Perception-aware Path Planning

Gabriele Costante   Christian Forster   Jeffrey Delmerico   Paolo Valigi   Davide Scaramuzza

Abstract—In this paper, we give a double twist to the problem of planning under uncertainty. State-of-the-art planners seek to minimize the localization uncertainty by only considering the geometric structure of the scene. In this paper, we argue that motion planning for *vision-controlled* robots should be *perception aware* in that the robot should also favor texture-rich areas to minimize the localization uncertainty during a goal-reaching task. Thus, we describe how to optimally incorporate the *photometric information* (i.e., texture) of the scene, in addition to the the geometric one, to compute the uncertainty of vision-based localization during path planning. To avoid the caveats of feature-based localization systems (i.e., dependence on feature type and user-defined thresholds), we use *dense, direct methods*. This allows us to compute the localization uncertainty directly from the intensity values of every pixel in the image. We also describe how to compute trajectories online, considering also scenarios with no prior knowledge about the map. The proposed framework is general and can easily be adapted to different robotic platforms and scenarios. The effectiveness of our approach is demonstrated with extensive experiments in both simulated and real-world environments using a vision-controlled micro aerial vehicle.

I. INTRODUCTION

Most of the literature on robot vision has focused on the problem of *passive* localization and mapping from a predefined set of view points—also known as visual odometry or SLAM [1]—where impressive results have been demonstrated over the last decade [2, 3, 4, 5]. Minor work has instead tackled the problem of how to *actively* control the perception pipeline in order to improve the performance of a given task [6, 7].

In this paper, we address the problem of how to optimally leverage vision in a goal-reaching task to select trajectories with minimum localization accuracy. State-of-the-art path planners seek to minimize the localization uncertainty by only considering the geometric structure of the scene. However, for vision-controlled robots it is crucial to also consider the photometric appearance (i.e., texture) of the environment when designing reliable trajectories (cf. Figure 1).

The basic observation is that the uncertainty of vision-based localization is strongly affected by the photometric appearance of the observed scene (cf. Figure 2). Thus, highly-textured areas should be preferred to locations with poor photometric information when planning reliable trajectories (i.e., with low localization uncertainty). Driven by this observation, we aim to answer the following question: *What is the trajectory that minimizes the camera pose-estimation uncertainty in a robot-navigation task?* In practice, the *best* trajectory depends on different factors: (i) the current robot pose and uncertainty, (ii) the geometry of the scene, and (iii) the photometric appearance

G. Costante and P. Valigi are with the Department of Engineering, University of Perugia, Italy.

C. Forster, J. Delmerico, and D. Scaramuzza are with the Robotics and Perception Group, University of Zurich, Switzerland.
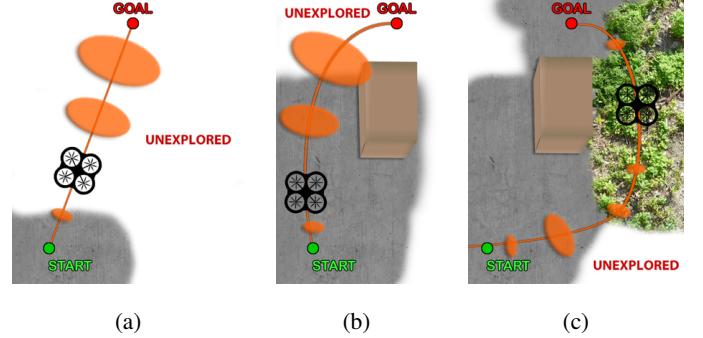
Fig. 1: Online perception-aware path planning: An initial plan is computed without prior knowledge about the environment (a). The plan is then updated as new obstacles (b) or new textured areas (c) are discovered. Although the new trajectory (c) is longer than the one in (b), it contains more *photometric* information and, thus, is optimal with respect to the pose localization uncertainty.
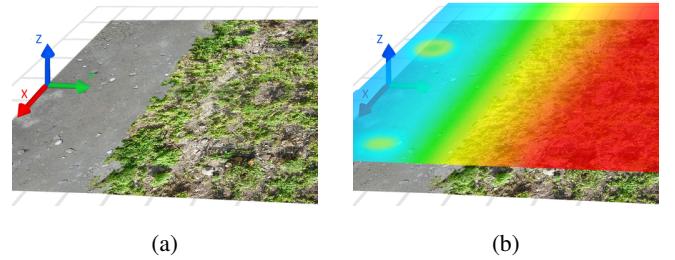

Fig. 2: (a) A scene and (b) its localization uncertainty (notably, the trace of the covariance matrix) for a downward-looking camera at a given height. The localization uncertainty is visualized as a heat-map (blue means high uncertainty, red means low).

of the scene. Based on the these considerations, we describe how to incorporate the photometric information, in addition to the the geometric one, to compute the uncertainty of vision-based localization during path planning. The best trajectory can then be computed as a function of the robot's current pose and the expected pose-uncertainty reduction due to the predicted 3D structure and photometric appearance of the scene (see Figure 1).

Since we want to handle scenarios with no prior knowledge about the map, we also present an online adaptation of the proposed framework. In particular, we update the plan as the robot explores the scene, adapting the perception-aware trajectory as new photometric information becomes available.

II. RELATED WORK

A. Planning in Information Space

The selection of trajectories that minimize the localization uncertainty is often referred to as "Planning under Uncertainty" or "Planning in Information Space". This problem

has generally been solved with Partially Observable Markov Decision Processes (POMDPs) or through graph-search in the belief space [8]. While these approaches are well-established, in general their computational complexity grows exponentially in the number of possible actions and observations. To overcome this issues, Rapidly-exploring Random Tree (RRT*) [9] were introduced to perform fast trajectory computation and guarantee asymptotic optimality. Furthermore, Rapidly-exploring Random Belief Trees (RRBTs) were proposed by [10] as an extension of the RRT* framework to take into account the pose uncertainty. However, while the RRBTs are well-suited for energy minimization tasks, in this work, we specifically focus on selecting trajectories that maximize the visual information without considering robot dynamics or control efforts. Thus, we choose to extend the RRT* framework to take into account also the pose uncertainty when computing optimal trajectories.

B. Active Perception

When perception is incorporated into the path planning process, the problem of selecting optimal viewpoints to maximize the performance of a given task is referred to as *active perception* [6, 11, 12, 13, 14]. One of the goals of active perception is *active localization*, which seeks to compute control actions and trajectories that minimize the pose estimation uncertainty. Most active localization works have been in the context of robot SLAM or exploration. Depending on the sensor used, they can be classified into range-based [15, 16, 17] or vision-based [18, 19, 20, 21, 22].

While range sensors only perceive the geometric structure of the environment, vision sensors are more informative because they can capture both the geometry and appearance of a scene. Davison and Murray [18] were the first to take into account the effects of actions during visual SLAM. The goal was to select a fixation-point of a moving stereo head attached to a mobile robot in order to minimize the motion drift along a predefined trajectory. Vidal Calleja et al. [19] demonstrated an active feature-based visual SLAM framework that provides realtime user-feedback to minimize both map and camera pose uncertainty. Bryson and Sukkarieh [23] demonstrated a similar visual and inertial EKF-SLAM formulation for active control of flying vehicles. The goal was to cover a predefined area with a camera while maintaining an accurate estimation of both the map and the vehicle state. Extensive simulation results were provided of a MAV that is restricted to fly on a plane. Mostegel et al. [20] proposed a set of criteria to estimate the influence of camera motion on the stability of visual localization for MAVs.

The minimization of the pose covariance in vision-based path-planning systems was addressed in [21] and [22]. Achtelik et al. [22] used RRBTs to evaluate offline multiple path hypotheses in a known map and select paths with minimum pose uncertainty while at the same time considering the vehicle dynamics. They computed the pose covariance directly from bundle adjustment, by minimizing the reprojection errors of the 3D map points across all images. The approach was demonstrated on a MAV. Sadat et al. [21] proposed a strategy

to plan trajectories for MAVs, which prefers paths rich of visual features. A viewpoint score based on the number of observed features was used to measure the quality of localization. The system used RRT* to iteratively re-plan as the robot explored the environment. As a fixed part of the previous plan is executed, RRT* is recomputed from scratch.

C. Feature-based vs Dense, Direct Methods

All vision-based works previously mentioned represent the scene as a set of *sparse* 3D landmarks corresponding to discriminative features in the observed images (e.g., SIFT, SURF, etc.) and estimate structure and motion through reprojection-error minimization. A reason for the success of these methods is the availability of robust feature detectors and descriptors that allow matching images with large disparity. The disadvantage of feature-based approaches is the dependence on the feature type, the reliance on numerous detection and matching thresholds, the necessity for robust estimation techniques to deal with incorrect correspondences (e.g., RANSAC), and the fact that most feature detectors are optimized for speed rather than precision.

The alternative to feature-based methods is to use *dense, direct methods* [24]. Direct methods have the advantage that they estimate structure and motion directly by minimizing an error measure (called *photometric error*) that is based on images pixel-level intensities. The local intensity gradient magnitude and direction is used in the optimization compared to feature-based methods that only consider the distance to a feature-location. Pixel correspondence is given directly by the geometry of the problem, eliminating the need for robust data association techniques. Direct methods are said *dense* if they exploit the visual information even from areas where gradients are small (i.e., not just edges). Dense, direct methods have been shown to outperform feature-based methods in terms of robustness in scenes with little texture [25] or in the case of camera defocus and motion blur [26, 27]. Using dense, direct methods, the 6-DoF pose of a camera can be recovered by *dense image-to-model alignment*, which is the process of aligning the observed image to a view synthesized from the estimated 3D map through photometric error minimization.

The first approach taking advantage of dense, direct methods in the context of active perception was proposed by Forster et al. [28]. However, the task was specified in terms of maximizing the quality of the map (i.e., minimizing the map uncertainty). Thus, the robot localization uncertainty was not considered. Additionally, path planning from a start to a goal point was not investigated. Conversely, in this paper we are interested in computing trajectories towards a predefined goal while minimizing the robot pose uncertainty along the path. In contrast to previous works based on sparse features, we use dense, direct methods.

D. Contributions

Our contributions are:

- An *online perception-aware path planning* framework that computes the best path towards a predefined goal

through the exploitation of both the geometric *and photometric* information (i.e., texture) of the scene. To the best of our knowledge, this is the first attempt to use the photometric appearance in addition to the geometric 3D structure for planning under uncertainty.

- We use *dense, direct methods* to compute the photometric information gain directly from the intensity values of every pixel in the image. This avoids the caveats of feature-based localization systems, such as the dependence on the type of feature detector and descriptor and the reliance on user-defined thresholds for detection and matching.
- We integrate the Lie Group-based propagation proposed in [29] and we extend the Rapidly-exploring Random Tree (RRT*) [9] framework to take into account the pose uncertainty when computing trajectories.
- We implement and demonstrate the effectiveness of our approach on an actual vision-based quadrotor performing vision-based localization, dense 3D reconstruction, and online perception-aware planning.

