Autonomous Flights through Image-defined Paths

Tien Do, Luis C. Carrillo-Arce, and Stergios I. Roumeliotis

Abstract This paper addresses the problem of autonomous quadrotor navigation through a previously-mapped indoor area. In particular, we focus on the case where a user walks through a building and collects images. Subsequently, a visual map of the area, represented as a graph of linked images, is constructed and used for automatically determining visual paths (i.e., sequences of images connecting the start to the end image locations specified by the user). The quadrotor follows the desired path by iteratively (i) determining the desired motion to the next reference frame, (ii) controlling its roll, pitch, yaw-rate, and thrust, and (iii) appropriately switching to a new reference image. For motion estimation and reference-image switching, we concurrently employ the results of the 2pt and the 5pt RANSAC to distinguish and deal with both cases of sufficient and insufficient baseline (e.g., rotation in place). The accuracy and robustness of our algorithm are evaluated experimentally on two quadrotors navigating along lengthy corridors, and through tight spaces inside a building and in the presence of dynamic obstacles (e.g., people walking).

1 Introduction and Related Work

In order for a quadrotor to autonomously navigate within a *known*, GPS-denied area (e.g., indoors), it must be able to find where it is, determine the path towards its goal, and control itself to follow the desired trajectory. One way to solve this problem would be to construct, typically offline, a metric 3D map of the environment that the quadrotor will then use to (i) identify where it is (i.e., localize), and (ii) compute a path towards its destination. Creating *dense* 3D maps that can be used for path planning, however, is quite challenging, especially for large buildings, due to the computational cost of the mapping process.

The authors are with the University of Minnesota, Minneapolis, MN 55455.
Emails:{doxxx104|carrillo}@umn.edu, stergios@cs.umn.edu
This work was supported by the AFOSR (FA 9550-10-1-0567)

An alternative approach to this problem, is to represent a building using visual data collected beforehand, and describe a path as *a sequence of images* that the quadrotor needs to follow in order to reach its destination. The main advantages of an image-space representation of a path within a building are *scalability* (no metric global map needs to be constructed) and *ease of implementation* (the images can be collected by a person walking throughout the building with, e.g., a cell phone). On the other hand, though, controlling a quadrotor to follow a visual path becomes significantly more challenging due to the lack of scale and geometric information in the reference trajectory.

Controlling a robot to reach a specific destination defined in the image space can be achieved using visual servoing (VS) [8, 9]. Most VS approaches can be classified into two categories: (i) Position-based VS (PBVS), where the control input is computed directly using a relative position, up to scale, and orientation (pose) estimate; and (ii) Image-based VS (IBVS), where the control input is determined in the image domain, while often it is assumed that the depth to the scene is, at least approximately, constant [8]. Prior work on VS for quadrotors equipped with a downward-pointing camera has addressed the problem of landing on a known target [24, 7] and hovering over an arbitrary target [5]. Furthermore, for quadrotors equipped with a forward-pointing camera, [6] classifies the environment into corridors, stairs, or "other" in order to determine the appropriate turn, side-ways, or upward motion so that the robot can continue exploration.

In the context of navigating along a visual path, VS techniques have been employed for *ground* robots (e.g., [25, 14, 10, 13]), while some of these techniques ([10, 13]) have been applied to aerial vehicles [26, 12]. In particular, in [26] an extension of the "funnel"-lane concept of [10] to 3D is presented and applied to controlling a quadrotor. Specifically, the geometric constraints based on the image coordinates of the reference features are used for determining the funnel region within which the robot should be moving in order to match the reference image. Then, the desired motion of the quadrotor is computed as the convex combination of the heading/height required for staying within the funnel region and the one the quadrotor had followed during the training phase. As criterion for switching to the next reference image, an error measure is defined based on the root mean square of the difference in the feature's pixel coordinates between the reference and the current image. In [12], the VS method of [13] is extended to the case of a quadrotor following a visual path comprising a sequence of keyframe images selected, during the experiment, by a person. In contrast to 3-view-geometry-based approaches (e.g., [14] and [18]), [12] uses the PBVS algorithm described in [9] for controlling the quadrotor. This method does not require triangulating points but instead, given sufficient baseline, it uses epipolar geometry for estimating the relative pose between the current and the reference camera frames.

A key limitation of both [26] and [12] is that they cannot deal with rotations in place (often required for navigating through tight spaces), or, for the case of [12], with translations through areas with only faraway features (e.g., featureless corridors). Moreover, in both cases, the quadrotor followed rather short and fairly simple (in terms of the motions required) paths comprising a short translation and a wide

turn in [26], or no turn in [12], where the quadrotor was moving back and forth between two locations connected via a direct path.

In this work, our objective is to enable a quadrotor to autonomously navigate inside a large-scale building by following a pre-recorded sequence of images that correspond to long ($\sim$ 75 m) and challenging (in terms of the motions involved and the type of scenes encountered) paths. To do so, our PBVS method *concurrently* minimizes the relative heading and baseline between the current and the reference images. In particular, we match features between these two images and use the 2pt[1] and the 5pt RANSACs' estimates to determine the state of the system and control the quadrotor.

