

# Exploiting Photometric Information for Planning Under Uncertainty

Gabriele Costante, Jeffrey Delmerico, Manuel Werlberger,  
Paolo Valigi and Davide Scaramuzza

## 1 Introduction

We consider the problem of planning an optimal trajectory between two spatial locations in an initially unknown environment with an autonomous, vision-controlled, micro aerial vehicle (MAV). In many previous works, optimal trajectories are those with the shortest or lowest effort path to the goal position. To improve the performance of vision-based control, and consequently all of the other perception functions that rely on the robot's pose estimate, we instead consider optimal trajectories to be those that minimize the uncertainty in this pose estimate. Because we compute the robot pose uncertainty as a function of the photometric information of the scene, we call this approach *Perception-aware Path Planning*.

Despite the impressive results achieved in visual SLAM applications [1, 2], most of vision-controlled MAVs navigate towards a goal location using a predefined set of viewpoints or by remote control, without responding to environmental conditions [3, 4]. Recently, several works have tackled the problem of autonomously planning an optimal trajectory towards a goal location [5, 6], and others have extended this to uncertainty-aware planning that tries to provide high localization accuracy [7, 8].

---

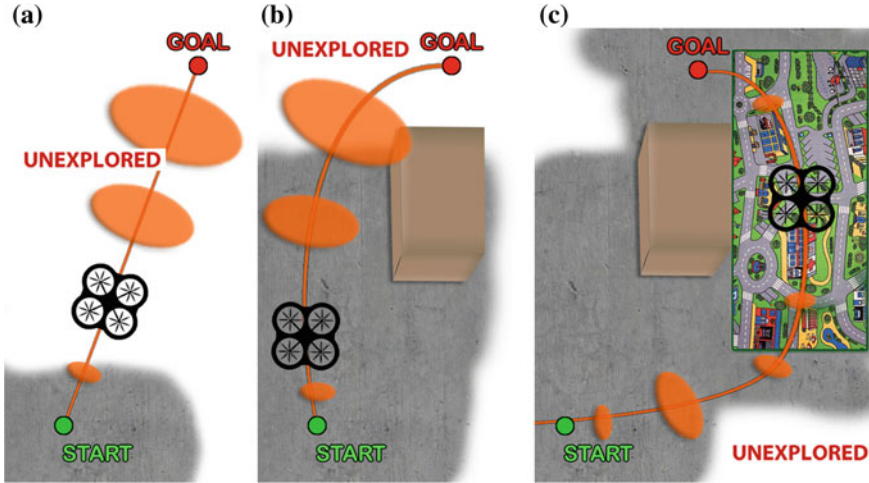
G. Costante (✉) · P. Valigi  
University of Perugia, Perugia, Italy  
e-mail: gabriele.costante@unipg.it

P. Valigi  
e-mail: paolo.valigi@unipg.it

J. Delmerico · M. Werlberger · D. Scaramuzza  
University of Zürich, Zürich, Switzerland  
e-mail: jeffdelmerico@ifi.uzh.ch

M. Werlberger  
e-mail: werlberger@ifi.uzh.ch

D. Scaramuzza  
e-mail: sdavide@ifi.uzh.ch



**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 is longer than the one in b, it contains more photometric information and, thus, is optimal with respect to the visual localization uncertainty

However, these approaches discard the photometric information (i.e. texture) of the scene and plan the trajectory in advance, which requires prior knowledge of the full 3D map of the environment. We propose a system that instead selects *where to look*, in order to capture the maximum visual formation from the scene to ensure pose estimates with low uncertainty.

Additionally, we consider the scenario where the robot has no prior knowledge of the environment, and it explores to generate a map and navigate to a goal. Without an a priori map, we update the planned path as new images are collected by the camera while the robot explores the surroundings (see Fig. 1). In particular, we utilize the photometric information in the newly observed regions of the environment to determine the optimal path with respect to pose uncertainty. To the best of our knowledge, this is among the first works that propose to plan a *perception-aware* trajectory *on-the-fly*, while perceiving the environment with only a camera sensor.

We evaluate the proposed methods with several different experiments designed to illustrate the feasibility of our approach for an autonomous MAV, and to demonstrate the improvement in pose uncertainty when planning with perception awareness.

## 1.1 Related Work

When the minimization of the localization uncertainty is considered in the planning process, the problem is often referred to as “Planning under Uncertainty” or “Planning in Information Space”. Probabilistic planning with the inability to directly observe all the state information is often based on Partially Observable Markov Decision

Processes (POMDPs) or solved as a graph-search problem. The major drawback of these approaches is their exponential growth in computational complexity. Sim and Roy [9] selected trajectories that maximize the map reconstruction accuracy in SLAM applications. They proposed to use a breadth-first search over possible robot positions to predict a sequence of EKF estimates and select the one that lead to the maximum information gain. Recently, sampling-based methods have been introduced to plan trajectories in complex configuration spaces. Optimal Rapidly-exploring Random Trees (RRT\*s) [10] have been widely used in path planning problems and their extension to Rapidly-exploring Random Belief Trees (RRBTs) [7] takes pose uncertainty into account and avoids collisions.

Selecting sequences of viewpoints that optimize for a certain task (e.g. pose estimation or map uncertainty minimization) is referred to as *active perception* [11, 12]. While previous papers on active perception relied on using range sensors (e.g. [8]), Davison and Murray [13] were among the first to use vision sensors (a stereo camera setup) to select where the camera *should look* to reduce the pose drift during visual SLAM. More recently, Sadat et al. [14] and Achtelik et al. [15] investigated optimal path planning by leveraging visual cues. The former ensures good localization accuracy by extending RRT\*s to select feature-rich trajectories, while the latter uses RRBT to compute the propagation of the pose uncertainty by minimizing the reprojection error of 3D map points. Kim and Eustice [16] proposed a Perception Driven Navigation (PDN) framework: the robot follows a pre-planned path and computes information gain at the viewpoints along it, but revisits already-explored, highly-salient areas to regain localization accuracy if its pose uncertainty increases.

