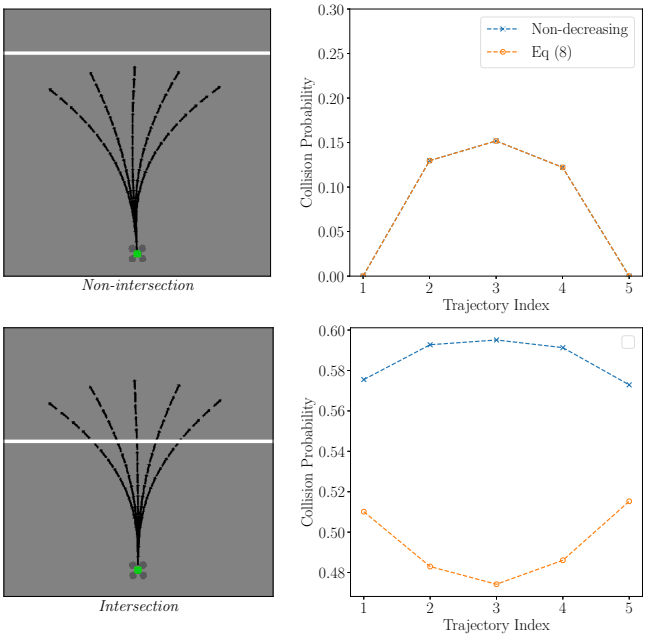

Fig. 7: The calculation of collision probability. The left column shows the simulated trajectories (black arrow) and obstacles (horizontal white band), and the right shows the collision probabilities calculated using different methods. The trajectory index increases from left to right.

C. Collision Probability

We use a similar method to calculate the probability of collision as [8], which is illustrated in the Fig. 6. In particular, for each sampled position $\mathbf{p}_j$, we find the nearest point in the dense map M_D:

$$\mathbf{d}_j = \arg\min_{\mathbf{d} \in M_D} \|\mathbf{d} - \mathbf{p}_j\| \tag{7}$$

Then we calculate the collision probability of $\mathbf{p}_j$ using a multivariate Guassian distribution

$$p_j = V \times \frac{1}{\sqrt{2\pi\Sigma}} \exp[-\frac{1}{2}(\mathbf{d}_j - \mathbf{p}_j)^\top \Sigma^{-1} (\mathbf{d}_j - \mathbf{p}_j)], \tag{8}$$

where V is the volume of the safety sphere we want to keep around the quadrotor. In (8), $\Sigma = (\sigma_d + \sigma_p)\mathrm{I}_{3\times 3}$, where σ_d and σ_p are the uncertainty of the map point and the position on the trajectory respectively. Then the collision probability for the trajectory c is

$$p_{\text{col}} = 1 - \prod_{j=1}^{K}(1 - p_j), \tag{9}$$

and the collision reward is

$$R_{\text{col}} = k_{\text{col}}, \tag{10}$$

where k_{col} is a negative constant indicating how much we need to penalize collision risk.

When the above method is used to calculate the collision probability, one drawback, for example, is that when the trajectory reaches beyond a planar obstacle, the collision probability of the positions behind the obstacle will be small, which is not realistic. Strictly speaking, we have no information about the space behind the obstacle. However, it is often true that the space behind is also occupied. Therefore, different from (8) used by [8], we calculate the collision probability of $\mathbf{p}_j$ as

$$p_j = \begin{cases} \text{eq (8)} & p_j \geq p_{j-1} \\ p_{j-1} & p_j < p_{j-1} \end{cases} \tag{11}$$

Basically speaking, we assume the collision probabilities for positions along the trajectory is non-decreasing. To demonstrate the effect of (11), we compare the collision probabilities using both methods when a quadrotor approaches a planar obstacle, as shown in Fig. 7. In the first row, we can see that when the candidate trajectories do not intersect with the obstacle, (8) and (11) have the same result. When there is intersection, as shown in the second row, the result from (8) indicates that the central trajectory is less likely to collide with the obstacle. By contrast, our method assigns a higher collision probability to the central trajectories. One may argue that both methods are not correct anyway because no information is known about the space behind the obstacle. However, when the calculated collision probabilities are used to select the best trajectory, our method is still advantageous by preferring the trajectories away from the center.