The key advantages of the proposed PBVS algorithm are as follows: (i) We employ a geometry-based algorithm which computes and compares the 2pt versus the 5pt RANSAC inliers for selecting the next reference image. In contrast, [26, 12] rely on the features' pixel coordinates, which are unreliable when dealing with features at various depths; (ii) Our algorithm is capable of dealing with a wide range of conditions, such as motion along lengthy corridors, open spaces, and rotations in place, and in environments where the appearance-based feature matching may return unreliable results; (iii) Our approach does not require recording the images by manually controlling the robot through the reference paths as is done in [26, 12]. Instead, one can easily define desired paths by simply walking through the area of interest carrying a cell phone or the quadrotor. Lastly and, in order to demonstrate the efficiency, accuracy, and robustness of the proposed algorithm, we have implemented it on two quadrotors: The Parrot Bebop [2] and the DJI F450, the latter carrying a cell phone. During testing, the quadrotors operated inside challenging environments (specular reflections, large lighting surfaces), comprising long paths, tight turns, and in the presence of dynamic obstacles.

2 Quadrotors and Objective

Both quadrotors have attitude-stabilization controllers, which take as input information from an observer that processes gyroscope and accelerometer measurements, from the onboard inertial measurement unit (IMU), to estimate the roll and pitch angles, yaw-rate, and thrust of the quadrotor. Additionally, each quadrotor carries a downward-pointing camera to estimate optical flow, and an ultrasonic sensor to measure the distance to the ground. Note that despite the availability of metric information from the velocity estimated based on the optical flow and the distance to the scene, we do not use it to triangulate features and create a local map as it can be

[1] The 2pt RANSAC estimates the relative orientation ${}^{I_1}_{I_2}\mathbf{R}$ between two images, I_1 and I_2, under the assumption of very small baseline compared to the depth of the scene. A closed-form solution for the 2pt minimal case is provided in 6.2, while the analytical solution for the least-squares solver is presented in [22]. The 5pt RANSAC [27] estimates the relative orientation ${}^{I_1}_{I_2}\mathbf{R}$ and the unit vector of translation ${}^{I_1}\mathbf{t}_{I_2}$ between two images I_1 and I_2.

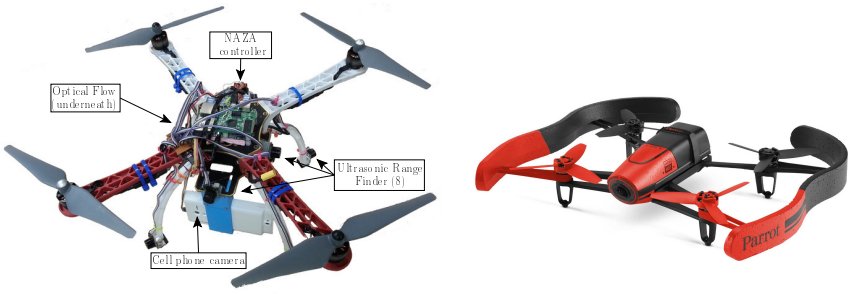

Fig. 1: The DJI F450 quadrotor (left) equipped with a NAZA controller, an optical-flow sensor, ultrasonic sensors, and a cell phone. The Parrot Bebop quadrotor (right) equipped with a 180 deg WFOV camera, an optical-flow sensor, and an ARM-based processor.

both unreliable[2] and computationally expensive. Furthermore, both quadrotors have access to (i) forward pointing wide field of view (WFOV) cameras (mounted in the front of the Bebop, or as part of the cell phone carried by the DJI) for collecting images and (ii) processors for executing in real time all image-processing and control algorithms necessary by the proposed PBVS method. Finally, and to increase safety, the DJI quadrotor carries 8 ultrasonic sensors spaced 45 deg apart and aligned with its arms and legs (see Fig. 1).

As mentioned earlier, the objective of this work is to develop a robust algorithm[3] that will allow the quadrotors to follow long and complex visual paths, defined as sequences of pre-recorded images between the start and final desired locations.

3 Technical Approach

Our approach comprises two phases. In the first (offline) phase, a visual-graph-based representation of the area of interest is constructed using images collected by a person walking through it. Then, given a start and an end pair of images, a feasible visual path is *automatically* extracted from the graph along with motion information (path segments that include significant translational motion or only rotations in place). In the second (online) phase, our PBVS algorithm controls the quadrotor to successively minimize the relative rotation and baseline between the images captured by its onboard camera and the corresponding reference images of the visual

[2] Under low-light conditions, the velocity measurements are reliable only for a fixed tilt angle of the vehicle. Note that when in motion, the quadrotor changes its roll and pitch which causes image blurriness (due to the increased exposure) and, hence, large errors in the optical-flow estimates.

[3] Note that although both the embedded controller and the cell phone contain IMUs, which can be used, in conjunction with the camera, to form a vision-aided inertial navigation system [19], in this work, we intentionally focus on a "light", in terms of processing, vision-only approach so as to assess its performance and use it as a baseline for future comparisons.