This paper extends the work presented in [30] with more experimental results and technical details.

E. Outline

The outline of the paper is as follows: in Section III, we introduce the Lie-Group–based propagation framework and describe how the pose uncertainties are propagated along the trajectory. Section III-C describes the dense image-to-model alignment strategy to compute the photometric information gain in terms of the scene texture. In Section IV, we adapt the RRT* framework to generate trajectories that minimize the camera pose uncertainty given the photometric information computed along the path. In Section V, we present the experimental evaluation. Finally, in Section VI we draw conclusions and highlight possible future improvements.

III. LIE GROUP BASED UNCERTAINTY PROPAGATION

Different trajectories lead to different evolutions of pose covariance. For this reason, it is crucial to predict how the pose uncertainty will be affected given a candidate route. To achieve this, we need a state representation to propagate the pose estimate, together with its uncertainty, when executing a predefined trajectory.

When choosing a state representation, most challenges arise because the rotation parametrizations have either singularities or constraints. This is related to the fact that rotation variables are not vectors but members of a non-commutative group, *i.e.*, the Lie group $SO(3)$. As a consequence, using a first-order approximation to propagate the covariance matrix (*e.g.*, in standard EKFs) does not guarantee a good estimate of the uncertainty. Conversely, Monte Carlo techniques are more reliable, but the computational effort required to reach a realistic estimate is often unacceptable. We can achieve both a robust and an efficient representation if we preserve the nature of the rotation matrices, *i.e.*, we represent the robot poses as Lie group members.

A. Associating Uncertainty to Rigid Body Motions

First of all, we provide some assumptions and preliminary notations that we use in our formulations in the following sections.

We represent the pose of the robot as a 6 Degree of Freedom (DoF) transformation matrix $\mathbf{T}$, member of the *special Euclidean group* in $\mathbb{R}^3$, which is defined as follows:

$$SE(3) := \left\{ \mathbf{T} = \left[\begin{array}{cc} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{array} \right] \,\middle|\, \mathbf{C} \in SO(3), \mathbf{r} \in \mathbb{R}^3 \right\}, \quad (1)$$

where

$$SO(3) := \left\{ \mathbf{C} \in \mathbb{R}^{3 \times 3} \,\middle|\, \mathbf{C}\mathbf{C}^T = \mathbf{1}, \det \mathbf{C} = 1 \right\} \quad (2)$$

is the special orthogonal group in $\mathbb{R}^3$ (the set of spatial rotations) and $\mathbf{1}$ is the 3×3 identity matrix.

In the following, the *Lie Algebra* associated to the $SE(3)$ Lie Group is referred as $\mathfrak{se}(3)$. To represent the uncertainty of the robot pose, we use the formulation proposed in [29]. We define a random variable for $SE(3)$ members according to:

$$\mathbf{T} := \exp(\boldsymbol{\xi}^\wedge)\bar{\mathbf{T}} \quad (3)$$

In this definition, $\bar{\mathbf{T}}$ is a noise-free value that represents the mean of the pose, while $\boldsymbol{\xi} \in \mathbb{R}^6$ is a small perturbation in the tangent space that we assume to be normally distributed with zero mean and covariance $\boldsymbol{\Sigma}$. We make use of the $\wedge$ operator to map $\boldsymbol{\xi}$ to a member of the Lie algebra $\mathfrak{se}(3)$ using:

$$\boldsymbol{\xi}^\wedge := \left[\begin{array}{c} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{array} \right] = \left[\begin{array}{cc} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}^T & 0 \end{array} \right], \quad (4)$$

where $\boldsymbol{\phi}$ is a member of the *Lie algebra* $\mathfrak{so}(3)$:

$$\boldsymbol{\phi}^\wedge := \left[\begin{array}{c} \phi_1 \\ \phi_2 \\ \phi_3 \end{array} \right]^\wedge = \left[\begin{array}{ccc} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{array} \right] \quad (5)$$
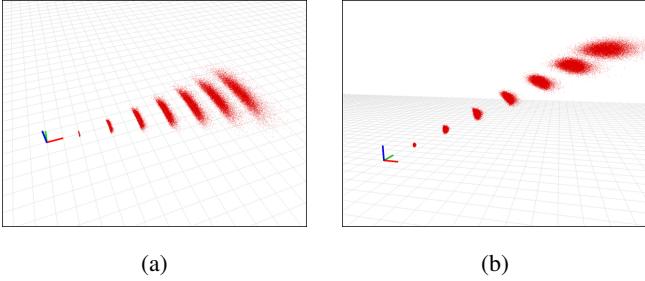
Observe that the operator $\wedge$ is 'overloaded' and can be applied to both 6×1 and 3×1 vectors [29, 31]. They are disambiguated by the context.

Furthermore, we indicate with $\mathbf{T}_{k,w}$ the robot pose at time k relative to the world frame w and with $\mathbf{T}_{k+1,k}$ the transformation between the pose at time k and $k+1$.

B. Pose Propagation

Properly modeling the uncertainty propagation according to the IMU odometry model would require the extension of the robot state vector with the instantaneous velocity. However, to reduce the problem complexity, we assume in the following that the velocity remains constant and, thus, the odometry uncertainty, denoted by $\boldsymbol{\Sigma}_{k+1,k}$, associated to all motions $\mathbf{T}_{k+1,k}$, is fixed.

Given the transformation $\mathbf{T}_{k+1,k}$, we reason about the propagation of the mean and the covariance of the resulting pose $\mathbf{T}_{k+1,w}$. Assuming no correlation between the current pose and the transformation between k and $k+1$, we can

(a)   (b)

Fig. 3: Examples of propagation using the fourth-order Lie group framework. The two columns show two different propagation tests. In 3(a), the covariance is propagated after 100 motions of 1 meter along the x axis, with a motion uncertainty of $\mathbf{\Sigma}_{k+1,k} = \mathbf{diag}(0, 0, 0, 0, 0, 0.03)$. In 3(b), we perform 100 motions $((1.0, 0.0, 0.1)$ meters) starting from the pose $(0, 0, 0, 0, 0, \pi/8)$, and with $\mathbf{\Sigma}_{k+1,k} = \mathbf{diag}(0.01, 0.01, 0.01, 0.001, 0.001, 0.03)$. The covariances are depicted as point clouds, sampling the distributions every 10 motions.

consider $\mathbf{T}_{k,w}$ and $\mathbf{T}_{k+1,k}$ as represented by their means and covariances:

$$\{\bar{\mathbf{T}}_{k,w}, \mathbf{\Sigma}_{k,w}\}, \ \{\bar{\mathbf{T}}_{k+1,k}, \mathbf{\Sigma}_{k+1,k}\}. \tag{6}$$

Combining them, we get

$$\mathbf{T}_{k+1,w} = \mathbf{T}_{k,w} \ \mathbf{T}_{k+1,k}. \tag{7}$$

To compute the mean and the covariance of the compound pose, we use the results from [29]. The mean is

$$\bar{\mathbf{T}}_{k+1,w} = \bar{\mathbf{T}}_{k,w} \ \bar{\mathbf{T}}_{k+1,k}, \tag{8}$$

and the covariance, approximated to fourth order, is

$$\mathbf{\Sigma}_{k+1,w} \simeq \mathbf{\Sigma}_{k,w} + \mathcal{T}\mathbf{\Sigma}_{k+1,k}\mathcal{T}^\top + \mathcal{F} \tag{9}$$

where $\mathcal{T}$ is $Ad(\bar{\mathbf{T}}_{k,w})$, *i.e.*, the adjoint operator for SE(3), and $\mathcal{F}$ encodes the fourth-order terms. Equations (8) and (9), we can propagate the uncertainty along a nominal trajectory. Figure 3 depicts examples of covariance propagations.

C. Measurement Update

In this section, we describe the computation of the photometric information associated to a measurement at a particular viewpoint in order to update the predicted pose uncertainty. The measurement process defines the information that can be obtained from images, hence, we summarize it in the following. In contrast to previous works based on *sparse* keypoints, we use a dense image-to-model alignment approach for the measurement update, which uses the intensity and depth of *every* pixel in the image.

1) Preliminary Notation: At each iteration of the navigation process, we can compute a dense surface model $\mathcal{S} \in \mathbb{R}^3 \times \mathbb{R}^+$ (3D position and grayscale intensity) relative to the explored part of the scene (see Figure 5(a)). The rendered synthetic image is denoted with $\mathbf{I}_s : \Omega_s \subset \mathbb{R}^2 \to \mathbb{R}^+$, where Ω_s is the image domain and $\mathbf{u} = (u, v)^T \in \Omega_s$ are pixel coordinates. Furthermore, we refer to the depthmap $\mathbf{D}_s$, associated to an image $\mathbf{I}_s$, as the matrix containing the distance at every pixel to the surface of the scene:

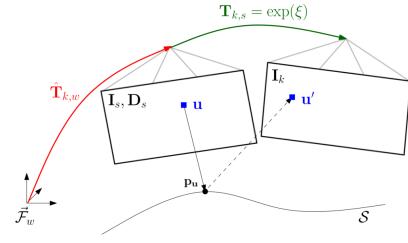$$\mathbf{D}_s : \Omega_s \to \mathbb{R}^+; \ \mathbf{u} \mapsto d_{\mathbf{u}}, \tag{10}$$

Fig. 4: Illustration of the dense image-to-model alignment used in the measurement update. Given an estimate of the pose $\hat{\mathbf{T}}_{k,w}$, we can synthesize an image and depthmap $\{\mathbf{I}_k, \mathbf{D}_k\}$ from the 3D model $\mathcal{S}$. The best update $\boldsymbol{\xi}$ of the pose estimate is computed by minimizing the intensity difference of corresponding pixels $\{\mathbf{u}, \mathbf{u}'\}$.

where $d_{\mathbf{u}}$ is the depth associated to $\mathbf{u}$. Note that, since we need to predict the uncertainty propagation during the planning phase, the actual image at a given location is not available at the beginning. As a consequence, we synthesize the predicted image for each waypoint selected using the reconstructed map and we update the pose uncertainty estimates accordingly.

A 3D point $\mathbf{p} = (x, y, z)^T$ in the camera reference frame is mapped to the corresponding pixel in the image $\mathbf{u}$ through the camera projection model $\pi : \mathbb{R}^3 \to \mathbb{R}^2$

$$\mathbf{u} = \pi(\mathbf{p}). \tag{11}$$

On the other hand, we can recover the 3D point associated to the pixel $\mathbf{u}$ using the inverse projection function π^{-1} and the depth $d_{\mathbf{u}}$:

$$\mathbf{p}_{\mathbf{u}} = \pi^{-1}(\mathbf{u}, d_{\mathbf{u}}). \tag{12}$$

Note that the projection function π is determined by the intrinsic camera parameters that are known from calibration.