It should be noted that all approaches mentioned so far [13–16] rely on *sparse 2D features* to compute highly-informative trajectories. By contrast, in this paper we rely on *direct methods* [17]. Contrarily to feature-based approaches—which only use small patches around corners—direct methods use all information in the image, including edges. They have been shown to outperform feature-based methods in terms of robustness and accuracy in sparsely-texture scenes [1, 2, 18].

Several works have addressed the problem of online planning. Efficient replanning was addressed in Ferguson et al. [19] by updating the trajectory whenever a new obstacle is spotted. For RRT\*, Boardman et al. [20] proposed to dynamically update an initial planned trajectory by computing a new RRT\* tree rooted by the robot’s current location and reusing branches from the initially-grown tree. Otte and Frazzoli [21] further address the problem of online planning in dynamic environments by modifying the original search graph whenever changes in the environment are observed. Among MAVs, Grzonka et al. [5] considered a quadrotor equipped with an on-board laser scanner, and scanned the environment, adapting the trajectory as new objects were spotted. Similarly, Nieuwenhuisen et al. [6] also used a 3D laser scanner on an autonomous quadrotor to build and update an obstacle map and replan collision-free trajectories. Similar approaches based on different sensors, such as cameras or depth sensors, were proposed in [22]. However, the previous approaches [5, 6, 22] rely on configurations that include other sensors (e.g. IMU, Laser Scanner) in addition to cameras. Furthermore, planning is performed without considering the visual perception and, in particular, the photometric information.

## 1.2 Contribution

In contrast to the previous works, in this paper we propose a novel method to update the optimal trajectory that leverages the *photometric information* (i.e, texture) and the 3D structure of newly-explored areas on the fly (i.e, *online*), avoiding full replanning. In order to use that information for minimizing pose uncertainty, we perform path planning in four degrees of freedom (*x, y, z, and yaw*). Furthermore we proposed a novel *textured* volumetric map representation that allows us to efficiently synthesize views to compute the photometric information in the scene and plan accordingly. To the best of our knowledge, this is among the first works to propose a fully autonomous robotic system that performs onboard localization and online perception-aware planning. The main contributions of this paper are:

1. We propose to leverage the *photometric* appearance of the scene, in addition to the 3D structure, to select trajectories with minimum pose uncertainty. The photometric information is evaluated by using *direct* methods. As direct methods use all the information in the image, they provide a more robust and effective way to exploit visual information compared to feature-based strategies.
2. Perception-aware planning is performed online as the robot explores the surroundings, without prior knowledge of the full map of the environment.
3. A novel *textured* volumetric map formulation is proposed to efficiently synthesize views for perception-aware planning.
4. We demonstrate the effectiveness of our approach with experiments in both real-world and simulated environment with a MAV only equipped with vision sensors.

## 2 Perception-Aware Pose Uncertainty Propagation

The visual localization system relies on the availability of texture in the scene to reduce the pose estimation uncertainty. As a consequence, selecting the trajectory that is optimal with respect to the localization accuracy requires evaluation of the pose-uncertainty propagation along a candidate path and the uncertainty reduction associated with the photometric information in the scene.

### 2.1 Pose Propagation

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:

$$\text{SE}(3) := \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \mid \mathbf{C} \in \text{SO}(3), \mathbf{r} \in \mathbb{R}^3 \right\}, \quad (1)$$

where  $\text{SO}(3)$  is the special orthogonal group in  $\mathbb{R}^3$  (the set of spatial rotations, i.e,  $\mathbf{C}\mathbf{C}^T = \mathbf{1}$ ,  $\det \mathbf{C} = 1$ ) and  $\mathbf{1}$  is the  $3 \times 3$  identity matrix. The *Lie Algebra* associated

to the SE(3) Lie Group is indicated as  $\mathfrak{se}(3)$ . To represent the uncertainty of the robot pose, we define a random variable for SE(3) members according to:

$$\mathbf{T} := \exp(\xi^\wedge) \bar{\mathbf{T}}, \quad (2)$$

where  $\bar{\mathbf{T}}$  is a noise-free value that represents the pose and  $\xi \in \mathbb{R}^6$  is a small perturbation that we assume to be normally distributed  $\mathcal{N}(\xi|\mathbf{0}, \Sigma)$ . We make use of the  $\wedge$  operator to map  $\xi$  to a member of the Lie algebra  $\mathfrak{se}(3)$  (see [23]).

We refer to  $\mathbf{T}_{k,w}$  as the robot pose at time  $k$  relative to the world frame  $w$  and to  $\mathbf{T}_{k+1,k}$  as the transformation between the pose at time  $k$  and  $k+1$ .

Assuming no correlation between the current pose and the transformation between  $k$  and  $k+1$ , we can represent  $\mathbf{T}_{k,w}$  and  $\mathbf{T}_{k+1,k}$  with their means and covariances  $\{\bar{\mathbf{T}}_{k,w}, \Sigma_{k,w}\}$  and  $\{\bar{\mathbf{T}}_{k+1,k}, \Sigma_{k+1,k}\}$ , respectively. Combining them, we get

$$\mathbf{T}_{k+1,w} = \mathbf{T}_{k,w} \mathbf{T}_{k+1,k}. \quad (3)$$

To compute the mean and the covariance of the compound pose, we use the results from [23]. The mean and the covariance, approximated to fourth order, are:

$$\bar{\mathbf{T}}_{k+1,w} = \bar{\mathbf{T}}_{k,w} \bar{\mathbf{T}}_{k+1,k}, \quad \Sigma_{k+1,w} \simeq \Sigma_{k,w} + \mathcal{T} \Sigma_{k+1,k} \mathcal{T}^\top + \mathcal{F} \quad (4)$$

where  $\mathcal{T}$  is  $Ad(\bar{\mathbf{T}}_{k,w})$ , the adjoint operator for SE(3), and  $\mathcal{F}$  encodes the fourth-order terms. Using Eq. (4), we propagate the uncertainty along a given trajectory.