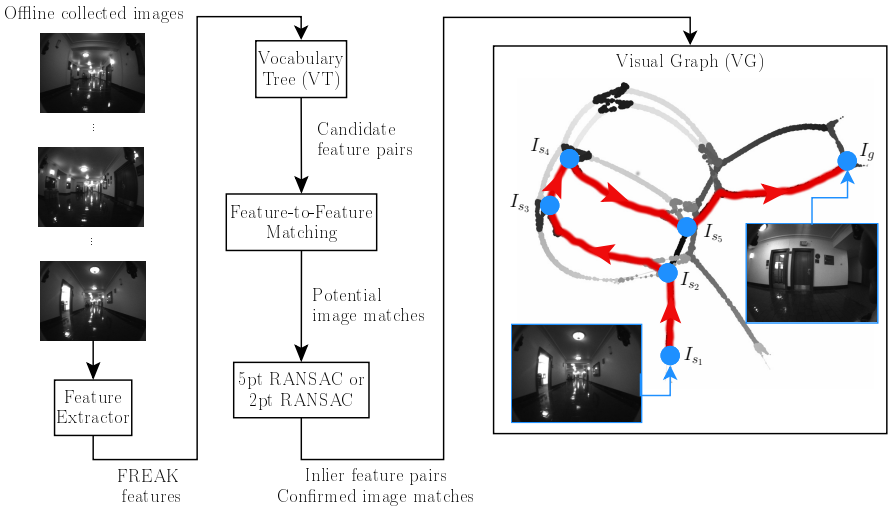

Fig. 2: Offline phase: The area of interest is described as a visual graph (VG) whose nodes correspond to images, while edges link images containing a sufficient number of common features for reliably visually-servoing between them. In the VG, I_{s1}, I_g denote the start and goal images, respectively, while $I_{s2}, \ldots, I_{s5}$ signify intermediate goal locations along the quadrotor's path specified by the user.

path. Additionally, and in order to increase robustness, our navigation approach employs a vocabulary tree (VT) [28] for relocalizing inside the previously-constructed visual graph when losing track of the reference image path. Lastly, we include an obstacle avoidance routine for the DJI so as to increase safety.

3.1 Offline phase

3.1.1 Map generation

A person carrying a cell phone, or a quadrotor, walks through the area of interest collecting images at 30 Hz. Subsequently, we extract FREAK image points [4] and employ a VT to generate the visual map which is represented as a visual graph (VG) whose nodes correspond to the recorded images. An edge between two images signifies that these were matched by the VT and at least 30 point-correspondences passed the 5pt or 2pt RANSAC. Furthermore, we assign *weights* to these edges inversely proportional to the number of common features (inlier matches) found between linked images. This choice is justified by the fact that the VG will be used to determine paths that the quadrotor can reliably navigate through in the image space towards its destination. This process is depicted in Fig. 2.

The VG is constructed in a matter of minutes even for large areas containing tens of thousands of images. Moreover, it can be easily updated by adding/replacing subsets of images corresponding to new/altered regions of a building.

3.1.2 Path specification

The VG is used for computing paths between the quadrotor's start and end locations, possibly via intermediate points. Specifically, consider the graph shown in Fig. 2. Assume that the quadrotor knows its current location (e.g., it is provided by the user, automatically determined using the VT, or saved from the previous run) corresponding to image node I_s. Then, the user specifies a destination image I_g in the VG and the reference path is determined automatically by employing Dijkstra's algorithm [11]. This process is easily extended to include intermediate locations (e.g., $I_{g_1}, I_{g_2} \dots I_{g_n}$), by simply resetting as the start of the next path segment the end image of the previous one (e.g., $I_{s_{i+1}} = I_{g_i},\ i = 1 \dots n$).

Once the path is extracted from the VG, we prune images that are very close to each other and only keep the ones that have substantial translational and/or rotational motion between them. To do so, we use an iterative process that starts from the reference image $I_1^r = I_s$ and moves along the path matching FREAK features using both the 5pt and 2pt RANSAC algorithms until it finds the first image, I_{s+m}, $m \geq 1$, that either has more 5pt than 2pt inliers, or the relative yaw angle between them is greater than 10 deg. In the first case, we declare that the quadrotor is in translation, otherwise, in rotation and set I_{s+m}, as the next reference image I_2^r. The resulting path $\mathscr{P} = \{I_1^r, I_2^r, \dots, I_n^r\}$ is provided to the quadrotor along with two additional pieces of information: (i) We specify which images correspond to rotation-only motion and provide the yaw angle between consecutive rotation-only images; (ii) We provide the FREAK features extracted from each reference image along with their coordinates. The former is useful in case the quadrotor gets lost (see Section 3.2.4), while the latter is used by the quadrotor for efficiently finding and matching its next reference image through the process described hereafter.

3.2 Online phase

3.2.1 System state determination

When there is sufficient baseline (as in [12]), and in order to minimize the relative motion between I_t (the image taken by the quadrotor's onboard camera at time-step t) and a reference image $I_k^r \in \mathscr{P}$, we use the 5pt RANSAC to estimate the 5 dof, ${}^{I_k^r}_{I_t}\hat{\mathbf{R}}, {}^{I_k^r}\hat{\mathbf{t}}_{t_t}$, desired motion. This estimate, however, is not reliable when the baseline between I_t and I_k^r is short [9]. Furthermore, the appearance-based feature matching between I_t and I_k^r (the 5pt RANSAC's input), is not always reliable (e.g., due to adverse lighting conditions and/or occlusions).

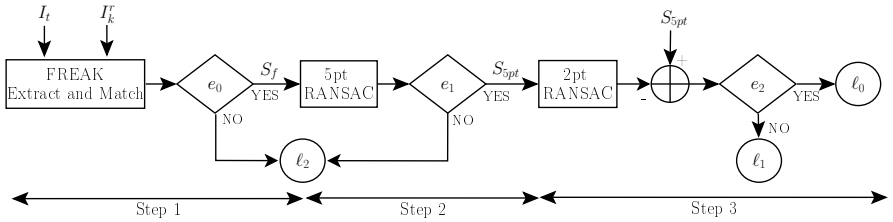

Fig. 3: Online: Schematic diagram of the steps and transitions between the different states of the automaton $\mathscr{H}$.