Finally, a rigid body transformation $\mathbf{T} \in \mathrm{SE}(3)$ rotates and translates a point $\mathbf{q}$ as follows:

$$\mathbf{q}'(\mathbf{T}) := (\mathbf{1} \,|\, \mathbf{0}) \, \mathbf{T} \, (\mathbf{q}^T, 1)^T. \tag{13}$$

2) Dense Image-to-Model Alignment: Given the dense 3D model of the environment we can synthesize an image and the relative depthmap $\mathbf{I}_s$, $\mathbf{D}_s$ at the estimated pose of the camera $\mathbf{T}_{k,w}$. To refine the current pose estimate $\hat{\mathbf{T}}_{k,w}$ of the frame k with respect to the global world frame w, we use dense image-to-model alignment [26, 32] (see Figure 4). This approach determines the incremental updates $\boldsymbol{\xi}$ to the current pose estimate by minimizing the photometric error between the observed image and the synthetic one. Once converged, this approach also provides the uncertainty of the alignment through evaluation of the *Fisher Information Matrix*, which is used in our approach to select informative trajectories. The image residual $r_{\mathbf{u}}$ for a pixel $\mathbf{u}$ is the difference of the intensity value at pixel $\mathbf{u}$ in the real image acquired at time step k and the intensity value in the synthetic image rendered at the estimated position $\hat{\mathbf{T}}_{k,w}$:

$$r_{\mathbf{u}} = \mathbf{I}_k(\mathbf{u}) - \mathbf{I}_s(\pi(\mathbf{p}'_{\mathbf{u}}(\hat{\mathbf{T}}_{k,w}))) \tag{14}$$

The residual is assumed to be normally distributed $r_{\mathbf{u}} \sim \mathcal{N}(0, \sigma_i^2)$, where σ_i is the standard deviation of the image noise.
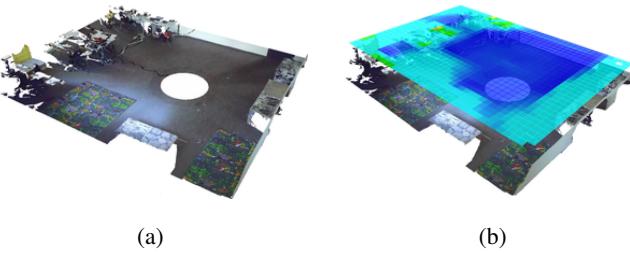
(a)   (b)

Fig. 5: Figure 5(b) shows the information gain related to the scene in 5(a) (Figure 11.a) in the case of fixed height.

The dense image-to-model alignment approach computes the pose $\mathbf{T}_{k,w}$ of the synthetic image $\mathbf{I}_s$, which minimizes the residual with the actual image and, hence, the pose of the robot. Due to the nonlinearity of the problem, we assume that we have an initial guess of the pose $\hat{\mathbf{T}}_{k,w}$ and iteratively compute update steps $\boldsymbol{\xi}^\wedge \in \mathfrak{se}(3)$

$$\hat{\mathbf{T}}_{k,w} \leftarrow \exp(\boldsymbol{\xi}^\wedge)\hat{\mathbf{T}}_{k,w} \qquad (15)$$

that minimize the residual. The update step minimizes the following least-squares problem

$$\boldsymbol{\xi} = \arg\min_{\boldsymbol{\xi}} \sum_{\mathbf{u}\in\Omega_s} \frac{1}{2\sigma_i^2}\Big[\mathbf{I}_k(\mathbf{u}') - \mathbf{I}_s(\pi(\mathbf{p}_\mathbf{u}'(\hat{\mathbf{T}}_{k,w})))\Big]^2 \quad (16)$$

with $\mathbf{p_u}$ given by (12), $\mathbf{p_u'}$ as in (13), and

$$\mathbf{u}' = \pi\big(\mathbf{p}_\mathbf{u}'(\exp(\boldsymbol{\xi}^\wedge))\big). \qquad (17)$$

Addressing the least-squares problem (16) using the Gauss-Newton method leads to the normal equations that can be solved for $\boldsymbol{\xi}$:

$$\mathbf{J}^T\mathbf{J}\boldsymbol{\xi} = -\mathbf{J}^T\mathbf{r}, \qquad (18)$$

where $\mathbf{J}$ and $\mathbf{r}$ are the stacked Jacobian and image residuals of all pixels $\mathbf{u} \in \Omega_s$ respectively.

Specifically, the least-squares minimization requires the computation of the Jacobian of the residual in (16) at each pixel $\mathbf{u}$, which can be written as a function of the gradient in the observed image and the synthetic depthmap[1]:

$$\mathbf{J_u} = \big(\nabla\mathbf{I}_k(\mathbf{u})\big)^T \frac{\partial\pi(\mathbf{b})}{\partial\mathbf{b}}\bigg|_{\mathbf{b}=\mathbf{p}_\mathbf{u}'} \frac{\partial\mathbf{p}_\mathbf{u}'\big(\exp(\boldsymbol{\xi}^\wedge)\big)}{\partial\boldsymbol{\xi}}\bigg|_{\boldsymbol{\xi}=\mathbf{0}} \qquad (19)$$

In this work, for sake of simplicity, we assume depth uncertainty to be zero. However, non-zero values can easily be integrated into our framework.

At the convergence of the optimization, the quantity

$$\boldsymbol{\Lambda}_k = \frac{1}{\sigma_i^2}\mathbf{J}^T\mathbf{J} \qquad (20)$$

is the *Fisher Information Matrix* [34] and its inverse is the covariance matrix $\boldsymbol{\Sigma}_{\mathbf{I}_k}$ of the measurement update.

According to [29], we find the covariance matrix after the measurement update at time k by computing

$$\boldsymbol{\Sigma}_{k,w} \leftarrow \Big(\boldsymbol{\Lambda}_k^{-1} + \mathcal{J}^{-T}\boldsymbol{\Sigma}_{k,w}\mathcal{J}^{-1}\Big)^{-1}, \qquad (21)$$

[1]see Appendix B in [33] for a detailed derivation of the exponential map Jacobian computation.

where the "left-Jacobian $\mathcal{J}$ is a function of how much the measurement update modified the estimate. Note that the information is not only a function of the image gradient but also of the depth at every pixel (see last term in (19)). However, the uncertainty in the orientation is only a function of the texture and independent of the depth.

Solving the dense image-to-model alignment optimization, allows us to estimate the camera pose during execution of the trajectory, by means of iteratively synthesizing synthetic images from the environment model, and to refine the alignment. However, during planning, the location of viewpoints evaluated along a trajectory is known and only the computation of the uncertainty in (21) is relevant. Therefore, the photometric information $\boldsymbol{\Lambda}_k$ can directly be incorporated into the pose covariance with Equation (21).

Given the information matrix in (21), we define the photometric information gain as $\mathrm{tr}(\boldsymbol{\Lambda}_k)$. Figure 5(b) depict the photometric information gain map for the scenario in Figure 5(a).

IV. PLANNING UNDER UNCERTAINTY

Thanks to the propagation framework described in the previous sections, we are able to predict the pose uncertainty after sequences of camera motions. Furthermore, we can update the pose covariance according to the expected photometric information gain computed with the dense image-to-model alignment strategy presented in Section III-C. To compute the optimal path we need to evaluate all possible trajectories and we need to do that *efficiently*. In the following, we describe how the sequence of viewpoints that minimize the localization uncertainty is selected with low complexity. Furthermore, as we do not assume to have any given prior knowledge about the scene, the photometric information of the environment, as well as its 3D geometry, are unknown. Hence, the trajectory that is considered optimal in the beginning will be adapted as new information is gathered by the robot.

As stated in the previous sections, RRT* provides an efficient framework to efficiently compute trajectories. Nevertheless, in its original formulation, the RRT* does not take into account the pose uncertainty.

To benefit from the RRT* advantages and overcome its limitations, we adapt this framework in the next section to our scenario, proposing a cost function that encodes both the distance term and the amount of uncertainty associated with a candidate path.

A. Perception-aware RRT*

At a high level, the rapidly-exploring random trees algorithm explores the state space to compute the optimal path $\mathcal{T}$ from the start location to each point in the space. In particular, the tree is composed of a set of vertices V representing elements of the state space along with their associated pose covariances. Each vertex $v \in V$ has a list of neighboring vertices $v.N$, a state $v.x$, where $x \in SE(3)$, a state covariance $v.\boldsymbol{\Sigma}$, a cost value $v.c$, a unique parent vertex $v.p$, and the photometric information gain $v.\boldsymbol{\Lambda}$ associated to the camera viewpoint at $v.x$. Figure 6 depicts the properties of the tree.

Algorithm 1 Perception-aware RRT*

01: **Init:** Initial vertex $v_0.x = x_{\text{init}}$; $v_0.p = $ root;
    Initial pose covariance $v_0.\Sigma = \Sigma_0$; Initial cost $v_0.c = 0$;
    Initial Vertex set $V = \{v_0\}$; Number of iterations T;
    Collision radius c
02: **for** $t = 1, \dots, T$ **do**
03:    $x_{\text{new}} = $ Sample()
04:    $v_{\text{nst}} = $ Nearest(x_{new})
05:    **if** ObstacleFree(v_{new}, v_{nst}, c)
06:      $\Sigma_t = $ Propagate($v_{\text{nst}}.x$, $v_{\text{nst}}.\Sigma$, $v_{\text{new}}.x$)
07:      $\Sigma_t = $ Update(Σ_t, $v_{\text{new}}.\boldsymbol{\Lambda}$)
08:      $J_{\min} = v_{\text{nst}}.c + (1 - \alpha)\operatorname{tr}(\Sigma_t) + \alpha\operatorname{Dist}(v_{\text{nst}}.x, v_{\text{new}}.x)$
09:      $v_{\min} = v_{\text{nst}}$
10:      $V = V \cup v(x_{\text{new}})$
11:      $V_{\text{neighbors}} = $ Near(V, v_{new})
12:      **for all** $v_{\text{near}} \in V_{\text{neighbors}}$**do**
13:        **if** CollisionFree(v_{near}, v_{new}, c)
14:          $\Sigma_t = $ Propagate($v_{\text{near}}.x$, $v_{\text{near}}.\Sigma$, $v_{\text{new}}.x$)
15:          $\Sigma_t = $ Update(Σ_t, $v_{\text{new}}.\boldsymbol{\Lambda}$)
16:        **if** $v_{\text{near}}.c + (1 - \alpha)\operatorname{tr}(\Sigma_t)$
                $+\alpha\operatorname{Dist}(v_{\text{near}}.x, v_{\text{new}}.x) < J_{\min}$
17:          $J_{\min} = v_{\text{near}}.c + (1 - \alpha)\operatorname{tr}(\Sigma_t) + \alpha\operatorname{Dist}(v_{\text{near}}.x, v_{\text{new}}.x)$
18:          $v_{\text{new}}.\Sigma = \Sigma_t$
19:          $v_{\text{new}}.c = J_{\min}$
20:          $v_{\min} = v_{\text{near}}$
21:        **end if**
22:        **end if**
23:        ConnectVertices($v_{\min}$, v_{new})
24:      **end for**
25:      RewireTree()
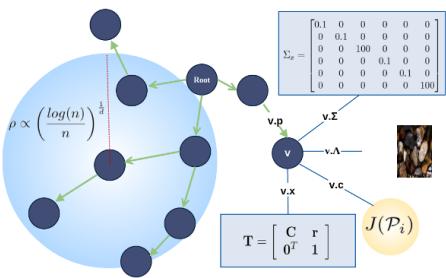26:   **end if**
27: **end for**

Fig. 6: Example of a tree configuration. The green arrows connect different vertices in the tree. Each leaf has a unique path to the root. The blue circle includes all the vertexes affected by the rewire procedure when a new element is sampled and added in the tree. The vertex v is expanded to show the properties of each node.