## 2.2 Measurement Update

In contrast to previously published approaches, which mostly rely on *sparse* image features, we use direct methods, in the form of *dense image-to-model alignment*, for the measurement update. Integrating the intensity and depth of *every* pixel in the image enables us to consider photometric information when planning the trajectory.

### 2.2.1 Preliminary Notation

At each time step of the robot navigation, we can compute a dense surface model  $S \in \mathbb{R}^3 \times \mathbb{R}^+$  (3D position and grayscale intensity) of the explored part of the scene. The rendered synthetic image is denoted with  $\mathbf{I}_s : \Omega_s \subset \mathbb{R}^2 \rightarrow \mathbb{R}^+$ , where  $\Omega_s$  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:  $\mathbf{D}_s : \Omega_s \rightarrow \mathbb{R}^+$ ;  $\mathbf{u} \mapsto d_{\mathbf{u}}$  where  $d_{\mathbf{u}}$  is the depth associated to  $\mathbf{u}$ . 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  $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ ,  $\mathbf{u} = \pi(\mathbf{p})$ . On the other hand, we can recover the 3D point associated to the pixel  $\mathbf{u}$  using the inverse projection function  $\pi^{-1}$  and the depth  $d_{\mathbf{u}}$ :

$$\mathbf{p}_u = \pi^{-1}(\mathbf{u}, d_u). \quad (5)$$

Note that the projection function  $\pi$  is determined by the intrinsic camera parameters that are known from calibration. Finally, a rigid body transformation  $\mathbf{T} \in \text{SE}(3)$  rotates and translates a point  $\mathbf{q}$  to:

$$\mathbf{q}'(\mathbf{T}) := (\mathbf{1} \mid \mathbf{0}) \mathbf{T} (\mathbf{q}^T, 1)^T. \quad (6)$$

### 2.2.2 Dense Image-to-Model Alignment

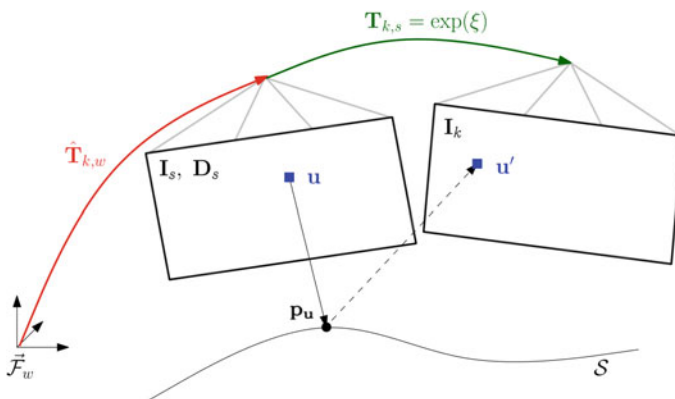
To refine the current pose estimate, we use *dense image-to-model alignment* [18, 24] (see Fig. 2). This approach computes the pose  $\mathbf{T}_{k,w}$  of the synthetic image  $\mathbf{I}_s$  by minimizing the photometric error between the observed image and the synthetic one. Once converged, it also provides the uncertainty of the alignment by evaluating the *Fisher Information Matrix*, which we use to select informative trajectories.

The photometric error  $r_u$  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  $\hat{\mathbf{T}}_{k,w}$ :

$$r_u = \mathbf{I}_k(\mathbf{u}) - \mathbf{I}_s(\pi(\mathbf{p}'_u(\hat{\mathbf{T}}_{k,w}))) \quad (7)$$

The error is assumed to be normally distributed  $r_u \sim \mathcal{N}(0, \sigma_i^2)$ , where  $\sigma_i$  is the standard deviation of the image noise.

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  $\hat{\mathbf{T}}_{k,w} \leftarrow \exp(\hat{\boldsymbol{\xi}}) \hat{\mathbf{T}}_{k,w}$ ,  $\hat{\boldsymbol{\xi}} \in \mathfrak{se}(3)$  that minimize the error. The update step minimizes the least-squares problem:



**Fig. 2** 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  $\{\mathbf{I}_s, \mathbf{D}_s\}$  from the 3D model  $S$

$$\xi = \arg \min_{\xi} \sum_{\mathbf{u} \in \Omega_s} \frac{1}{2\sigma_i^2} \left[ \mathbf{I}_k(\mathbf{u}') - \mathbf{I}_s(\pi(\mathbf{p}'_{\mathbf{u}}(\hat{\mathbf{T}}_{k,w}))) \right]^2, \quad (8)$$

with  $\mathbf{p}_{\mathbf{u}}$  given by (5),  $\mathbf{p}'_{\mathbf{u}}$  as in (6), and  $\mathbf{u}' = \pi(\mathbf{p}'_{\mathbf{u}}(\exp(\xi^\wedge)))$ .

Addressing the least-squares problem (8) we can compute the optimal  $\xi$  using the Gauss-Newton method and solving the normal equations  $\mathbf{J}^T \mathbf{J} \xi = -\mathbf{J}^T \mathbf{r}$ , where  $\mathbf{J}$  and  $\mathbf{r}$  are the stacked Jacobian and image residuals of all pixels  $\mathbf{u} \in \Omega_s$ , respectively.