To address these challenges, we model our system as a hybrid automaton $\mathscr{H}$ as follows:

Definition 1: $\mathscr{H} = (\mathscr{L}, \mathbf{x}, \mathscr{E})$, where:

- $\mathscr{L}$ is the set of discrete states including:
  - ℓ_0: wide baseline (nominal condition)
  - ℓ_1: short baseline (rotation in place is necessary or reference-image switching)
  - ℓ_2: lost mode due to, e.g., motion overshoot or failure in the appearance-based feature matching.
- $\mathbf{x}(t,k)$ is the abstract state vector with elements $\mathbf{x}(t,k) = [I_t, I_k^r, \mathbf{r}(t,k)]$ where $\mathbf{r}(t,k)$ is the desired motion for minimizing the relative pose between I_t and I_k^r.
- $\mathscr{E}$ is the set of relations governing transitions between the states in $\mathscr{L} = \{\ell_0, \ell_1, \ell_2\}$.

Given $\mathscr{H}$, and in order to complete $\mathscr{P}$, the system must ideally iterate between two steps until the last element of $\mathscr{P}$ is reached: (i) In case of state ℓ_0, we compute the motion $\mathbf{r}$ and control the quadrotor so as to bring the system to state ℓ_1 (see Section 3.2.2); (ii) When at ℓ_1, and if there is no significant rotation between I_t and I_k^r, we switch I_k^r to the next reference image in $\mathscr{P}$ (see Section 3.2.3), and the system returns to state ℓ_0. In case of external disturbances, the system may reach state ℓ_2. In this case, a recovery procedure will be executed to attempt to bring the system back to ℓ_0 or ℓ_1 (see Section 3.2.4).

In order to accurately classify the state of the system as ℓ_0, ℓ_1, or ℓ_2 based on I_t and I_k^r, we use the process summarized in Fig. 3, and define the relations in $\mathscr{E} = \{e_0, e_1, e_2\}$ in the following 3 steps.

Step 1: We first extract and match (Hamming distance between binary descriptors less than 60) FREAK features in I_t and I_k^r, and define as $S_f(I_t, I_k^r)$ the set of all feature correspondences. Note that if the condition for sufficient feature matches $e_0 : |S_f| \geq 25$, where $|S_f|$ is the cardinality of the set S_f, is satisfied, then the system proceeds to Step 2 of the current state, else it transitions to state ℓ_2 (see Fig. 3).

Step 2: Given the bearing vectors, ${}^{I_t}\mathbf{b}_f$ and ${}^{I_k^r}\mathbf{b}_f$, from both camera frames, I_t and I_k^r, to each feature f, we employ the 5pt RANSAC to compute the geometric constraint $({}^{I_k^r}_{I_t}\hat{\mathbf{R}}, {}^{I_k^r}\hat{\mathbf{t}}_{I_t})$ between I_t and I_k^r, as well as the set of features whose reprojection error [20] is within a threshold ε_1 (the error tolerance for outlier rejection [16]). At

this point, we require that the condition $e_1 : |S_{5pt}| \geq 25$ (i.e., the number of 5pt inliers is no less than 25; see [15] for a probabilistic justification) is satisfied in order to proceed to Step 3; else the system transitions to state ℓ_2 (see Fig. 3).

In Step 3, we distinguish between the states ℓ_0 and ℓ_1. Specifically, when the baseline is short (i.e., ${}^{I^r_k}d_{I_t} \ll {}^{I_t}d_f, {}^{I^r_k}d_f \Leftrightarrow {}^{I^r_k}\mathbf{b}_f \simeq {}^{I^r_k}_{I_t}\mathbf{R}\,{}^{I_t}\mathbf{b}_f$), the 5 degrees of freedom (dof) epipolar constraint:

$$ {}^{I^r_k}\mathbf{b}_f^T \lfloor {}^{I^r_k}\mathbf{t}_{I_t} \times \rfloor {}^{I^r_k}_{I^r_t}\mathbf{R}\,{}^{I_t}\mathbf{b}_f = 0 \tag{1} $$

degenerates into a 3 dof, rotation-only constraint that is satisfied by all the 5pt inliers. Our algorithm uses this observation to determine if there is sufficient baseline between the current, I_t, and reference, I^r_k, images. In particular, we employ the 2pt RANSAC on the features $f \in S_{5pt}$ to compute the rotation ${}^{I^r_k}_{I^r_t}\breve{\mathbf{R}}$ between two images and determine $S_{2pt} = \{f \in S_{5pt} \mid 1 - {}^{I^r_k}\mathbf{b}_f^T\, {}^{I^r_k}_{I^r_t}\breve{\mathbf{R}}\, {}^{I_t}\mathbf{b}_f < \varepsilon_2\}$, which is the subset of 5pt inliers that are also 2pt inliers. Lastly, and in order to compensate for the noise in the measurements and the randomness of RANSAC, instead of requiring $|S_{2pt}| = |S_{5pt}|$, we employ the condition $e_2 : \frac{|S_{2pt}|}{|S_{5pt}|} > 0.94$ to declare small baseline (i.e., state ℓ_1).

Depending on the state of our system (ℓ_0, ℓ_1, or ℓ_2), in what follows, we describe the process for controlling the quadrotor.