D. Perception Quality

Given a trajectory c and the active map $M_A = \{\mathbf{l}_k\}_{k=1}^{K}$, we need to quantify the pose estimation error if the quadrotor follows the trajectory. The smaller the pose estimation error is, the larger the R_{perc} in (6) will be. To this end, we first simulate the observations for the sampled poses $\{\mathrm{T}_j\}_{j=1}^{J}$ and then construct a least squares problem and evaluate the estimation error from the information matrix of the least squares problem.

For each pose T_j, we can find the visible landmarks in M_A and denote their indexes collectively as O_j, as illustrated in Fig. 6. Then if the quadrotor moves to T_j, its pose is usually estimated by solving the following least squares problem

$$\mathrm{T}_j^* = \arg\min_{\mathrm{T}} \sum_{k\in O_j} \|\text{proj}(\mathrm{T}_{cb}\mathrm{T}^{-1}\mathbf{l}_k) - \tilde{\mathbf{u}}_{jk}\|^2, \tag{12}$$

where T_{cb} is the relative transformation from the camera frame c to the body frame b, $\text{proj}(\cdot)$ is the projection function of the camera, and $\tilde{\mathbf{u}}_{jk}$ is the noisy observation of the kth landmark. (12) is usually solved using iterative optimization methods such as the Gauss-Newton algorithm. To put it formally, the following problem is considered instead

$$\xi^* = \arg\min_{\xi} \sum_{k\in O_j} \|\text{proj}(\mathrm{T}_{cb}(\mathrm{T}_j\text{Exp}(\xi))^{-1}\mathbf{l}_k) - \tilde{\mathbf{u}}_{jk}\|^2 \tag{13}$$

and T_j is updated each iteration as $\mathrm{T}_j \leftarrow \mathrm{T}_j\text{Exp}(\xi)$. ξ is the element in $\mathfrak{se}(3)$, and $\text{Exp}(\cdot)$ maps the element in $\mathfrak{se}(3)$ to $SE(3)$. (13) is solved by linearizing around the current estimate of T_j:

$$\xi^* = \arg\min_{\xi} \sum_{k\in O_j} \|\mathbf{u}_{jk} - \tilde{\mathbf{u}}_{jk} + \mathrm{J}_{\xi}^{jk}\xi\|^2, \tag{14}$$

where $\mathbf{u}_{jk} = \text{proj}(\mathrm{T}_{cb}\mathrm{T}_j^{-1}\mathbf{l}_k)$ and $\mathrm{J}_{\xi}^{jk} = \frac{\partial \mathbf{u}_{jk}}{\partial \xi}$. (14) can be solved in a closed form. The covariance of the estimated parameter is [25]

$$\Sigma_j = (\mathrm{J}_j^{\top}\Sigma_{\mathbf{u}}^{-1}\mathrm{J}_j)^{-1} = (\mathrm{H}_j)^{-1}, \tag{15}$$

where J_j is stacked up by J_{ξ}^{jk}, and $\Sigma_{\mathbf{u}}$ is the covariance matrix of $\{\mathbf{u}_{jk}\}_{k\in O_j}$, which is usually a diagonal matrix with the same value $\sigma_{\mathbf{u}}$ on the diagonal when the observations are independent and have the same noise level. H_j is the *information matrix*.

We do not only try to minimize the pose estimation error of one pose on the trajectory but all the sampled poses. The intuition is that accurate pose estimation along the trajectory can help better triangulate new landmarks, which is beneficial for the pose estimation afterwards. Because the active map M_A contains only well estimated points and is fixed, the poses $\{\mathrm{T}_j\}_{j=1}^{J}$ are actually independent (i.e., conditionally independent, conditioned on the landmark positions). Therefore we can write the whole information matrix as

$$\mathrm{H} = \begin{bmatrix} \mathrm{H}_1 & 0 & \dots & 0 \\ 0 & \mathrm{H}_2 & \dots & 0 \\ 0 & \dots & \dots & 0 \\ 0 & \dots & 0 & \mathrm{H}_J \end{bmatrix}, \tag{16}$$

and the full covariance is $\Sigma = \mathrm{H}^{-1}$, which is also a diagonal block matrix.

Using (16), we calculate the reward using an analog form of the *D-opt* criterion [26]

$$R_{\text{perc}} = k_{\text{perc}} \exp(\log([\det(\mathrm{H})]^{\frac{1}{6J}})), \tag{17}$$

where k_{perc} is a parameter determining the weight for the perception quality.