At the convergence of the optimization, the quantity

$$\mathbf{A}_k = \frac{1}{\sigma_i^2} \mathbf{J}^T \mathbf{J} \quad (9)$$

is the *Fisher Information Matrix* and its inverse is the covariance matrix  $\Sigma_{\mathbf{I}_k}$  of the measurement update. According to [23], we find the covariance matrix after the measurement update at time  $k$  by computing

$$\Sigma_{k,w} \leftarrow \left( \mathbf{A}_k^{-1} + \mathcal{J}^{-T} \Sigma_{k,w} \mathcal{J}^{-1} \right)^{-1}, \quad (10)$$

where the ‘‘left-Jacobian’’  $\mathcal{J}$  is a function of how much the measurement update modified the estimate. Given the information matrix in (9), we define the photometric information gain as  $\text{tr}(\mathbf{A}_k)$ .

### 3 Online Perception-Aware Path Planning

The framework described in Sect. 2 is able to predict the propagation of the pose uncertainty along a given trajectory by integrating the photometric information when available. However, to select the *best* sequence of camera viewpoints we need to evaluate all the possible trajectories. 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 plan that is considered optimal in the beginning, will be adapted as new information is gathered by the robot.

In this section, we describe how we enhance the RRT\* [10] with the perception-aware nature that takes benefit from the photometric information to select the trajectory that is optimal with respect to the localization accuracy.

The RRT\* incrementally grows a tree in the state space by randomly sampling and connecting points through collision-free edges. Optimality is guaranteed through the *rewire* procedure, which checks for better connections when adding a new point to the tree. The tree is composed of a set of vertices  $V$  that represent points in the state space. Each vertex  $v = \{x_v, \Sigma_v, \mathbf{A}_v, c_v, p_v\} \in V$  is described via its state  $x_v = \mathbf{T}_{v,w}$  (i.e, the pose relative to the vertex  $v$  with respect to the reference frame  $w$ ),  $c_v$  being the accumulated cost of the trajectory up to  $v$  and a unique parent vertex  $p_v$ . In addition, we add the pose covariance  $\Sigma_v$  and the photometric information  $\mathbf{A}_v$

(relative to the camera view associated to the pose  $x_v$ ) to the vertex  $v$  in order to update the pose covariance according to the photometric information.

To select the best path among all possible trajectories  $\mathcal{T}_i \in \mathcal{P}$ , we minimize:

$$J(\mathcal{T}_i) = \sum_{j=1}^{N_i} \alpha \text{Dist}(x_{v_j^i}, x_{v_{j-1}^i}) + (1 - \alpha) \text{tr}(x_{v_j^i} \cdot \Sigma), \quad (11)$$

---

### Algorithm 1 Perception-aware RRT\*

---

```

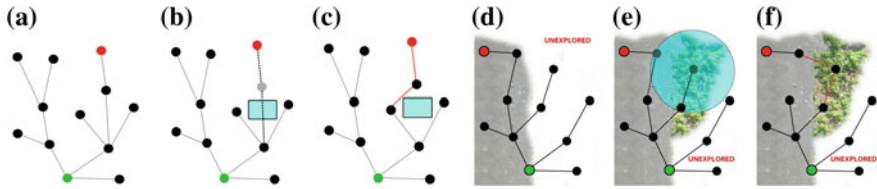
01: Init:  $x_{v_0} = x_{\text{init}}$ ;  $p_{v_0} = \text{root}$ ;  $\Sigma_{v_0} = \Sigma_0$ ;  $c_{v_0}$ ;  $V = \{v_0\}$ ; Number of iterations  $T$ 
02: for  $t = 1, \dots, T$  do
03:    $x_{\text{new}} = \text{SampleUnexplored}()$ 
04:    $v_{\text{nst}} = \text{Nearest}(x_{\text{new}})$ 
05:   if  $\text{ObstacleFree}(v_{\text{new}}, v_{\text{nst}})$ 
06:      $\Sigma_t = \text{PropagateAndUpdate}(x_{v_{\text{nst}}}, \Sigma_{v_{\text{nst}}}, x_{v_{\text{new}}}, \mathbf{A}_{v_{\text{new}}})$ 
07:      $J_{\text{min}} = c_{v_{\text{nst}}} + (1 - \alpha) \text{tr}(\Sigma_t) + \alpha \text{Dist}(x_{v_{\text{nst}}}, x_{v_{\text{new}}})$ 
08:      $v_{\text{min}} = v_{\text{nst}}$ 
09:      $V = V \cup v(x_{\text{new}})$ 
10:      $V_{\text{neighbors}} = \text{Near}(V, v_{\text{new}})$ 
11:     for all  $v_{\text{near}} \in V_{\text{neighbors}}$  do
12:       if  $\text{CollisionFree}(v_{\text{near}}, v_{\text{new}})$ 
13:          $\Sigma_t = \text{PropagateAndUpdate}(x_{v_{\text{near}}}, \Sigma_{v_{\text{near}}}, x_{v_{\text{new}}}, \mathbf{A}_{v_{\text{new}}})$ 
14:         if  $c_{v_{\text{near}}} + (1 - \alpha) \text{tr}(\Sigma_t) + \alpha \text{Dist}(x_{v_{\text{near}}}, x_{v_{\text{new}}}) < J_{\text{min}}$ 
15:            $J_{\text{min}} = c_{v_{\text{near}}} + (1 - \alpha) \text{tr}(\Sigma_t) + \alpha \text{Dist}(x_{v_{\text{near}}}, x_{v_{\text{new}}})$ 
16:            $\Sigma_{v_{\text{new}}} = \Sigma_t$ ,  $c_{v_{\text{new}}} = J_{\text{min}}$ ,  $v_{\text{min}} = v_{\text{near}}$ 
17:         end if
18:       end if
19:      $\text{ConnectVertices}(v_{\text{min}}, v_{\text{new}})$ 
20:   end for
21:    $\text{RewireTree}()$ 
22: end if
23: end for