3.2.2 Wide baseline (ℓ_0)

Improving the motion estimate

In practice, when the quadrotor navigates through long corridors or open spaces, S_f may contain features at various depths, some of which, typically the faraway ones, may negatively affect the motion estimate. Note that such features, satisfy the 2pt RANSAC. To remove them, we define as $S'_{5pt} = S_{5pt} \setminus S_{2pt}$, run again the 5pt RANSAC on the features $f \in S'_{5pt}$, and use the winning hypothesis to initialize an iterative batch-least squares algorithm [23] to improve the accuracy of the estimated desired motion between I_t and I^r_k.

At this point, we note that although the desired motion between I_t and I^r_k may comprise 5 dof (3 for the relative roll, pitch, yaw and 2 for the unit vector, $\mathbf{t}$, of translation), given the kinematic and actuation constraints of the quadrotor (e.g., it cannot achieve non-zero roll or pitch angle while staying still), our controller seeks to match the desired motion only along 3 dof: The t_x, t_y projection of the desired unit vector, $\mathbf{t}$, of translation on the horizontal plane,[4] and the desired (relative) yaw angle ${}^{I^r_k}\hat{\psi}_{I_t}$. Moreover, and in order to maintain an almost constant-velocity flight, we scale t_x and t_y by v_0 (the maximum velocity that the optical-flow sensor can measure) and provide our controller with the following desired motion vector:

[4] Note that since all images were recorded at about the same height, the z component of the desired motion estimate is rather small after the first reference image and we subsequently ignore it. Instead, we use the distance-to-the-ground measurements to maintain a constant-altitude flight.

$$\mathbf{r} = \begin{bmatrix} v_x^d \\ v_y^d \\ \hat{\psi} \end{bmatrix} = \begin{bmatrix} t_x \; v_0 \\ t_y \; v_0 \\ I_k^r \, \hat{\psi}_{I_t} \end{bmatrix} \tag{2}$$

Note that this desired motion vector will need to be appropriately modified in the presence of obstacles (see Section 3.2.5).

Controller

In order to determine the control input, $\mathbf{u}_k(t)$ (roll, pitch, yaw-rate, and thrust), that we must provide to the quadrotor's attitude controller so as to achieve the desired velocity, we employ the vehicle's kinematic equations, linearized about the equilibrium (near hover condition - see [15]):

$$\begin{bmatrix} \dot{v}_x(t) \\ \dot{v}_y(t) \end{bmatrix} = g \begin{bmatrix} \theta(t) \\ -\phi(t) \end{bmatrix} \tag{3}$$

$$\ddot{z}(t) = \frac{1}{m}\tau(t) - g \tag{4}$$

where g is the gravity, m is the quadrotor's mass, and $\phi(t)$, $\theta(t)$, and $\tau(t)$ are the roll, pitch, and thrust of the quadrotor in ego-centric coordinates, respectively.

To compute the velocity error, we use the estimates, $\hat{v}_x, \hat{v}_y$, from the optical-flow sensor, to form:

$$\begin{bmatrix} e_{v_x}(t) \\ e_{v_y}(t) \end{bmatrix} = \begin{bmatrix} v_x^d(t) - \hat{v}_x(t) \\ v_y^d(t) - \hat{v}_y(t) \end{bmatrix} \tag{5}$$

Furthermore, the height error, e_z, is defined as the difference between the desired altitude and the estimated height $\hat{z}$ from the downward-pointing ultrasonic sensor:

$$e_z(t) = z^d(t) - \hat{z}(t) \tag{6}$$

Lastly, based on (4), (5), (6) and $\hat{\psi}$ in (2), we form a PID controller that computes the desired control input to the system as:

$$\mathbf{u}_k(t) = \begin{bmatrix} \theta^d(t) \\ \phi^d(t) \\ \tau^d(t) \\ \dot{\psi}^d(t) \end{bmatrix} = \begin{bmatrix} k_{p,v_x} e_{v_x}(t) + k_{i,v_x} \int e_{v_x}(t)dt \\ -k_{p,v_y} e_{v_y}(t) - k_{i,v_y} \int e_{v_y}(t)dt \\ k_{p,z} e_z(t) + k_{i,z} \int e_z(t)dt + k_{d,z} \dot{e}_z(t) \\ -k_{p,\psi} \hat{\psi} \end{bmatrix} \tag{7}$$

The gains k_p, k_i, and k_d that ensure high response, zero tracking error, and robustness were found as described in [17].

Fig. 4, describes the 3-control-loop implementation of our algorithm on the DJI F450 quadrotor. The outer loop runs at 7.5 Hz and determines the desired 2D velocity, v_x, v_y, and the yaw $\hat{\psi}$ (see Section 3.2.2). The desired velocity and height control loop (middle loop) runs at 50 Hz and provides the roll, pitch, and thrust setpoints to the attitude controller (see [15] for more details).