The graph is incrementally built by sampling new states and connecting them to the existing vertices, propagating the covariances towards the new one. Furthermore, since each location x is associated with a view and a depth map, we can anticipate what the robot will see in a specific position and compute the associated photometric information gain. The algorithm makes use of the dense image-to-model alignment strategy, presented in Section III-C, to compute the predicted information gain and update the pose covariance accordingly.

Each nominal trajectory $\mathcal{T}_i \in \mathcal{P}$ is described by a sequence of N_i waypoints v_j^i, where each of them is a vertex of the tree. To solve the problem of finding the plan that represents the best trade-off between path length and pose estimation

accuracy, we propose a cost function that weighs both the distance between waypoints, and the pose covariances. Among all the candidate paths $\mathcal{P}$, we select the trajectory $\mathcal{T}_i \in \mathcal{P}$ that minimizes the following function:

$$J(\mathcal{T}_i) = \sum_{j=1}^{N_i} \alpha \operatorname{Dist}(v_j^i.x, v_{j-1}^i.x) + (1 - \alpha)\operatorname{tr}(v_j^i.\Sigma) \quad (22)$$

where α is the trade–off factor between path length minimization and information maximization, and $\operatorname{Dist}(\cdot, \cdot)$ computes the distance between the two locations. It should be noticed that, by choosing to minimize the sum of the trace of all the pose covariances, we suggest the algorithm to seek the trajectory that keeps small the camera pose uncertainty along the candidate path. We choose the trace to include the visual information into the cost function following the considerations in [35]. In particular, minimizing the trace of the pose covariance matrix (A-optimality) guarantees that the majority of the state space dimensions is considered (in contrast to the D-optimality), but does not require us to compute all the eigenvalues (E-optimality).

Algorithm 1 describes the proposed Perception-aware RRT*. At each iteration, the algorithm samples a new state from the state space, then it creates and adds the associated vertex to the tree. After that, the vertices near the new one are selected through the function Near(). This function looks for the vertices whose states are within a ball of radius ρ, defined as follows (see [9]):

$$\rho \propto \left(\frac{\log(n)}{n}\right)^{\frac{1}{d}}. \quad (23)$$

In the above equation, the radius depends on the dimension of the state d and on the number of state vertices n. It is important to notice that, before checking for adjacent vertices, the function Nearest() selects the nearest node without checking if it is inside the ball of radius ρ. This is required especially during the first iterations, when the tree is very sparse and, thus, the Near() function can easily return an empty list. The new vertex is then connected along a minimum cost path to one of the neighbors (lines 10-23). In particular, for each element in the neighborhood we first check whether there is a safe connection between the two vertices, *i.e.*, whether there are any collisions along the path. The collision radius c (see Algorithm 1) depends on the geometrical structure of the robot and is provided as an input parameter. Afterwards, the pose uncertainty associated with the current v_{near} vertex is propagated using (9) and updated according to the photometric information gain expected from receiving an image measure when reaching the state x_{new}. Finally, we check whether the overall cost of connecting v_{near} to v_{new} (which represents the cost of the candidate path $\mathcal{T}$ through those waypoints) is smaller than the current minimum, and update it if necessary.

In the final stage of the algorithm, we update the tree connections following the strategy proposed in [9]: the vertices in the neighborhood are visited, updating their parent relationships in the tree if the path through v_{new} is more convenient. This procedure is referred as RewireTree(). During

(a) Standard RRT* - 10 steps

(b) Standard RRT* - 500 steps

(c) Standard RRT* - 2500 steps

(d) Perception-aware RRT* - 10 steps

(e) Perception-aware RRT* - 500 steps
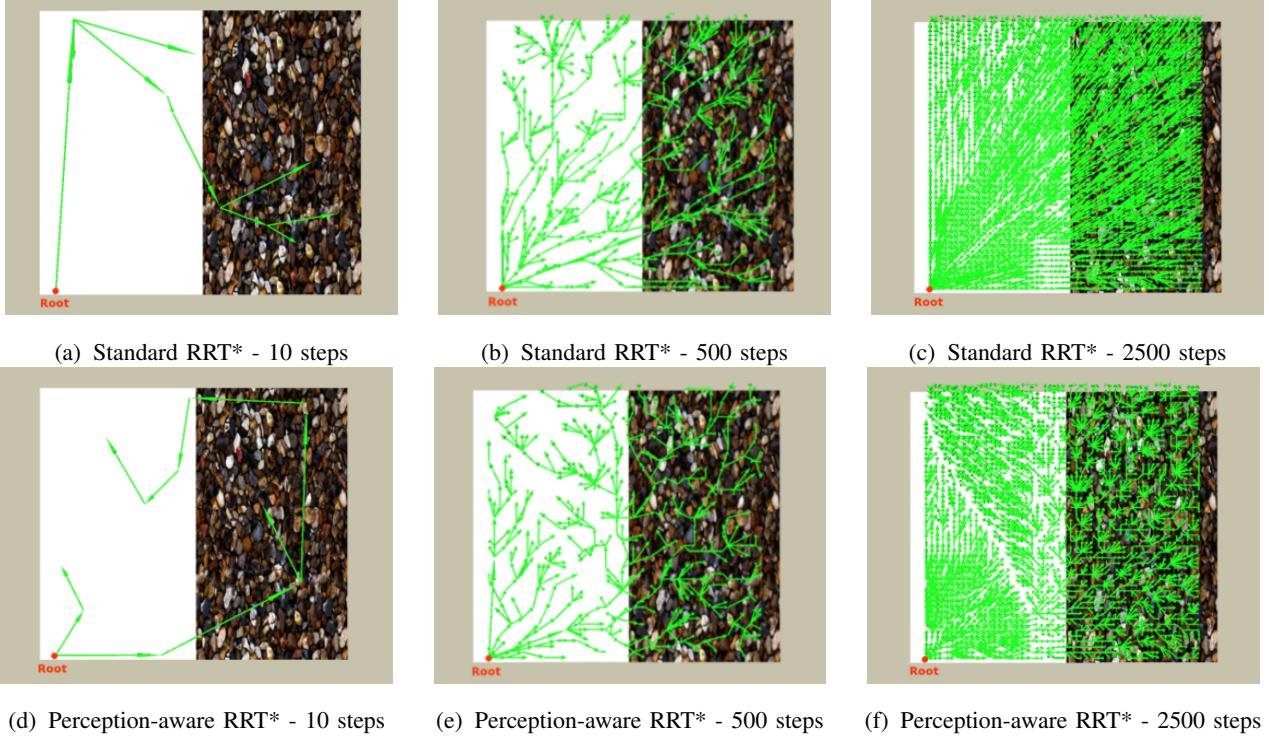
(f) Perception-aware RRT* - 2500 steps

Fig. 7: Evolution of the optimal policy tree after different iterations. From left to right, we plot the state of the tree respectively after 10, 500 and 2500 sampling steps. In 7(a)-7(c) the planner follows the standard RRT* strategy, *i.e.*, the shortest path, without taking into account the information from the vision sensor. By contrast, our framework 7(d)-7(f) computes trajectories that attempt to minimize the pose uncertainty using the photometric information gain.

the RewireTree() procedure we iterate through the subtrees of each v_{near} whose parent relationships has been changed with v_{new} to propagate the updated covariances and maintain the child nodes consistent after rewiring.

The output of the overall procedure is a connected tree, from which we can extract the optimal policy to a generic goal vertex following the parent relationships from the final to the start state. Figure 7 shows the evolution of the tree at different iteration steps and compares the standard RRT* with our perception-aware formulation.

B. Online Perception-aware Planning

Given an initially optimal path, we can now start exploring the environment. When new parts of the scene are revealed, the current trajectory might become non-optimal or even infeasible in case of obstacles. One possibility would be to recompute the tree from scratch after every map update but this would be costly and computationally intractable to have the system integrated into an MAV application. For this reason, we propose to update the planning tree *on-the-fly* by only processing vertices and edges affected by new information. This online update is illustrated in Figure 8 and its fundamental steps are depicted in Algorithm 2.

Consider an initial planning tree as in Figure 8(a), that is grown from a starting point (indicated by a green circle) to a desired end point location (the red circle). Whenever a new obstacle is spotted, the respective edge and the affected subtree get invalidated and regrown (lines 04-06) as in Figure 8(b). Note that the SampleUnexplored() function is now

---

Algorithm 2 Online perception-aware RRT*

01: **while** 1 **do**
02:     UpdateCollisionMap()
03:     UpdatePhotometricInformationMap()
04:     $V_{\text{colliding}}$ = NewCollidingVertices()
05:     InvalidateSubTree($V_{\text{colliding}}$)
06:     Run PerceptionAwareRRT* 1
07:     V_{inf} = UpdatedVertices()
08:     **for all** $v_{\text{inf}} \in V_{\text{inf}}$ **do**
09:         $\Lambda_v = \Lambda_v^{new}$
10:         RewireTree()
11:     **end for**
12: **end while**

---

bounded within the subspace corresponding to the invalidated subtree, which results in a drastically reduced number of iterations compared to fully regrowing the RRT* tree from scratch. The second scenario in Figures 8(d) to 8(f) demonstrates the case of gaining areas with distinctive photometric information. As newly discovered areas provide photometric information, as shown in Figure 8(e), the neighboring vertices are updated by the RewireTree() procedure (lines 07-10 in Algorithm 2). Potentially better connections are considered to form a new path with lower costs (Figure 8(f)).