```

---

where the trajectory is represented by a sequence of  $N_i$  waypoints  $v_j^i$ ,  $\text{Dist}(\cdot, \cdot)$  computes the distance between two vertices and  $\alpha$  defines the trade-off between this distance and the photometric information gain. Note that we jointly optimize for position and yaw orientation w.r.t. information gain, so the optimal poses are not just the RRT\* poses with optimized orientation. We choose the trace to include the visual information into the cost function following the considerations in [25]. In particular, the minimization of the trace of the pose covariance matrix (A-optimality) guarantees that the majority of the state space dimensions are be considered (in contrast to the D-optimality), but does not require us to compute all the eigenvalues (E-optimality). The fundamental steps of the perception-aware RRT\* are summarized in Algorithm 1. At each iteration, it samples a new state from the state space and connects it to the nearest vertex (lines 3–19). Next, the function  $\text{Near}(\cdot)$  checks on the vertices within a ball, centered at the sampled state (see [10]), and propagate the pose covariance from these vertices to the newly sampled one. The one that minimizes the cost function (11) gets selected. Finally, we update the tree connections through the *rewire* procedure. Note that although the optimization is performed on the trace, the full covariance is propagated along each trajectory for evaluation.





**Fig. 3** 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

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 Fig. 3 and its fundamental steps are depicted in Algorithm 2. Consider an initial planning tree as in Fig. 3a, 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 Fig. 3b. Note that the `SampleUnexplored()` function is now 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 Fig. 3d–f demonstrates the case of gaining areas with distinctive photometric information. As newly discovered areas provide photometric information, as shown in Fig. 3e, 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 (Fig. 3f).

---

### Algorithm 2 Online perception-aware RRT\*

---

```

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

```

---

## 4 Textured Volumetric Mapping

We implement an extension to the popular *OctoMap* [26] 3D mapping framework that records texture information within the volumetric map, allowing novel views to be synthesized for perception-aware path planning (see Fig. 4).

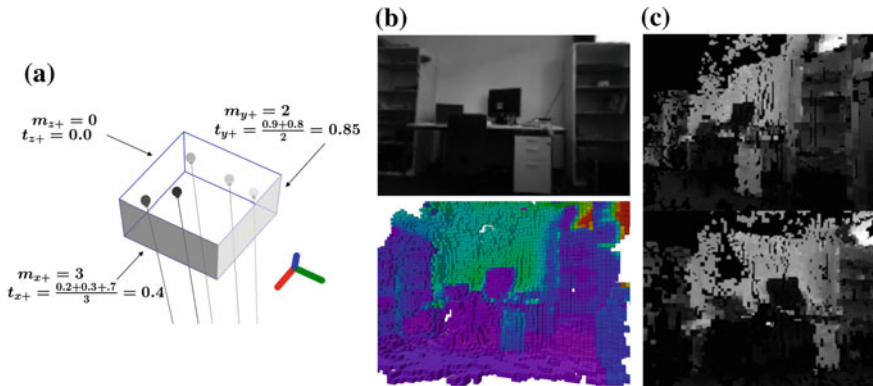
We utilize this stored texture to synthesize views of the known map from hypothetical positions for the camera. For each synthetic view, we synthesize an image of what it would look like to observe the environment from that pose—at least up to the currently observed state of the map—and use these synthetic images in the computation of information gain during planning. As an extension to an *OctoMap* that stores an estimate of occupancy probability for each voxel, we maintain an estimate of the texture for *each face* of each voxel as an intensity value, averaged over all of the observations of that face. We chose this approach because of its compactness—we must only store the current estimate and the number of cumulative observations—and because it is not depth dependent for either updating or querying. It is also directly extensible to a hierarchical representation, such that texture values at higher levels of the octree can be computed from the faces of their child voxels. While our approach to rendering images from a volumetric map is similar to the one in [27], we chose to store texture for the faces, and not just for the volume, because the space represented by a voxel does not necessarily have the same appearance when observed from different sides. Storing more descriptive representations of texture (e.g. Harris corner scores) for the faces would be beneficial, but these metrics are often dependent on the range at which they are observed, presenting a barrier for maintaining a general estimate. The average intensity representation is efficient to update with new observations, efficient to query for the current estimate, and adds only minimal overhead to the computation required for mapping.

Our update method proceeds as follows. Given an input point cloud, occupancy is updated as in [26], where ray casting from the sensor origin is used to update each leaf voxel along the ray, until the observed point, and each leaf voxel is updated at most once for occupancy. To update the texture, for each point  $p_i^k$  in the  $k^{\text{th}}$  point cloud  $C^k$ , we determine the face  $f$  that its ray intersects in the leaf voxel  $n$  containing  $p_i^k$ . At leaf voxel  $n$ , we maintain the current intensity estimate  $t_f$  and number of cumulative observations  $m_f$  for each face  $f \in 1 \dots 6$  of the voxel cube (see Fig. 4a). After the insertion of  $k$  point clouds, these quantities are:

$$m_f^k = \sum_{j=1}^k |p_i^j \in n|, \quad t_f^k = \frac{\sum_{j=1}^k t_{p_i^j} \in f}{m_f} \quad (12)$$

This can be updated efficiently for each new point cloud input:

$$m_f^{k+1} = m_f^k + |p_i^{k+1} \in n|, \quad t_f^{k+1} = \frac{m_f^k t_f^k + t_{p_i^k}}{m_f^{k+1}} \quad (13)$$



**Fig. 4** Textured volumetric mapping: texture information is stored for each face of each voxel in the OctoMap. Each face maintains a mean intensity value for all of the sensor observations that have intersected with it when adding data to the map (a). A visualization of a *Textured Octomap* is shown in b, where an office scene was observed with a handheld stereo camera. In c, we have synthesized some images from the map, at poses that the camera has not yet observed

The inclusion of texture in the OctoMap requires an additional computational overhead of only 15% for both insertion and querying.