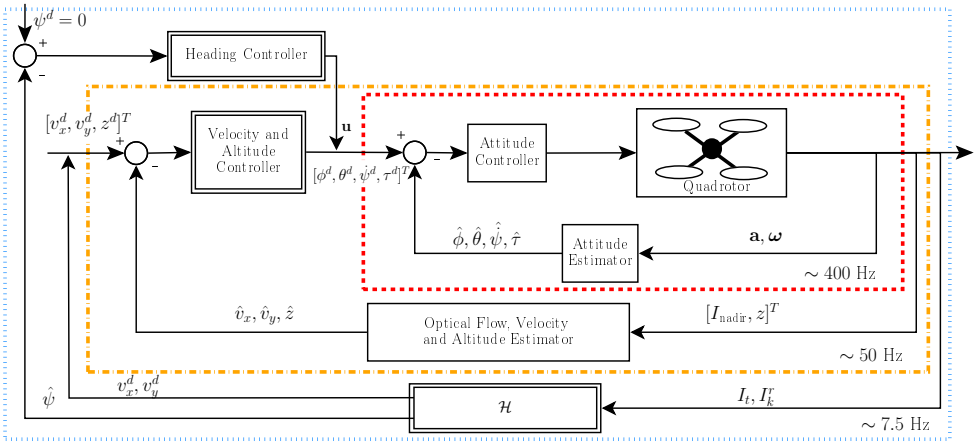

Fig. 4: System block diagram. The double-line blocks denote components of our algorithm described in Sections 3.2.1 ($\mathscr{H}$) and 3.2.2 (Controller).

3.2.3 Short baseline (ℓ_1)

In case of short baseline, we detect if there is any rotational motion needed to minimize the relative yaw, ${}^{I_k^r}\psi_{I_t}$, between I_t, and I_k^r. To do so, we first improve the rotation matrix estimate, ${}^{I_k^r}_{I_t}\breve{\mathbf{R}}$, by employing the least-squares method of [22] on the features $f \in S_{2pt}$ using as initial estimate, the one from the minimal solver of the 2pt RANSAC. After extracting the yaw component, if $|{}^{I_k^r}\breve{\psi}_{I_t}| > \tau_3$,[5] we send the desired rotation-in-place motion $\mathbf{r}^T = [0 \;\; 0 \;\; {}^{I_k^r}\breve{\psi}_{I_t}]^T$ to the controller to minimize the relative yaw between I_t, and I_k^r; else we switch to the next reference image in the path $\mathscr{P}$.

Note that when we have direct access to the attitude estimator, as in the case of the Bebop, we can leverage the yaw angle (computed off-line - see Section 3.1.2) between the first and last rotation-only reference images to speed up the execution of this path segment. Specifically, the precomputed relative yaw angle is provided to the controller to perform a "blind" rotation in place. Once this is complete, the quadrotor queries the VT to confirm that the last rotation-only reference image has been reached or determine the remaining rotation between the current image and the last rotation-only reference image.

3.2.4 Lost mode (ℓ_2)

When the quadrotor loses track of the current reference image, we refer to the last reference image where it computed good matches as I_{lost}^r. There are four possible scenarios that can cause the quadrotor to get lost:

- The robot enters a featureless region.

[5] This threshold depends on the onboard camera's fov and is selected so as to ensure a significant overlap (more than 80%) between the current camera image and the next reference image.

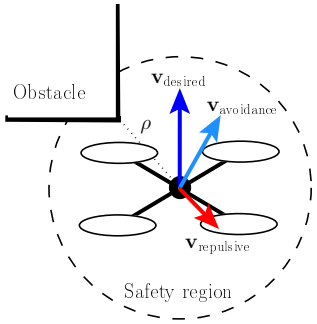

Fig. 5: Schematic of the velocity determination for obstacle avoidance.

- The robot enters a region where the result from the FREAK feature matching between I_t and I_k^r is unreliable.
- The quadrotor significantly deviates from its current path, in order to avoid an obstacle.
- Dynamic obstacles (e.g., people) obstruct the quadrotor's view or path.

Our recovery method is as follows: While hovering, the quadrotor queries the VT with I_t and successively evaluates among the returned images to find the one that has at least 35 features in common with I_t that pass the 5pt RANSAC. If the above search fails for the top 10 images, the quadrotor switches to a "blind" motion strategy following the same type of motion as before it was lost (i.e., translation or rotation based on I_{lost}^r) for 0.5 sec and then attempts again to retrieve a good reference image I_{best}^r. This iterative process is repeated for 10 times before declaring that the quadrotor is lost, in which case, it autonomously lands.

3.2.5 Obstacle detection and avoidance

To avoid collisions while following the reference path, we combine the desired motion from the PBVS algorithm with a "repulsive" velocity defined using the ultrasonic sensors. Specifically. we denote as $\rho = 1$ m the radius of the safety region centered around the quadrotor, $o_k(t)$ the measurement of the k^{th} ultrasonic sensor, and $\boldsymbol{\xi}_k$ the 2D unit vector of direction of this ultrasonic sensor relative to the quadrotor. Let γ be a constant relative-distance-to-velocity gain, we then construct a 2D repulsive velocity vector as:

$$\mathbf{v}_{\text{repulsive}}^k(t) = \begin{cases} \mathbf{0} & \text{if } o_k(t) \geq \rho \\ -\gamma * (\rho - o_k(t)) * \boldsymbol{\xi}_k & \text{if } o_k(t) < \rho \end{cases}$$

and set as: $\mathbf{v}_{\text{avoidance}}(t) = \mathbf{v}_{\text{desired}}(t) + \sum_{k=1}^{8} \mathbf{v}_{\text{repulsive}}^k(t)$

4 Experimental Results

We performed experiments with two quadrotors (the Parrot Bebop and the DJI F450) in two different scenarios: Firstly, inside a motion-capture room to evaluate the accuracy of our approach using ground truth from a VICON motion-capture system [3]. Then, in a large indoor area to evaluate the algorithm's performance under challenging conditions.