V. EXPERIMENTS

To validate the proposed method, we run experiments assuming both known and unknown scenarios. The formers (Section V-A) aim to to show how, in contrast to standard
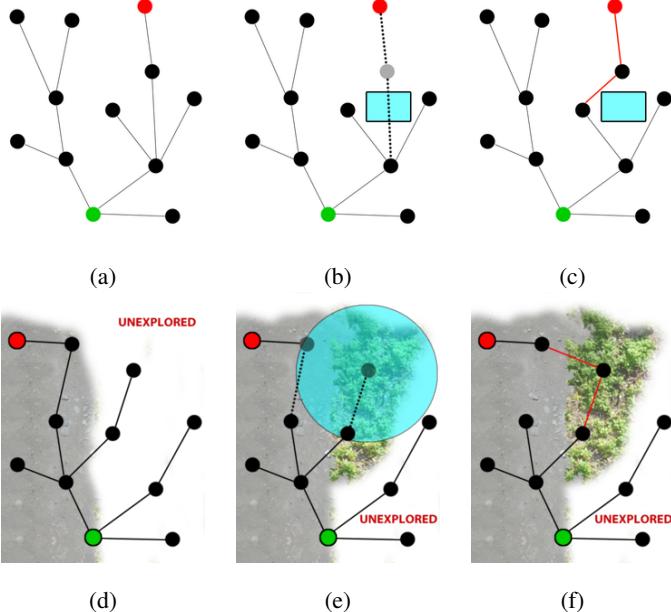
(a)   (b)   (c)

(d)   (e)   (f)

Fig. 8: Online update steps during exploration: Figures (a)-(c) depict the subtree invalidation and rewiring update when an obstacle is spotted, while (d)-(f) show how the tree is rewired when new photometric information is available from the scene.

strategies, our perception-aware path planner selects trajectories that favor highly-textured areas. In the latter ones (Section V-B) we demonstrate the capability to adapt the perception-aware plan in an online fashion as new information is available from the environment. Furthermore, we test our approach within a complete visual navigation system that explores, localizes itself and computes trajectory considering the visual information from the scene.

A. Experiments in Known Scenarios

We evaluate the approach in both simulated and real scenes. In the simulated experiments, we used Blender to generate photorealistic, textured scenes and render images with the associated depth maps. We assume a down-looking camera in both simulated and real scenarios. In contrast to the experiments in the following sections, here we assume to have full knowledge about the map and the texture in the scene.

Our framework can handle 6DoF state representations (i.e., $(x, y, z, \rho, \phi, \theta)$). However, since we assume flight in near-hover conditions, without loss of generality, we can omit the roll and pitch angles (i.e., $\rho = 0, \phi = 0$). Furthermore, since the orientation angle θ does not affect the information-gain computation with down-looking camera, we can also omit θ (in the experiments in unknown scenarios, described in Section V-B, we consider also the front-looking configuration, i.e. we plan including the yaw angle).

1) Simulation Results: We set up two different simulation scenarios to prove that our approach can effectively compute the optimal trajectory with respect to the uncertainty reduction. In particular, we discuss the effect of the trade-off factor α (22) on the computed path. In the first experiment (Figure 9),

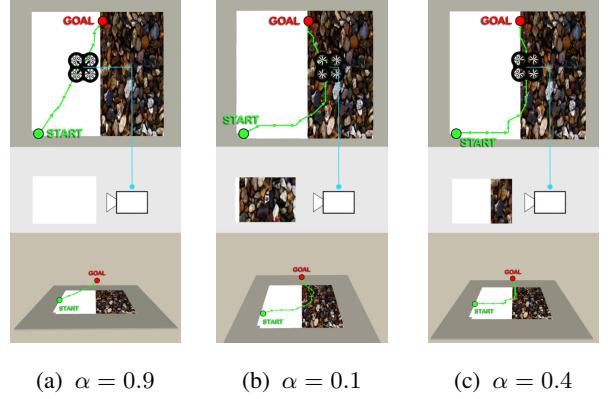(a) $\alpha = 0.9$ (b) $\alpha = 0.1$ (c) $\alpha = 0.4$

Fig. 9: Results of the experiment with two textures. The images are extracted from the graphical interface of the planner where the measures are displayed as a colored point cloud. The green arrows indicate the optimal path. The experiments with $\alpha =$0.9, 0.1 and 0.4 are shown from left to right. The first row shows the top view above the scene and in the second one we depict the image from the down-looking camera acquired at an intermediate pose along the trajectory. In the third row a 3D perspective view is depicted.

the scene is divided into two areas: the first one textureless and the second one with texture. The second scenario (Figure 10) contains texture that only reduces the uncertainty along one dimension, e.g., with zero intensity gradient along specific directions. In particular, the scene is characterized by black and white stripes along the x and y directions. This test is designed to demonstrate how our approach predicts the pose uncertainty specifically for each state dimensions and plans accordingly.

For each simulated scenario, we render images at different locations. This way, we can compute the photometric information gain with different camera viewpoints and update the predicted pose covariance along the trajectory.

In the first test (Figure 9), the space is limited to a 10×10 meter area. The states $(x, y, z) = (0.0, 0.0, 2.0)$ and $(x, y, z) = (2.0, 9.0, 2.0)$ are chosen respectively as the start and the goal state. We split the scenario in two areas: the first is textureless, while the second one is highly-textured. For this experiment we keep the camera height at 2 meters above the ground. As the start and the goal state are both located in the white zone, selecting a straight trajectory that only minimizes the distance leads to a viewpoint sequence without texture.

We run three tests setting the parameter α to 0.9, 0.1 and 0.4, respectively. In the first one, the planner penalizes long paths, while in the second one a higher cost is associated to trajectories with high uncertainty. Finally, in the last one, the computed trajectory is a trade-off between localization accuracy and trajectory length.

Figure 9 shows that in the case $\alpha = 0.9$, the planner correctly selects the trajectory close to the shortest one (i.e., a line). In the second case $\alpha = 0.1$, the optimal viewpoint sequence includes the textured area, to keep the uncertainty small as long as possible along the path. Finally, in the case $\alpha = 0.4$ the computed path keeps the pose covariances small, but, since more weight is given to the distance term in the cost function, the planner reduces the trajectory length as much as possible.
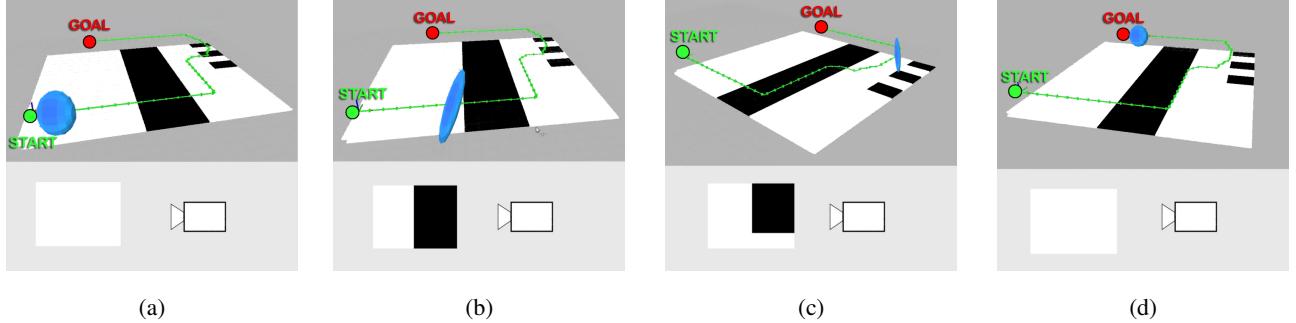
Fig. 10: Uncertainty propagation samples from the computed optimal policy. In this test α is set to 0.1 In the first row the red cloud indicates the covariance at the given position, while in the second row the camera image rendered in that position is displayed. In the second and in the third column is it possible to see how the uncertainty is reduced first along the x axis and then along the y and the z axes.

Within the second simulation (see Figure 10), we demonstrate how the proposed approach seeks to maximize the information gain along all dimensions of the space domain. As explained in section IV-A, we can achieve this behavior through the proposed cost function, which tries to minimize the sum of the traces of the pose covariances.

In this case the space is constrained to a 20×20 meter area. The start state is still set at $(x, y, z) = (0.0, 0.0, 2.0)$ and the final state is $(x, y, z) = (5.0, 19.0, 2.0)$. The simulated scenario is composed of three types of texture: one completely white and the remaining two with black and white stripes along the y and the x axes respectively. Furthermore, we set $\alpha = 0.1$, *i.e.*, we look for the path that minimizes the uncertainty. Figure 10 shows the resulting optimal path. The planner suggests first reaching the area with the stripe along the y axis, then navigating to the stripe along the x axis to minimize the uncertainty along both directions. Furthermore, as shown in Figure 10, we can also reduce the uncertainty of the z dimension. When two gradients with known relative position are available we can gain information about the depth.

In Table I we also report the comparison between the trajectory length and the trace of the pose covariance matrix for different values of α. In particular, we run tests in the two scenarios varying α from 0.05 to 0.95 with a 0.1 step. We perform 10 runs per tests averaging the trajectory length, the mean trace along the path, and the trace at the goal location. In the first test, as we give more importance to the pose uncertainty minimization, the length of the trajectories varies between 9.21 m and 12.91 m, while the mean and the goal state traces are reduced. The results vary almost linearly because the optimal trajectory changes smoothly between a straight towards the goal (shorter paths) and two almost orthogonal segments (safer paths), as we change the value of α. Conversely, in the second experiment we can observe three different behaviors. When more importance is given to the trajectory length, the planner selects a straight path towards the goal (over the area with no texture), thus, the uncertainty is very high. On the other hand, with small α values, the trajectories selected are similar to the one depicted in Figure 10. However, when α is between 0.35 and 0.65, the planner computes paths that reach only the first black stripe, without going over the black squares on the other side of the space. In this way, it is not possible to reduce the uncertainty with respect to the y axis, but the trajectory is shorter.

2) Real Experiments: While simulated scenarios are well-suited to demonstrate the capabilities of the proposed framework, a real-world experimental setup is important to prove the effectiveness of the approach in actual environments. The 3D surface model of the scene was computed using a Faro 3D laser scanner[2] to gather a fine-grained point cloud representation of the scenario. After recording the scans, we generated the state space and computed trajectories given different start and goal points. In addition, we used a quadrotor with a down-looking camera to perform the computed trajectories.

We set up two scenarios to test our approach with different texture and object arrangements. For each scenario, a full 3D scan of the room was acquired. Figure 11 shows different scenario setups during scan acquisitions.

In the first configuration, shown in Figure 11(a), the scene is left without texture on the floor, apart from the area near the walls, where we added highly-textured boxes and carpets (Figure 5(b) shows the photometric information gain at different locations in the scene). The start position is set close to the room door, while the goal position is located in the opposite corner of the room. We compare the standard RRT* planner ($\alpha = 1.0$) with our Perception-aware RRT* using two different configurations: in the first one, more importance is given to pose-uncertainty minimization ($\alpha = 0.1$); in the second one, the planner is asked to select the trajectory that also favors short path lengths ($\alpha = 0.3$). As shown in Figure 12, while the obvious solution for the standard RRT* planner is to go straight to the goal along the diagonal of the room (see Figure 12(c)), our framework understands that it is not the best path with respect to visual localization, as no texture
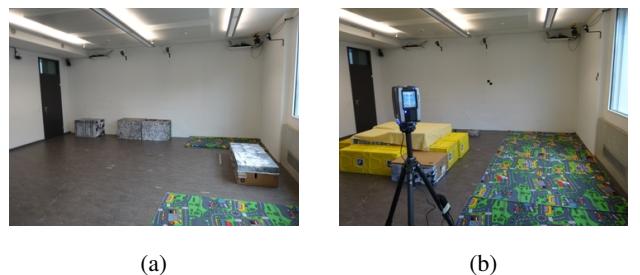
Fig. 11: Two different scenario setups in our laboratory.