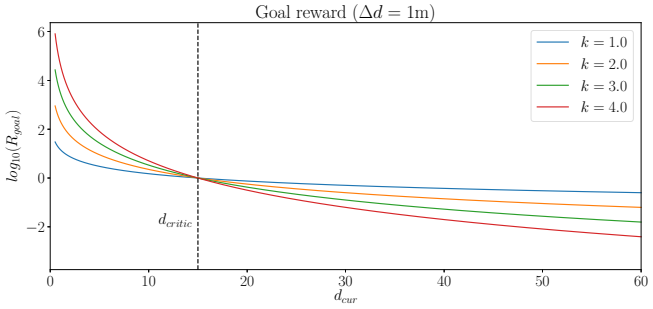

Fig. 8: Goal reward function when $d_{\text{critic}} = 15\text{m}, \Delta d = 1\text{m}$. For the same Δd, the goal reward is higher when the quadrotor is close to the goal. When the distance to goal is smaller than d_{critic}, the goal reward increases more rapidly.

E. Goal Progress

One straightforward way to evaluate how much a trajectory approaches a destination is to use the distance decrease Δd from the start point of the trajectory to the end point. However, in practice we find this approach does not generalize well to different situations. One reason is that the evaluation of the goal progress should be related to the current distance d_{cur} to the goal. For example, reducing the distance from 2m to 1m should be better rewarded than from 100m to 99m. Therefore, we define the goal reward as

$$R_{\text{goal}} = k_{\text{goal}} \Delta d \times (\frac{d_{\text{critic}}}{d_{\text{cur}}})^k, \tag{18}$$

where d_{critic} is a parameter that controls the size of strong attraction area near the goal, and k controls the strength of the distance-dependent property of the weight. An example of the goal reward at different distances is shown in Fig. 8.

To summarize, we combine the collision probability (10), perception quality (17) and goal progress (18) and select the best trajectory defined by (6). In the next section, we will demonstrate the effectiveness of the proposed system in both simulation and real-world experiments.

V. Experiments

To prove the effectiveness of the proposed method, we performed experiments in both simulation and real-world environments. The parameters related to calculating the total reward (6) are shown in Table. I.

TABLE I: Parameters for simulation and real-world experiments. They share the same parameters except d_{critic} and l, which depend on the scene dimensions.

	k_{col}	k_{perc}	k_{goal}	d_{critic}	k	l (meter)
Simulation	-10000	1.5	10	15	3	5.0
Real-world	-	-	-	1	-	2.5

A. Simulation

We tested in different scenarios in simulation to show that our system does not overfit a particular environment. Statistical results are summarized from multiple runs.

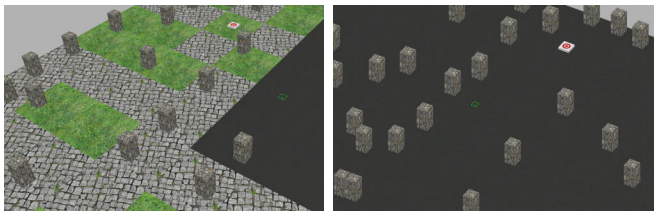

Fig. 9: A close look of the two scenes (*L shape* and *obstacles*) used for simulation.

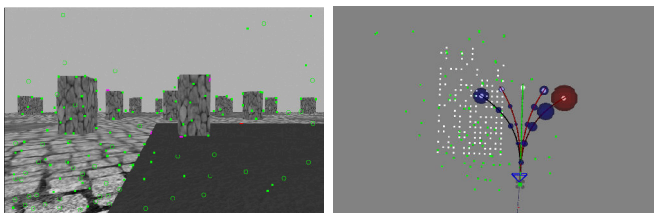

Fig. 10: An instant of the perception aware receding horizon navigation in execution. On the left is the image from the forward-looking camera, where the green solid dots and circles are landmarks and seeds in SVO respectively. On the right is the visualization of the trajectory generation and evaluation. Five trajectories are generated in this case. The blue/red spheres on each trajectory denotes the covariance at the sampled poses, where red means the corresponding pose is not constrained (the information matrix is singular). Our method correctly identifies the rightmost trajectory, which steers the quadrotor towards textureless region, is the worst in terms of perception quality.