4.1 System setup

The DJI F450 quadrotor is equipped with a NAZA attitude controller, a PX4Flow [21] for height and velocity measurements, 8 MaxBotix ultrasonic range finders, and Google's Tango smartphone. The phone has a built-in 180 deg fisheye camera and a quad-core ARM processor. Note that we do not use the built-in estimator of this device. The quadrotor also carries: (i) An Arduino microcontroller to generate the signals required to operate the quadrotor and switch between manual and autonomous mode, (ii) An ODroid-U3 ARM-based computer used for allowing the cell phone and the Arduino to communicate, and (iii) A wireless router for debugging during test flights. It should be pointed out that the ODroid-U3 and the router are only used for debugging purposes; all computations are performed on the smartphone.

The Bebop, on the other hand, carries a MEMS IMU, a downward-pointing Aptina MT9V117 camera used for optical flow, an ultrasonic sensor for measuring the distance to the ground, and a forward-pointing Aptina MT9V002 camera which we use for visual navigation. All processing is carried onboard Bebop's ARM Cortex A9 800 MHz dual-core processor.

4.2 Short experiment with ground truth

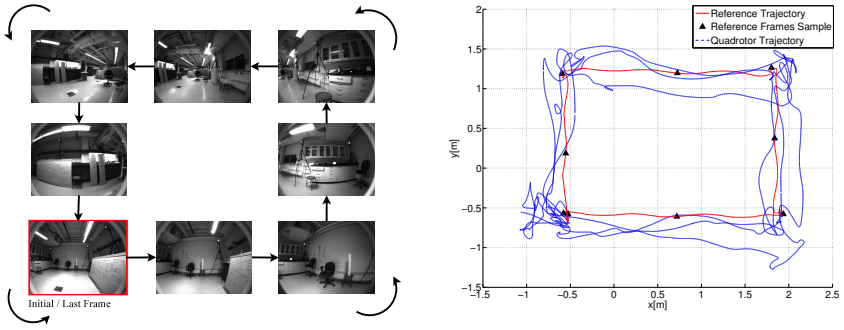

Fig. 6: Short experiment with ground truth: (left) Sample of camera frames used as reference images; (right) Comparison between the reference trajectory and the actual quadrotor trajectory.

In this experiment, we recorded an image sequence describing a rectangular path (approximately 2×2.5 m) within the motion-capture room while recording the

ground truth for the phone's camera motion. The path was created such that the poses of the initial and the last frame coincide. Then, we ran our PBVS algorithm three times on the image sequence while recording the quadrotor's pose with the VICON. Fig. 6 (left) shows a subset of the reference camera images along the path, while Fig. 6 (right) depicts the reference trajectory, the location of the reference images of Fig. 6 (left), and the path followed by the quadrotor. As evident, the error between the actual and the reference trajectories is typically within ± 0.5 m. Similar performance was achieved when using the Bebop quadrotor.

4.3 Long experiments

These experiments took place in the Walter Library's basement which is a challenging environment with numerous specular reflections on the floor, where the quadrotors had to follow a 75 m long path comprising translational motion segments through open spaces as well as rotations in place in order to navigate through narrow passages. Some of these maneuvers were in front of areas where most of the visible scene was behind a coffee-shop glass front whose reflections posed a significant challenge to the motion-estimation algorithm. Fig. 7 shows the blueprint of the experimental area depicting the reference visual path (red bold line), and snapshots of two quadrotors in flight.

During the experiment, the Bebop was able to complete the reference trajectory in 240 sec, at an average speed of 40 cm/sec, despite getting lost and recovering its path twice. On the other hand, it took the F450 quadrotor 450 sec (average speed of 20 cm/sec) to complete approximately the same path after getting lost 3 times. The difference in the performance of the two quadrotors is mainly due to the lower maximum-velocity sensing capabilities of the PX4Flow sensor onboard the F450 but also because of the agility and safety (no sonars were required) that the smaller size Bebop quadrotor provided. The videos for the Bebop and F450 experiments are available at [1].

5 Conclusion and Future Work

In this paper, we presented a visual-servoing algorithm that allows quadrotors to autonomously navigate within a previously-mapped area. In our work, the map is constructed offline from images collected by a user walking though the area of interest and carrying a cell phone or a quadrotor. Specifically, and in order to increase efficiency, the visual map is represented as a graph of images linked with edges whose weights (cost to traverse) are inversely proportional to the number of features common to them. Once the visual graph is constructed, and given as input the start and goal location of the quadrotor, it automatically determines the desired path as a sequence of reference images. This information is provided to the quadrotor, which estimates in real time the motion that minimizes the difference between its current and reference images, and controls its roll, pitch, yaw-rate, and thrust for achieving that.

Besides the ease of path-specification, a key advantage of our approach is that by employing a mixture of 2pt and 5pt RANSAC for determining the type of motion

Fig. 7: Long experiment: Blueprint of the experimental area, reference path for the DJI F450 (1-4-5-6-7-8-1) and the Parrot Bebop (1-4-5b-6b-7-8-1), and snapshots of both quadrotors along their paths (1-8).

required (rotational, translational with close-by or faraway scene), and for selecting, on the fly, the next reference image, it is able to navigate through areas comprising featureless corridors, as well as narrow passages. Moreover, it is able to cope with static and moving obstacles and, in many cases, recover its path after losing track of its reference image. The accuracy of the proposed algorithm was assessed using motion-capture data within a small area, while its robustness to lighting conditions and in-place rotations was demonstrated by autonomously navigating along a 75 m path through a large building. Lastly, and as part of our ongoing work, we are currently extending our algorithm to combine visual and inertial measurements [19] that will improve the accuracy of the velocity estimates, and, thus, allow us to further increase the quadrotors' operational speed.