α	First Scenario (Figure 9)			Second Scenario (Figure 10)		
	Avg. Length [m]	Avg. Mean Trace	Avg. Goal Trace	Avg. Length [m]	Avg. Mean Trace	Avg. Goal Trace
0.05	12.91	2.1	1.0	40.12	7.60	9.05
0.15	12.91	3.56	1.5	40.45	7.75	9.25
0.25	11.50	6.4	1.8	40.23	7.23	9.28
0.35	11.05	9.05	4.0	23.36	35.23	18.35
0.45	11.05	9.12	3.87	23.33	37.11	17.98
0.55	11.04	10.36	4.23	23.45	36.59	19.12
0.65	10.5	25.4	8.34	23.5	38.67	18.81
0.75	9.89	30.22	18.45	19.67	65.61	76.34
0.85	9.67	30.67	18.12	19.63	67.04	78.24
0.95	9.21	30.5	19.09	19.64	69.12	79.67

TABLE I: Comparison between the trajectory length and the pose uncertainty for different α values.

(a) $\alpha = 0.1$

(b) $\alpha = 0.3$

(c) $\alpha = 1.0$

Fig. 12: Results of the first scenario tests in a real environment. Each row shows the computed optimal path for each planner parametrization. In particular 12(a) and 12(b) depict the computed trajectories with $\alpha = 0.1$ and $\alpha = 0.3$, while 12(c) displays the standard RRT* output ($\alpha = 1.0$). The first two columns from the left show two different perspective views of each trajectory in the scenario, while in the rightmost column depicts the interpolated trajectory in red.

for sufficient pose estimation is available. In particular, when $\alpha = 0.1$, it selects the trajectory along the walls, retrieving photometric information from the scenario. It should also be noticed that setting $\alpha = 0.3$ results in a path that keeps the robot close to the walls to minimize the pose covariance while at the same time reducing the path length as much as possible.

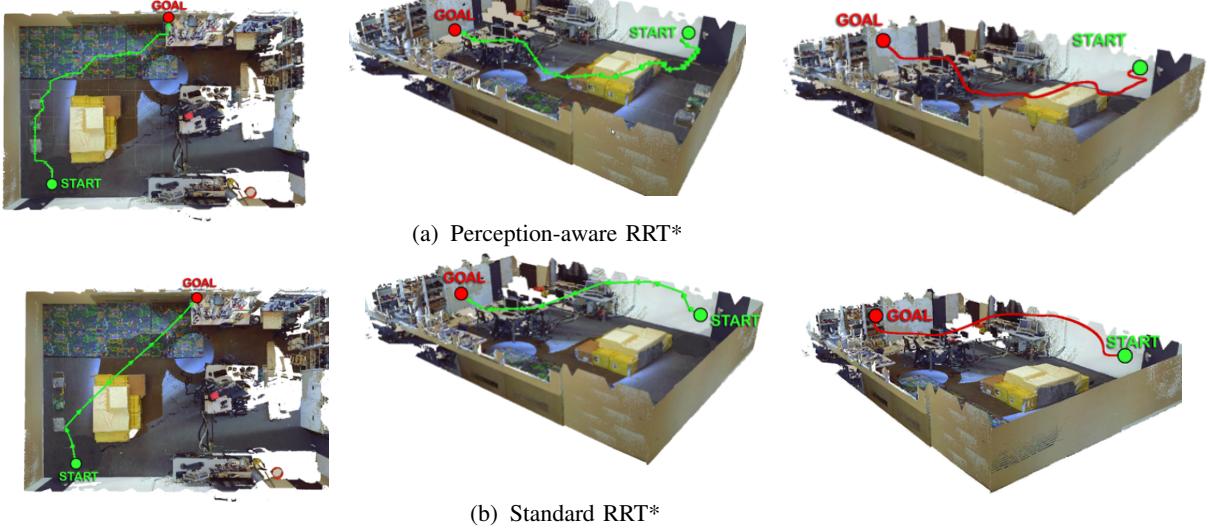(a) Perception-aware RRT*

(b) Standard RRT*

Fig. 13: The trajectories computed in the second scenario experiment. Our Perception-aware RRT* with $\alpha = 0.1$ 13(a) is compared with the standard RRT* 13(b).

Thus, compared to the $\alpha = 0.1$ experiment, the trajectory is shorter but with a less accurate pose estimation.

The last scenario, shown in Figure 11(b), was set with some boxes with uniform color in the center of the room and with texture-rich carpets and boxes along the walls. The start and the goal states are the same as in previous scenarios. Our Perception-aware planner with $\alpha = 0.1$ is again compared with the standard RRT* approach. The minimum-length trajectory (*i.e.*, the output of the RRT* planner) is obtained selecting viewpoints over the boxes in the middle of the room (see Figure 13(b)). However, this path is poor in photometric information, thus, the strategy implemented by our approach chooses a trajectory along the walls, circumventing the central boxes and keeping the quadrotor height low to maximize the uncertainty reduction (cf. Figure 13(a)). Inspecting the results in Figure 13, it can be seen that our planner increases the height in some parts of the trajectory. Although, in general, higher depth values reduce the photometric information for the translational components, higher waypoints have larger scene coverage. In this scenario, as some parts of the scene are poor in texture, photometric information increases with height because this boosts the possibility of acquiring richer texture from other areas.

Finally, in Figures 14 and 15 we compare the pose covariance estimates relative to the trajectories computed with the standard RRT* and the proposed Perception–aware RRT*. The plots clearly show that we can effectively reduce the pose uncertainty by selecting paths over highly-textured areas. Conversely, since the standard RRT* planner does not take into account any photometric information, the resulting trajectories provide a small amount of texture to the visual localization system and, thus, they are characterized by larger covariance values.

B. Experiments in Unknown Scenarios

In this section we discuss the experiments in unknown scenarios. We first describe the architecture of the visual navigation system that performs online localization, map reconstruction and planning. Afterwards, we present the results achieved in both simulated and real environments.

1) System Overview: We consider an MAV that explores an unknown environment by relying only on its camera to perform localization, dense scene reconstruction and optimal trajectory planning. We have integrated the online perception-aware planner with two different mapping systems (see Figure 16): a monocular dense reconstruction system that generates a point cloud map, and a volumetric system that uses stereo camera input.

In the monocular system, the localization of the quadrotor runs onboard, providing the egomotion estimation to perform navigation and stabilization. To achieve real-time performance, the dense map reconstruction and the online perception-aware path planning runs off-board on an Intel i7 laptop with a GPU, in real-time.

At each time step k, the quadrotor receives a new image to perform egomotion estimation. We use the Semi-direct monocular Visual Odometry (SVO) proposed in [4], which allows us to estimate the quadrotor motion in real-time. The computed pose $\mathbf{T}_{k,w}$ and the relative image are then fed into the dense map reconstruction module (REMODE [36], a probabilistic, pixelwise depth estimator to compute dense depthmaps). Afterwards, the dense map provided by the reconstruction module is sent to the path planning module and is used to update both the collision map (using Octomap [37]) and the photometric information map. The last one is then used to update $\mathbf{\Lambda}_v$ for each vertex affected by the map update. Finally, we update the optimal trajectory following the procedure described in Algorithm 2.

For the textured volumetric map system, we take input from

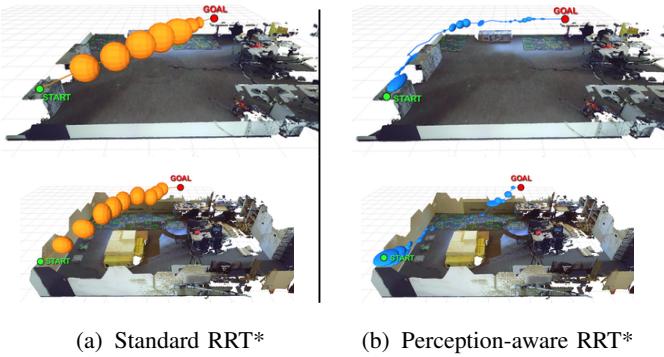(a) Standard RRT* (b) Perception-aware RRT*

Fig. 14: Pose estimation uncertainties plotted for each experiment. The figure compares the standard planner output 14(a) with the proposed perception-aware results 14(b). The covariances and the sequences of viewpoints computed with the standard RRT* are pictured in orange, while the perception-aware RRT* is depicted in blue. From top to bottom, the figure shows the computed trajectories in the two scenarios.
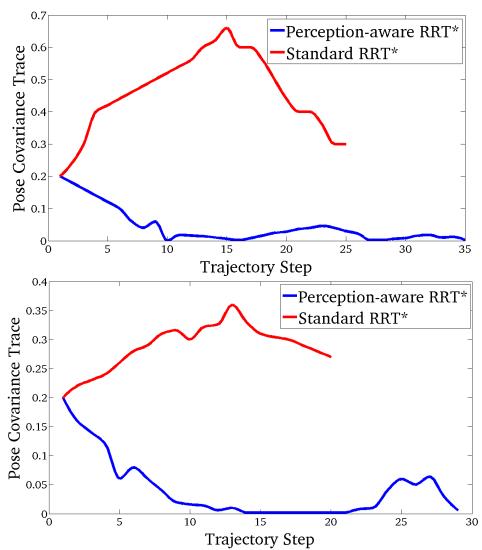

Fig. 15: Comparison of the pose covariance estimates along the trajectories computed with the standard RRT* and our perception-aware RRT*. The plot at the top depicts the comparison of the pose covariance trace for the first scenario (see Figure 11(a)), while the bottom one shows the results of the experiments on the second scenario (see Figure 11(b)). Despite the Standard RRT* trajectories are shorter, the pose covariance uncertainty along the paths is significantly higher than our perception-aware RRT*.

a stereo camera, perform egomotion estimation with SVO as above, and compute a dense depth map with OpenCV's Block Matcher. The estimated camera pose from SVO and the point cloud produced from the depth map are used to update a textured OctoMap. This volumetric map serves as a collision map, when it is queried for occupancy, and is used to synthesize views and compute photometric information gain during planning, when it is queried for texture. This pipeline runs in real time onboard an MAV's embedded single board computer (an Odroid XU3 Lite) using a map with $5cm$ resolution, and with the input images downsampled by a factor of 4 to 188×120, and throttled down to $1Hz$. However,
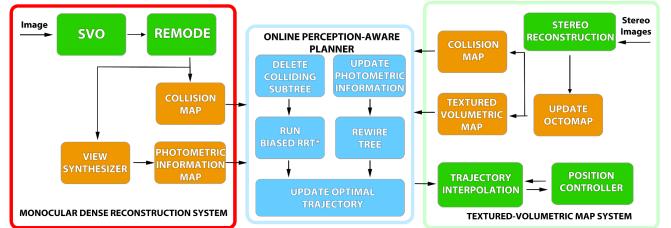