We utilized the RotorS simulator [27]. Examples of the scenes used are shown in Fig. 9 and their dimensions reported in Table. II. The simulated MAV is a AscTec Hummingbird quadrotor with a forward-looking camera. In each scenario, we started the quadrotor at slightly different beginning positions and commanded it to fly to a given destination for 10 times. In our simulation, the quadrotor rarely crashed into obstacles, even with large state estimation error, which is the advantage of using only local information for motion planning. Therefore we define the criteria for success using the state estimation error. During each run, once the state estimation diverged from the groundtruth over 5 meters, we terminated the execution and reported the trial as a failure. Once the distance of the state estimation to the given destination is smaller than 3 meters, we reported the trial as a success. To eliminate the inaccuracy induced by the initialization of visual odometry, we initialized SVO using the groundtruth from the simulator.

We compared the performance of our method and a purely-reactive navigation method (i.e., without R_{perc} in (6)). Fig. 12 shows the trajectories overlayed on the simulated scenes and the state estimation error over the traveled distance. Fig. 11 shows the final state estimation error when the simulation was terminated. Table. II reported the number of successful trials for each scene.

The first scene *L shape* consists of areas with strong texture (grass and stone) and weak texture (dark area in the bottom left part). We can observe that the purely-reactive navigation method commanded the quadrotor to fly directly to the goal but the trajectory passed visually degraded part, resulting in large state estimation error. By contrast,

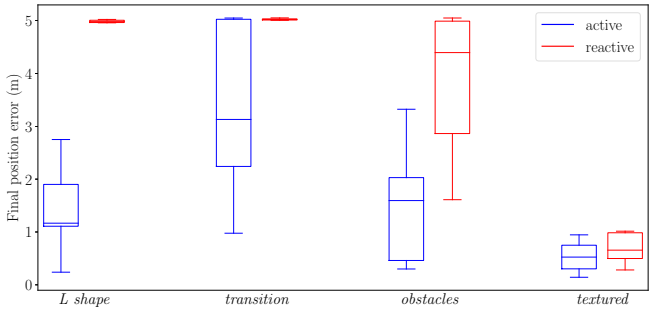

Fig. 11: Position error when the each run is terminated (either the goal is reached or the state estimation is 5 meters away from the groundtruth.)

our method prevented the quadrotor from entering the less textured area and was able to maintain a reasonable state estimation. Similarly, the second scene *transition* contains a visually degraded area in the middle, which lies between the start point and the destination. While our method was able to steer the quadrotor away from that area and reached the destination, reactive navigation was not able to finish the task successfully. An example of our system in execution is shown in Fig. 10.

Our method outperforms the reactive navigation in the first two scenes by a large margin. The main reason for the drastic difference is that these two scenes contain two obviously separated areas with good and poor texture (bottom left and the rest for *L shape*, middle and surround for *transition*), and the visually degraded area lies between the start position and the goal. Therefore, reactive navigation will inevitably enter the area with poor texture, resulting in poor performance. To illustrate the usefulness of our method in a more general setup, we tested on two more scenes. In the third scene *obstacles*, the ground has little texture and most of the visual information comes from the obstacles. In this scene, we can see that the trajectories of our method and reactive navigation are less different than the first two scenes, but the state estimation error of our method is still obviously smaller, as shown in Fig. 12 and Fig. 11. If the trajectories are inspected closely, we can see that our method steered the quadrotor to move closer to obstacles compared to the reactive one. This implies that obstacles with visual features are both repellers and attractors in our method: getting close to such obstacles will decrease state estimation uncertainty but also increase the collision risk. The fourth scene *textured* is fully textured without obviously visually degraded part. Both reactive navigation and our method performed well in this scene, but we can still observe slightly better performance from our method.

TABLE II: Successful runs out of 10 trials for different scenes.

Scene	Dimension (m)	Reactive	Ours
L shape	100×100	1	**5**
transition	100×100	0	**5**
obstacles	80×80	6	**9**
textured	60×60	9	**10**