Storing texture in a volumetric map allows us to hypothesize about the photometric information that our robot could obtain if it moved to a particular pose. We do this by synthesizing images of the map from a hypothetical pose, casting rays through each pixel in the image into the map (See Fig. 4c). When these rays intersect with the face of an occupied voxel, we record the texture of the face and the depth to that voxel in intensity and depth images. These synthetic images are generated for each sampled pose when the planner generates or rewires the tree.

## 5 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 Fig. 5): 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.

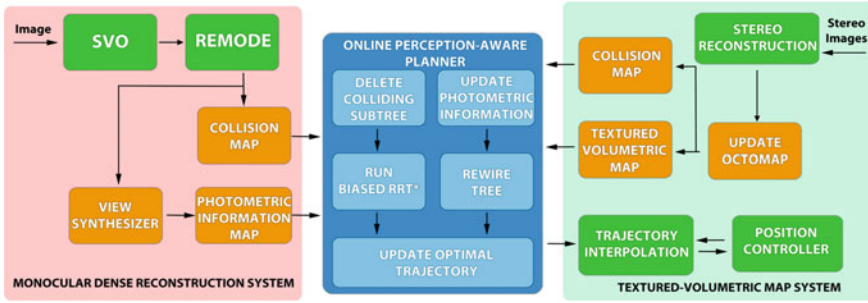


Fig. 5 Block diagram of the online perception-aware planning system

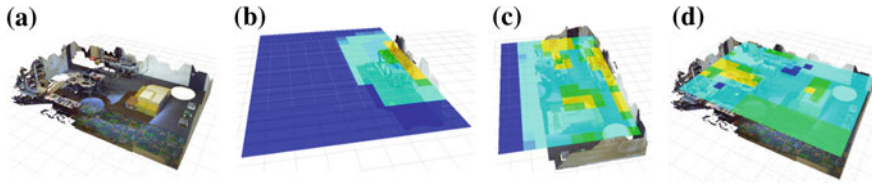
At each time step  $k$ , the quadrotor receives a new image to perform egomotion estimation. We use the Semi-direct monocular Visual Odometry (SVO) proposed in [2], which allows us to estimate the quadrotor motion in real-time. The computed pose  $\mathbf{T}_{k,w}$  and the relative image are then fed into the dense map reconstruction module (REMODE [28], a probabilistic, pixelwise depth estimate to compute dense depthmaps). Afterwards, the dense map provided by the reconstruction module is sent to the path planning pipeline and is used to update both the collision map (using Octomap [26]) and the photometric information map. The last one is then used to update  $\mathbf{A}_v$  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 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 the Textured OctoMap as in Sect. 4. 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 5 cm resolution, and with the input images downsampled by a factor of 4 to  $188 \times 120$ , and throttled down to 1 Hz. However, we evaluate this system in simulation, and for the experiments in Sect. 6.2, we run the simulation, visual pipeline, planner, and control software all on a laptop with an Intel i7.

## 6 Experiments

### 6.1 Real World Experiments

We motivate our approach by discussing how the photometric information distribution changes over time when exploring an unknown environment. Figure 6 shows the

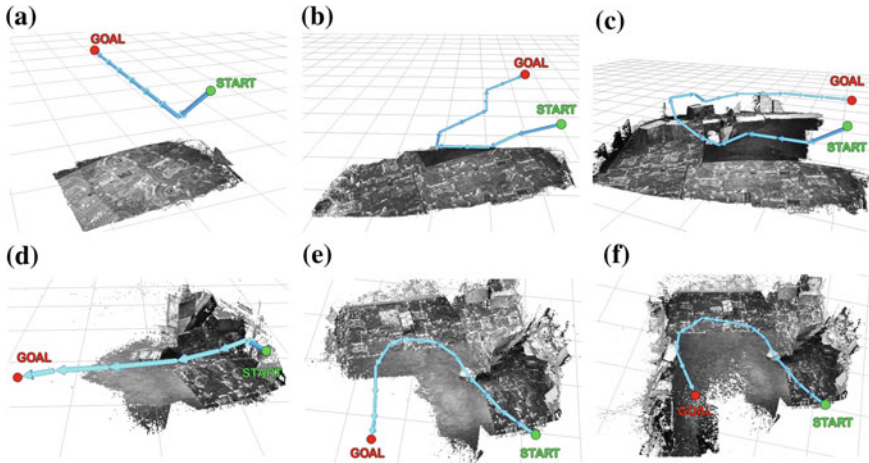


**Fig. 6** Computed photometric information gain at different exploration stages (**b**, **c** and **d**) for the scene in **a**. Warm (*yellowish*) colors refer to camera viewpoints exhibiting a higher amount of texture, while the cool (*bluish*) ones indicate less informative areas

map for the photometric information gain at different exploration stages. In Fig. 6b the almost unexplored scene has very little valuable information to compute a reliable plan. Standard planners, which calculate trajectories only once without performing online updates, compute sub-optimal plans or even collide with undiscovered objects. Hence, an online approach is needed to re-plan as new photometric information is gathered (see Fig. 6c, d).

To evaluate the proposed online perception-aware path planning, we ran experiments on an indoor environment with different configurations. We set up two scenarios with different object arrangements to vary the texture and the 3D structure of the scene. In the first scenario, the monocular camera on the MAV is downward-looking, while in the last one we choose a front-looking monocular 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 and boxes along the walls, while the floor in the center of the room is left without texture (i.e., with a uniform color). In the first scenario, we also put an obstacle in the center of the room. At the beginning of the exploration, the planner shows similar behavior in all the experiments (see Fig. 7a, d). 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 plan is updated by preferring areas with high photometric information (cf. Fig. 7b, c). In the second scenario, a front-looking camera provides photometric information about areas distant from the current MAV pose. As a consequence, we obtain an optimal plan earlier with respect to the first experiment (see Fig. 7e, f).