Fig. 16: Block diagram of the online perception-aware planning system.

we evaluate this system in simulation, and for the experiments in Sec. V-B3, we run the simulation, visual pipeline, planner, and control software all on a laptop with an Intel i7 processor.
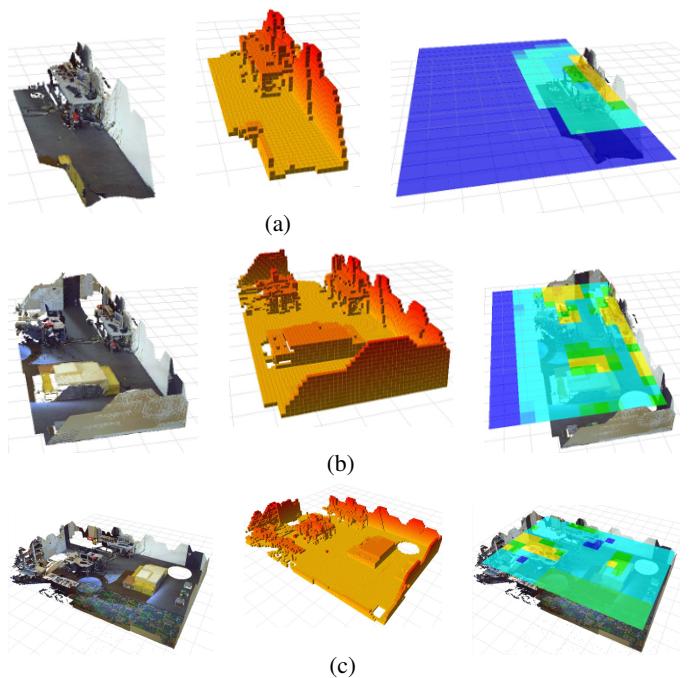

(a)

(b)

(c)

Fig. 17: Three different exploration stages of a scene (rows). The first column shows the scene layout, the second column the collision map and the third the computed photometric information gain.

2) Real Experiments: Before presenting the experimental results, we motivate our approach by discussing how the photometric information distribution changes over time when exploring an unknown environment. Figure 17 shows the map for collision avoidance and the photometric information gain at different exploration stages. In the photometric information map, warm (yellowish) colors refer to camera viewpoints exhibiting a higher amount of texture, while the cool (bluish) ones indicate less informative areas. In Figure 17(a) the almost unexplored scene has very little valuable information to compute a reliable trajectory. Hence, standard planners, that calculate trajectories only once (without performing online updates), compute sub-optimal trajectories or even collide with undiscovered objects. Therefore, an online approach is needed to integrate the information from newly unexplored areas and re-plan accordingly. While exploring, the collision map and photometric information get updated (see Figures 17(b)

and 17(c)) and become useful to update the optimal trajectory.

For the experiment in unknown real scenarios, we set up three scenarios with different object and carpet arrangements to vary the texture and the 3D structure of the scene. In the first scenario, the camera on the MAV is downward-looking, while in the last one choose a front-looking configuration with an angle of 45 degrees with respect to the ground plane. We made experiments with two different camera setups to investigate the influence of the camera viewpoint on the optimal trajectory computation. Intuitively, the front-looking configuration provides more information since also areas far from the quadrotor are observed. Conversely, with the downward-looking configuration, the pose estimation algorithm is more reliable, but less information is captured from the scene. Finally, in all the experiments we set $\alpha = 0.1$ to increase the importance of the pose uncertainty minimization.

In all the scenarios, we put highly-textured carpets along the walls, while the floor in the center of the room is left without texture (*i.e.*, with a uniform color). We also place some boxes on the carpets and near the walls. In the first scenario, we also put an obstacle in the center of the room. At the beginning of the exploration, the planner shows a behavior similar in all the experiments (see Figures 18(a), 18(d) and 18(g)). The information about the scene is very low, thus, our approach computes a simple straight trajectory to the goal. As the robot explores the environment, the trajectory is updated by preferring areas with high photometric information. In the first scenario, we can observe that, since a new obstacle (a box near the center of the room) is spotted at the end of the exploration, the previous trajectory (cf. Figure 18(b)) is invalid and a new collision-free one is computed (see Figure 18(c)). However, to guarantee the availability of photometric information, our approach correctly suggests to fly over the textured boxes and not toward the center of the room.

A front-looking camera configuration (second and third scenario) provides photometric information about areas distant from the current MAV pose. As a consequence, we can obtain an optimal trajectory, with respect to pose uncertainty minimization, earlier with respect to the previous experiment (see Figures 18(e), 18(f), 18(h) and 18(i)). In the final stage of the exploration of the third scenario, the obstacles near the goal are spotted (see Figure 18(i)). As a consequence, the trajectory in Figure 18(h) is invalidated. Despite more texture are available, flying over the top left corner of the room is not anymore convenient due to the presence of the boxes near the goal position. Therefore, our approach correctly updates the trajectory. In this last experiment, we can also observe that, even if the reconstructed map is noisy, our approach correctly computes the best trajectory with respect to the pose uncertainty minimization. Sparse methods would be more affected by the reconstruction error compared to the used dense image-to-model alignment strategy which can effectively capture the photometric information.

3) Simulated Experiments: To further evaluate the performance of our system in wider and more complex scenarios, we also run tests in a simulated environment, using the components described in Sec. V-B1. Two trials were performed in environments simulated with *Gazebo*, one designed to explicitly test perception (*labyrinth*) and one designed to simulate a real world environment (*kitchen*). The labyrinth scenario is designed with flat and highly-textured walls to test the capability of our perception-aware planner to choose the MAV orientations that maximize the amount of photometric information. The quadrotor starts in one of the two long corridors in the scene (see 19(a)) and is asked to reach the goal location that is located at $25m$ from the start location. In the kitchen world (see 19(d)), the MAV begins at a position that is separated by two walls from the goal location, which is $12.5m$ away. We compare the performance of the standard RRT* planner and our perception-aware planner in Figs. 19 and 20.

4) Discussion: The qualitative results shown for the real world (Fig. 18) and simulated (Fig. 19) experiments show that the perception-aware planner does indeed choose trajectories that allow the MAV to observe more photometric information. Quantitatively, this results in a dramatic improvement in the uncertainty of the vehicle's pose estimate. The results in Fig. 20 show that the pose uncertainty, measured as the trace of the covariance matrix and visualized as ellipses in Fig. 19, is up to an order of magnitude smaller when the planner considers the texture of the environment.

In both of the simulated experiments, the RRT* and perception aware planners both reached the goal location in all trials. On average, for the *labyrinth* it took $718s$ and $715s$, respectively, and for *kitchen* it took $578s$ and $580s$, respectively. The results are shown in Figs. (19(b)) and 19(c) for the labyrinth tests and in Figs. 19(e) and 19(f) for the kitchen ones. The most important distinction in this performance comparison is the pose uncertainty across the trajectory. The two planners produce similar trajectories in terms of waypoint positions, but the covariances for the RRT* trajectory are much larger due to the desired yaw angles that are chosen for the waypoints. The proposed perception aware planner specifically optimizes the waypoint position and yaw angle (i.e. *where to look*) in order to minimize this pose uncertainty. As a consequence, the trajectory computed with our strategy has low pose uncertainty values, while the RRT* trajectory, which does not consider the visual information, leads to very low localization accuracy, which can make the navigation infeasible due to the high risk of collisions.

VI. CONCLUSION AND FUTURE WORK

In this paper, we gave a new double twist to the problem of planning under uncertainty by proposing a framework (called Perception-aware Path Planning) to incorporate the photometric information of a scene, in addition to geometric one, to compute trajectories with minimum localization uncertainty of vision-control robots in goal-reaching tasks.

To avoid the caveats of feature-based localization systems (i.e., dependence of feature type and use-defined thresholds), we proposed to use *dense, direct methods* to compute the Fisher information matrix directly from the intensity values of every pixel in the image. We used Lie-Group-based propagation to approximate the localization uncertainty up to the
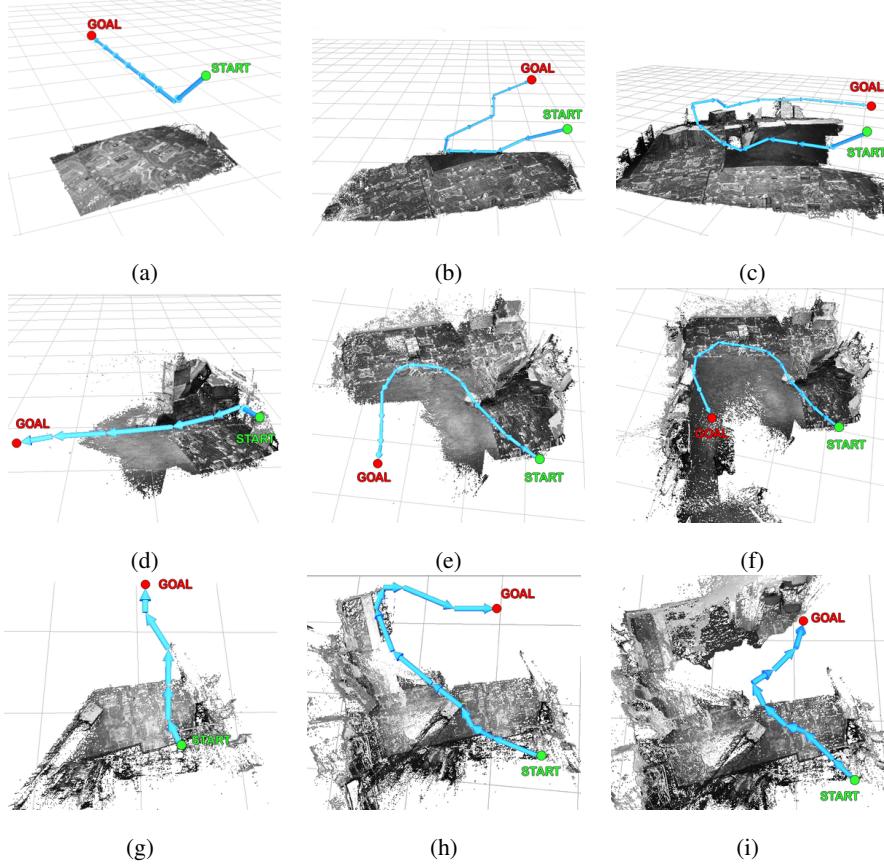
Fig. 18: Experimental results in three real scenarios (rows). The first column shows the initially computed trajectories, only having little information of the environment available. The second and third column demonstrate the update of the trajectory as new information is gathered by updating the scene.

fourth order. Finally, we proposed to adapt trajectories in an online fashion, considering also scenarios with no prior knowledge about the map.