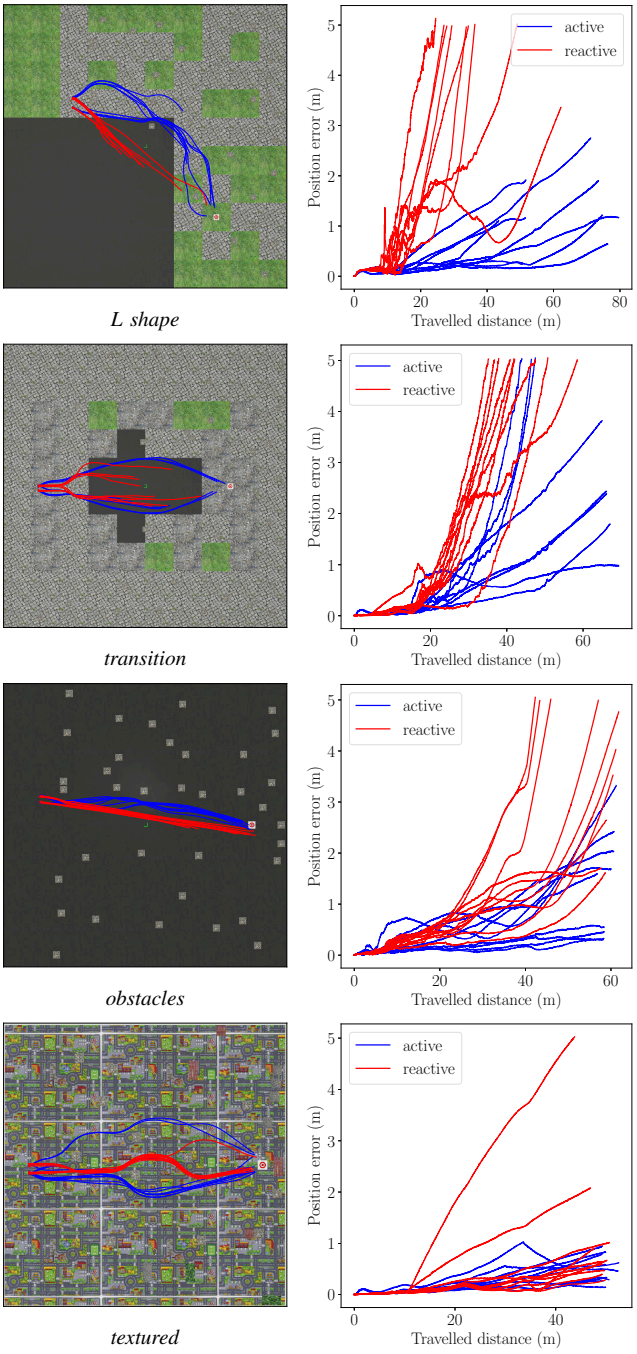

Fig. 12: Trajectory overlayed on the simulation environment (left column) and position error with respect to the traveled distance (right column) for different scenes. Each navigation strategy is executed 10 times on the same scene with slightly different start positions.

B. Real-world Experiments

The quadrotor used in real-world experiments is equipped with a forward looking MatrixVision mvBlueFOX-MLC200w monochrome global shutter camera. The onboard computer is ODROID-XU4. It also carries an PX4FMU autopilot board from Pixhawk that includes an IMU. In our experiments, the monocular state estimation and mapping system was done on-board, while the trajectory generation and evaluation was computed on a laptop at 50Hz.

Fig. 13: Real world scene used to test our method. The start point is at the bottom left and the goal top right. The plot on the right shows the topviews of the trajectories of the quadrotor when it was controlled by our method. The trajectories are "attracted" by the textured area instead of going directly from the start position to the goal.

We tested our method in a scene that contains both texture-rich and textureless areas. A photo of the scene and several example trajectories from our method are shown in Fig. 13. Similar to the results in simulation, our system was able to command the quadrotor to follow a more informative path to reach the goal.

VI. Conclusions and Future Work

In this work, we proposed to integrate active perception in a receding horizon setting for a goal reaching task. In particular, we designed a perception-aware receding horizon navigation system using a single forward looking camera for MAVs. We used a monocular visual odometry SVO and a dense reconstruction algorithm REMODE to provide the essential information for navigation. Using the information, we generated a library of trajectories and evaluated them in terms of collision probability, perception quality and goal progress to select the next motion for MAVs, which naturally combines different performance metrics. We demonstrated the effectiveness of our system by extensive simulation and real-world experiments: in addition of the capability of avoiding obstacles, our perception-aware receding horizon navigation system is able to select motion to favor the state estimation accuracy, which is especially advantageous in environments with visually degraded regions.

Future work would include further real-world validation in different environments to better understand the completeness and failure cases of the method. Generating informative motion primitives is also of interest.