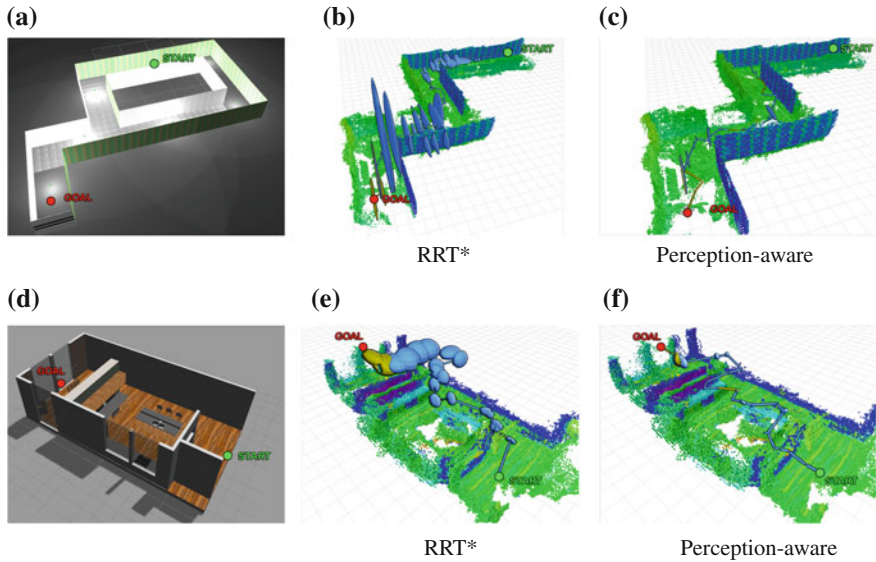
**Fig. 7** Experimental results in two real scenarios (rows). The first column shows the initially computed trajectories, only having little information of the environment available. The second and third column demonstrate the update of the plan as new information is gathered by updating the scene

## 6.2 Simulated Experiments

We demonstrate the proposed system in a simulated environment, using the components described in Sect. 5. 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 Fig. 8a) and is asked to reach the goal location that is located at 25 m from the start location. In the kitchen world (see Fig. 8d), the MAV begins at a position that is separated by two walls from the goal location, which is 12.5 m away. We compare the performance of the standard RRT\* planner and our perception-aware planner in Figs. 8 and 9.

## 6.3 Discussion

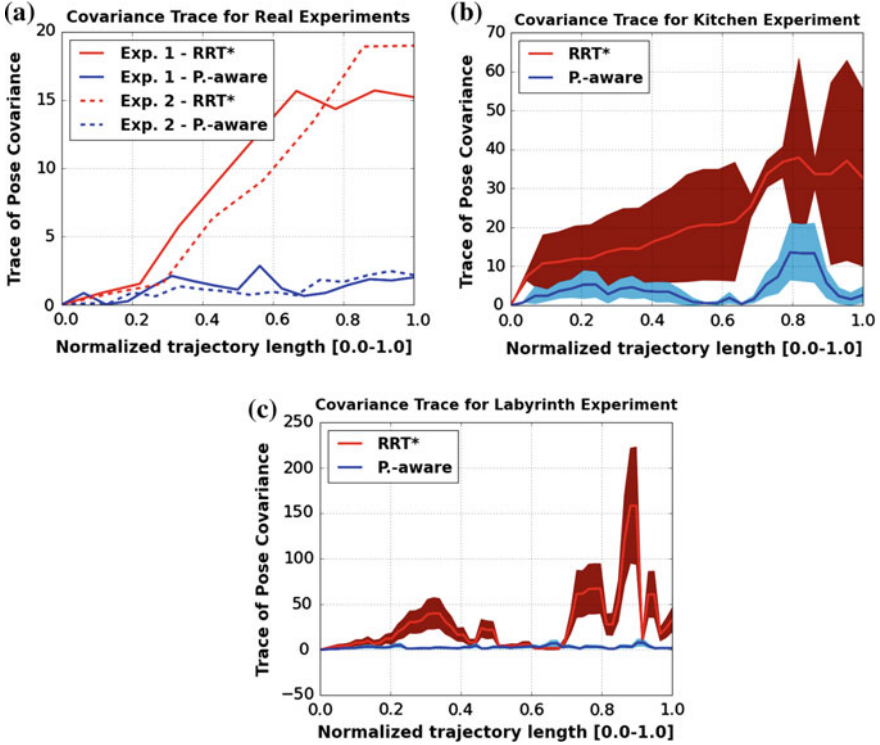
The qualitative results shown for the real world (Fig. 7) and simulated (Fig. 8) experiments show that the perception aware planner does indeed choose trajectories that allow the MAV to observe more photometric information. Quantitatively, this results in a dramatic improvement in the uncertainty of the vehicle’s pose estimate. The results in Fig. 9 show that the pose uncertainty, measured as the trace of the covari-



**Fig. 8** Exploration trial in the *labyrinth* (a) and in the *kitchen* (d) simulated environments. The trajectories computed by the RRT\* planner are shown in **b** for the labyrinth scenario and in **e** for the kitchen, while the ones computed with the perception aware planner are shown in **c** and **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 plan in *blue*

ance matrix and visualized as ellipses in Fig. 8, 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 718.3 and 715.2 s, respectively, and for *kitchen* it took 578.3 and 580.4 s, respectively. The results are shown in Fig. 8b, c for the labyrinth tests and in Fig. 8e, 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 plan 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.



**Fig. 9** Quantitative results for our experiments showing the evolution of the MAV’s pose covariance during the planned trajectory. **a** shows results of the real world experiments. **b** and **c** show the simulated *kitchen* and *labyrinth* trials, respectively. The plans for each trial result in different length trajectories, so the length of each trajectory 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

## 7 Conclusions

We have proposed a novel approach for performing online path planning that leverages the photometric information in the environment to plan a path that minimizes the pose uncertainty of a camera-equipped MAV that is performing vision-based egomotion estimation. These advances include a perception-aware path planner and a textured volumetric map. This planning framework has been evaluated with real and simulated experiments, and both the qualitative and quantitative results support the conclusion that taking photometric information into account when planning significantly reduces a vision-controlled MAV’s pose uncertainty. Utilizing perception awareness will enable more robust vision-controlled flight in arbitrary environments.