The proposed framework is general and can easily be adapted to different robotic platforms and scenarios. As an application, we showed how the proposed framework can be adapted to the well known RRT* planner.

The proposed framework was validated in both real and simulated environments. Finally, we presented the integration and demonstration of the overall system into a real quadrocopter performing vision-based localization, dense map reconstruction, and online perception-aware planning. The results clearly show that our framework can generate trajectories that outperforms standard path-planning approaches in terms of vision-based localization accuracy.

We believe that this will translate into safer trajectories for vision-controlled robots. Future work will investigate solutions to predict the photometric information gain in unexplored areas using past knowledge. This way, we will be able to reach better estimates of the optimal trajectory even before discovering all the scene elements. Finally, we plan to include dynamic constraints and control effort in the optimization process to generate smoother trajectories.

REFERENCES

[1] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]. Part I: The first 30 years and fundamentals," vol. 18, no. 4, pp. 80 –92, Dec. 2011.

[2] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007, pp. 225–234.

[3] H. Strasdat, J. Montiel, and A. J. Davison, "Scale Drift-Aware Large Scale Monocular SLAM," in *Robotics: Science and Systems (RSS)*, vol. 2, no. 3, 2010, p. 5.

[4] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast Semi-Direct Monocular Visual Odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[5] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *European Conference on Computer Vision (ECCV)*, 2014, pp. 834–849.

[6] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 966–1005, 1988.

[7] S. Soatto, "Steps Towards a Theory of Visual Information: Active Perception, Signal-to-Symbol Conversion and the Interplay Between Sensing and Control," *ArXiv e-prints*, 2011.

[8] B. Bonet and H. Geffner, "Planning with Incomplete Information as Heuristic Search in Belief Space," in *Conference on Artificial Intelligence (AAAI)*, 2000, pp. 52–61.

[9] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[10] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 723–730.

[11] A. Blake and A. Yuille, *Active vision*. MIT press, 1993.

[12] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 333–356, 1988.

(a)

(b) RRT*

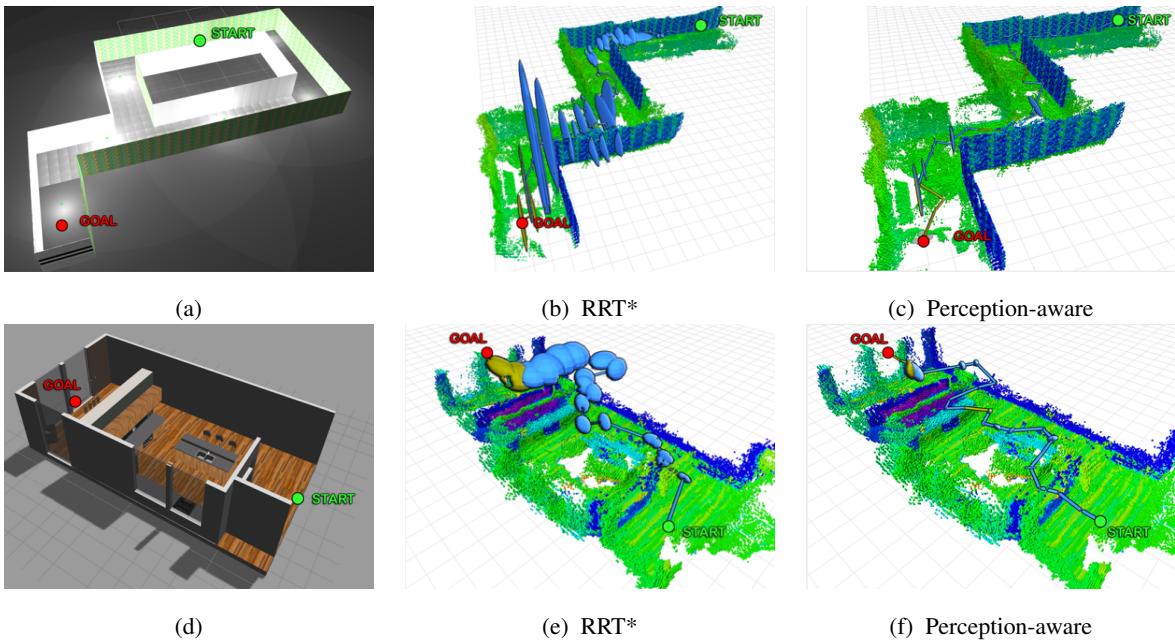(c) Perception-aware

(d)

(e) RRT*

(f) Perception-aware

Fig. 19: Exploration trial in the *labyrinth* (a) and in the *kitchen* (d) simulated environments. The trajectories computed by the RRT* planner are shown in Fig. (b) for the labyrinth scenario and in Fig. (e) for the kitchen, while the ones computed with the perception aware planner are shown in (c) and in (f), respectively. The Textured OctoMaps are visualized with a color corresponding to the mean intensity over all of the observed faces, with red representing high intensity, and purple representing low intensity. The pose covariance at each waypoint is shown as an ellipse, with the most recent update in orange, and the rest of the trajectory in blue.
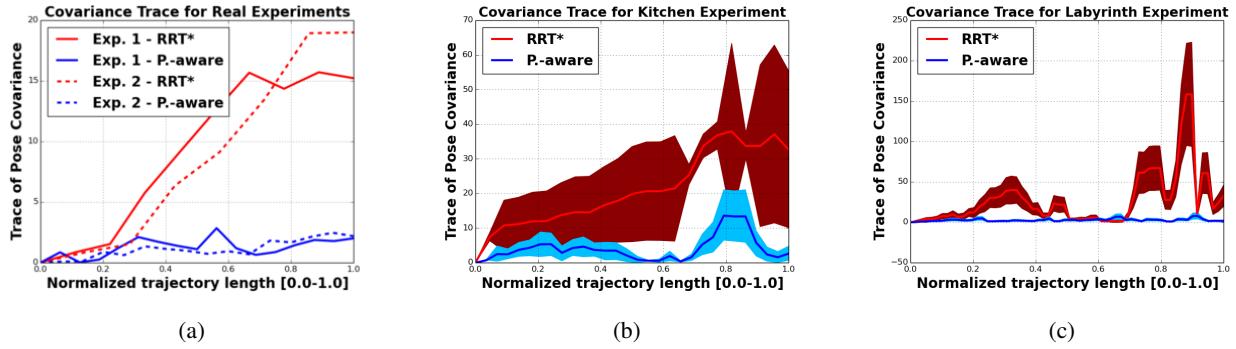

(a)

(b)

(c)

Fig. 20: Quantitative results for our experiments showing the evolution of the MAV's pose covariance during the planned trajectory. Fig. (a) shows results of the real world experiments. Figs. (b) and (c) show the simulated *kitchen* and *labyrinth* trials, respectively. The sequence of viewpoints for each trial result in different length trajectories, so the length of each one is normalized to one. For each simulated experiment, we conducted 15 trials, normalized the trajectories, and inferred Gaussian distributions at each point in a set of equally-spaced samples along a normalized trajectory. In (b) and (c), each solid line represents the mean over all of the trials, and the colored band is the 95% confidence interval.

[13] S. Chen, Y. Li, and N. M. Kwok, "Active vision in robotic systems: A survey of recent developments," *International Journal of Robotics Research*, vol. 30, no. 11, pp. 1343–1377, 2011.
[14] S. Soatto, "Actionable information in vision," in *Machine learning for computer vision*. Springer, 2013, pp. 17–48.
[15] H. J. S. Feder, J. J. Leonard, and C. M. Smith, "Adaptive mobile robot navigation and mapping," *The International Journal of Robotics Research*, vol. 18, no. 7, pp. 650–668, 1999.
[16] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte, "Information based adaptive robotic exploration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002, pp. 540–545.
[17] A. Bachrach, S. Prentice, R. He, P. Henry, A. S. Huang, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1320–1343, 2012.
[18] A. J. Davison and D. W. Murray, "Simultaneous localization and map-building using active vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 865–880, 2002.
[19] T. A. Vidal-Calleja, A. Sanfeliu, and J. Andrade-Cetto, "Action Selection for Single-Camera SLAM," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 40, no. 6, 2010.
[20] C. Mostegel, A. Wendel, and H. Bischof, "Active Monocular Localization: Towards Autonomous Monocular Exploration for Multirotor MAVs," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.
[21] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.
[22] M. W. Achtelik, S. Lynen, S. Weiss, M. Chli, and R. Siegwart, "Motion-and Uncertainty-aware Path Planning for Micro Aerial

Vehicles," *Journal of Field Robotics*, vol. 31, no. 4, pp. 676–698, 2014.

[23] M. Bryson and S. Sukkarieh, "Observability analysis and active control for airborne SLAM," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 44, no. 1, 2008.

[24] M. Irani and P. Anandan, "All About Direct Methods," in *Vision Algorithms: Theory and Practice*. Springer, 2000, pp. 267–277.

[25] S. Lovegrove, A. Davison, and J. Ibanez-Guzman, "Accurate visual odometry from a rear parking camera," *IEEE Intelligent Vehicle Symposium*, 2011.

[26] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense Tracking and Mapping in Real-Time," *IEEE International Conference on Computer Vision (ICCV)*, pp. 2320–2327, 2011.

[27] M. Meilland, T. Drummond., and A. Comport, "A unified rolling shutter and motion blur model for dense 3D visual tracking," December 2013.

[28] C. Forster, M. Pizzoli, and D. Scaramuzza, "Appearance-based Active, Monocular, Dense Depth Estimation for Micro Aerial Vehicles," *Robotics: Science and Systems (RSS)*, 2014.

[29] T. D. Barfoot and P. T. Furgale, "Associating Uncertainty with Three-Dimensional Poses for use in Estimation Problems," *IEEE Transactions on Robotics*, 2014.

[30] G. Costante, W. M. Delmerico, Jeffrey, and D. Paolo Valigi, Scaramuzza, "Exploiting Photometric Information for Planning Under Uncertainty," in *2015 International Symposium on Robotic Research (ISRR)*, 2015.

[31] G. S. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*. Springer, 2011, vol. 2.

[32] M. Meilland and A. I. Comport, "On unifying key-frame and voxel-based dense visual SLAM at large scales," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[33] H. Strasdat, "Local Accuracy and Global Consistency for Efficient Visual SLAM," *PhD Thesis, Imperial College London*, 2012.

[34] R. A. Fisher, "On the mathematical foundations of theoretical statistics," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 222, pp. 309–368, 1922.

[35] S. Haner and A. Heyden, "Optimal view path planning for visual slam," in *Image Analysis*. Springer, 2011, pp. 370–380.

[36] M. Pizzoli, C. Forster, and D. Scaramuzza, "REMODE: Probabilistic, Monocular Dense Reconstruction in Real Time," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2014, pp. 2609–2616.

[37] 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, 2013.