References

[1] A. J. Davison and R. M. Murray, "Simultaneous localization and map-building using active vision," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 7, 2002.

[2] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *2016 ICRA*, May 2016, pp. 1476–1483.

[3] L. Matthies, R. Brockers, Y. Kuwata, and S. Weiss, "Stereo vision-based obstacle avoidance for micro air vehicles using disparity space," in *2014 ICRA*, May 2014, pp. 3242–3249.

[4] B. Landry, R. Deits, P. R. Florence, and R. Tedrake, "Aggressive quadrotor flight through cluttered environments using mixed integer programming," in *2016 ICRA*, May 2016, pp. 1469–1475.

[5] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, July 2017.

[6] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, *et al.*, "Fast, autonomous flight in gps-denied and cluttered environments," *Journal of Field Robotics*, vol. 35, no. 1, pp. 101–120, 2018.

[7] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification," in *ICRA*, 2013, pp. 3480–3486.

[8] P. Florence, C. John, and R. Tedrake, "Integrated perception and control at high speed," in *WAFR: Workshop on the Algorithmic Foundations of Robotics*, 2016.

[9] C. Mostegel, A. Wendel, and H. Bischof, "Active monocular localization: Towards autonomous monocular exploration for multirotor mavs," in *2014 ICRA*, May 2014, pp. 3848–3855.

[10] S. A. Sadat, K. Chutskoff, D. Jungic, J. Wawerla, and R. Vaughan, "Feature-rich path planning for robust navigation of mavs with mono-slam," in *2014 ICRA*, May 2014, pp. 3870–3875.

[11] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.*, vol. 30, no. 7, pp. 846–894, June 2011.

[12] I. Alzugaray, L. Teixeira, and M. Chli, "Short-term uav path-planning with monocular-inertial slam in the loop," in *2017 ICRA*, May 2017, pp. 2739–2746.

[13] G. Costante, C. Forster, J. Delmerico, P. Valigi, and D. Scaramuzza, "Exploiting photometric information for planning under uncertainty," *Springer Tracts in Advanced Robotics (International Symposium on Robotic Research)*, 2017.

[14] K. Hausman, J. Preiss, G. S. Sukhatme, and S. Weiss, "Observability-aware trajectory optimization for self-calibration with application to uavs," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1770–1777, July 2017.

[15] J. A. Preiss, K. Hausman, G. S. Sukhatme, and S. Weiss, "Trajectory optimization for self-calibration and navigation," *Robotics: Science and Systems XIII*, 2017.

[16] C. Forster, M. Pizzoli, and D. Scaramuzza, "Appearance-based active, monocular, dense depth estimation for micro aerial vehicles," in *Robotics: Science and Systems (RSS)*, 2014.

[17] S. Isler, R. Sabzevari, J. Delmerico, and D. Scaramuzza, "An information gain formulation for active volumetric 3d reconstruction," in *2016 ICRA*, May 2016, pp. 3477–3484.

[18] Z. Rong and N. Michael, "Detection and prediction of near-term state estimation degradation via online nonlinear observability analysis," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Oct 2016, pp. 28–33.

[19] C. Papachristos, S. Khattak, and K. Alexis, "Uncertainty-aware receding horizon exploration and mapping using aerial robots," in *2017 ICRA*, May 2017, pp. 4568–4575.

[20] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera systems," *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, 2017.

[21] S. Lynen, M. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to MAV navigation," in *IROS*, 2013.

[22] M. Pizzoli, C. Forster, and D. Scaramuzza, "REMODE: Probabilistic, monocular dense reconstruction in real time," in *ICRA*, 2014, pp. 2609–2616.

[23] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: an efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, Apr 2013.

[24] C. Forster, M. Faessler, F. Fontana, M. Werlberger, and D. Scaramuzza, "Continuous on-board monocular-vision–based aerial elevation mapping for quadrotor landing," in *ICRA*, 2015, pp. 111–118.

[25] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003, second Edition.

[26] H. Carrillo, I. Reid, and J. A. Castellanos, "On the comparison of uncertainty criteria for active slam," in *2012 ICRA*, May 2012, pp. 2080–2087.

[27] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Robot operating system (ros)," *Studies Comp.Intelligence Volume Number:625*, vol. The Complete Reference (Volume 1), no. 978-3-319-26052-5, p. Chapter 23, 2016, iSBN:978-3-319-26052-5.