## References

1. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: large-scale direct monocular SLAM. In: European Conference on Computer Vision (ECCV), pp. 834–849 (2014)
2. Forster, C., Pizzoli, M., Scaramuzza, M.: SVO: fast semi-direct monocular visual odometry. In: IEEE International Conference on Robotics and Automation (ICRA) (2014)
3. Weiss, S., Achtelik, M.W., Lynen, S., Achtelik, M.C., Kneip, L., Chli, M., Siegwart, R.: Monocular vision for long-term micro aerial vehicle state estimation: a compendium. *J. Field Robot.* **30**(5), 803–831 (2013)
4. Scaramuzza, D., Achtelik, M.C., Doitsidis, L., Fraundorfer, F., Kosmatopoulos, E.B., Martinelli, A., Achtelik, M.W., Chli, M., Chatzichristofis, S.A., Kneip, L., Gurdan, D., Heng, L., Lee, G.H., Lynen, S., Meier, L., Pollefeys, M., Renzaglia, A., Siegwart, R., Stumpf, J.C., Tanksanen, P., Troiani, C., Weiss, S.: Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments. *IEEE Robot. Autom. Mag.* **21**, 26–40 (2014)
5. Grzonka, S., Grisetti, G., Burgard, W.: A fully autonomous indoor quadrotor. *IEEE Trans. Robot.* **28**(1), 90–100 (2012)
6. Nieuwenhuisen, M., Droschel, D., Beul, M., Behnke, S.: Obstacle detection and navigation planning for autonomous micro aerial vehicles. In: 2014 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1040–1047. IEEE (2014)
7. Bry, A., Roy, N.: Rapidly-exploring random belief trees for motion planning under uncertainty. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 723–730 (2011)
8. Bachrach, A., Prentice, S., He, R., Henry, P., Huang, A.S., Krainin, M., Maturana, D., Fox, D., Roy, N.: Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments. *Int. J. Robot. Res.* **31**(11), 1320–1343 (2012)
9. Sim, R., Roy, N.: Global a-optimal robot exploration in slam. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 661–666, April 2005
10. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **30**(7), 846–894 (2011)
11. Chen, S., Li, Y., Kwok, N.M.: Active vision in robotic systems: a survey of recent developments. *Int. J. Robot. Res.* **30**(11), 1343–1377 (2011)
12. Soatto, S.: Actionable information in vision. In: Cipolla, R., Battiato, S., Farinella, G.M. (eds.) *Machine Learning for Computer Vision*, pp. 17–48. Springer, Heidelberg (2013)
13. Davison, A.J., Murray, D.W.: Simultaneous localization and map-building using active vision. *IEEE Trans. Pattern Anal. Mach. Intell.* **24**(7), 865–880 (2002)
14. Sadat, S.A., Chutskoff, K., Jungic, D., Wawerla, J., Vaughan, R.: Feature-rich path planning for robust navigation of MAVs with mono-SLAM. In: IEEE International Conference on Robotics and Automation (ICRA) (2014)
15. Achtelik, M.W., Lynen, S., Weiss, S., Chli, M., Siegwart, R.: Motion-and uncertainty-aware path planning for micro aerial vehicles. *J. Field Robot.* **31**(4), 676–698 (2014)
16. Kim, A., Eustice, R.M.: Perception-driven navigation: active visual slam for robotic area coverage. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 3196–3203. IEEE (2013)
17. Irani, M., Anandan, P.: All about direct methods. In: *Vision Algorithms: Theory and Practice*, pp. 267–277. Springer (2000)
18. Newcombe, R.A., Lovegrove, S.J., Davison, A.J.: DTAM: dense tracking and mapping in real-time. In: IEEE International Conference on Computer Vision (ICCV), pp. 2320–2327 (2011)
19. Ferguson, D., Kalra, N., Stentz, A.: Replanning with rrt\*. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 1243–1248. IEEE (2006)
20. Boardman, B.L., Harden, T.A., Martinez, S.: Optimal kinodynamic motion planning in environments with unexpected obstacles. Technical report, Los Alamos National Laboratory (LANL) (2014)

21. Otte, M., Frazzoli, E.: RRT-X: real-time motion planning/replanning for environments with unpredictable obstacles. In: International Workshop on the Algorithmic Foundations of Robotics (WAFR) (2014)
22. Schmid, K., Lutz, P., Tomić, T., Mair, E., Hirschmüller, H.: Autonomous vision-based micro air vehicle for indoor and outdoor navigation. *J. Field Robot.* **31**(4), 537–570 (2014)
23. Barfoot, T.D., Furgale, P.T.: Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Trans. Robot.* **30**, 679–693 (2014)
24. Meilland, M., Comport, A.I.: On unifying key-frame and voxel-based dense visual SLAM at large scales. In: IEEE International Conference on Intelligent Robots and Systems (IROS) (2013)
25. Haner, S., Heyden, A.: Optimal view path planning for visual slam. In: Heyden, A., Kahl, F. (eds.) *Image Analysis*, pp. 370–380. Springer, Heidelberg (2011)
26. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: Octomap: an efficient probabilistic 3d mapping framework based on octrees. *Auton. Robot.* **34**(3), 189–206 (2013)
27. Mason, J., Ricco, S., Parr, R.: Textured occupancy grids for monocular localization without features. In: IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China (2011)
28. Pizzoli, M., Forster, C., Scaramuzza, D.: REMODE: probabilistic, monocular dense reconstruction in real time. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 2609–2616. IEEE, May 2014