6 Appendix

6.1 Singular condition of the 5pt RANSAC minimal solver

Consider a feature f appearing both in I_t and I_k^r which satisfies the following geometric constraint:

$$ {}^{I_k^r}d_f \, {}^{I_k^r}\mathbf{b}_f = {}^{I_t}d_f \, {}^{I_k^r}_{I_t}\mathbf{R} \, {}^{I_t}\mathbf{b}_f + {}^{I_k^r}d_{I_t} \, {}^{I_k^r}\mathbf{t}_{I_t} \tag{8} $$

Note that when sufficient baseline exists between the current and reference images, (8) can be projected on the normal to the epipolar plane to yield the epipolar constraint

$$ {}^{I_k^r}\mathbf{b}_f^T \lfloor {}^{I_k^r}\mathbf{t}_{I_t} \times \rfloor \, {}^{I_k^r}_{I_t}\mathbf{R} \, {}^{I_t}\mathbf{b}_f = 0 \tag{9} $$

By employing five pairs of features that satisfy (9), we can estimate the desired 5 dof motion using the 5pt RANSAC algorithm [27].

On the other hand, consider the case when the relative position between I_t and I_k^r is significantly smaller compared to either distance to the feature f. Without loss of generality, we assume that ${}^{I_k^r}d_{I_t} \ll {}^{I_t}d_f \Rightarrow \frac{{}^{I_k^r}d_{I_t}}{{}^{I_t}d_f} \simeq 0$ and employ the law of cosines in the triangle defined by the focal points of the two cameras and f:

$$
\begin{aligned}
&{}^{I_k^r}d_f^2 + {}^{I_t}d_f^2 - 2\,{}^{I_k^r}d_f\,{}^{I_t}d_f cos(\gamma) = {}^{I_k^r}d_{I_t}^2 \Rightarrow \\
&(\frac{{}^{I_k^r}d_f}{{}^{I_t}d_f})^2 + 1 - 2\frac{{}^{I_k^r}d_f}{{}^{I_t}d_f}cos(\gamma) = (\frac{{}^{I_k^r}d_{I_t}}{{}^{I_t}d_f})^2 \simeq 0 \Rightarrow (\frac{{}^{I_k^r}d_f}{{}^{I_t}d_f} - 1)^2 + 2\frac{{}^{I_k^r}d_f}{{}^{I_t}d_f}(1 - cos(\gamma)) \simeq 0
\end{aligned}
$$

which implies that ${}^{I_k^r}d_f \simeq {}^{I_t}d_f = d$. Dividing both sides of (8) with d and considering that ${}^{I_k^r}d_{I_t} \ll d$, yields

$$
{}^{I_k^r}\mathbf{b}_f \simeq {}^{I_k^r}_{I_t}\mathbf{R}\,{}^{I_t}\mathbf{b}_f \tag{10}
$$

Therefore, given two pairs of inliers, the rotation matrix ${}^{I_k^r}_{I_t}\mathbf{R}$ in (10) can be found in closed form (see Section 6.2).

6.2 2pt RANSAC minimal solver

Consider bearing measurements ${}^{I_1}\mathbf{b}_{f_1}$, ${}^{I_1}\mathbf{b}_{f_2}$, ${}^{I_2}\mathbf{b}_{f_1}$, ${}^{I_2}\mathbf{b}_{f_2}$ to two features from two images, and assume that the motion between them is purely rotational, thus

$$
{}^{I_2}\mathbf{b}_{f_i} = \mathbf{R}({}^{I_2}_{I_1}\bar{q})\,{}^{I_1}\mathbf{b}_{f_i},\ i = 1,2
$$

where ${}^{I_2}_{I_1}\bar{q}$ is the unit quaternion of rotation. Then, the closed-form solution is:

$$
{}^{I_2}_{I_1}\bar{q} = \gamma \begin{bmatrix} ({}^{I_2}\mathbf{b}_{f_1} - {}^{I_1}\mathbf{b}_{f_1}) \times ({}^{I_2}\mathbf{b}_{f_2} - {}^{I_1}\mathbf{b}_{f_2}) \\ ({}^{I_2}\mathbf{b}_{f_2} - {}^{I_1}\mathbf{b}_{f_2})^T ({}^{I_2}\mathbf{b}_{f_1} + {}^{I_1}\mathbf{b}_{f_1}) \end{bmatrix}
$$

where γ is the normalization factor that ensures unit length.

References

1. Autonomous flights through image-defined paths (videos). http://mars.cs.umn.edu/projects/isrr2015/quadrotor.html.
2. Bebop Drone. http://www.parrot.com/products/bebop-drone/.
3. Vicon Motion Systems Ltd. http://www.vicon.com/.
4. A. Alahi, R. Ortiz, and P. Vandergheynst. "FREAK: Fast retina keypoint". In *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 510–517, Providence, RI, Jun. 16–21 2012.
5. S. Azrad, F. Kendoul, and K. Nonami. "Visual servoing of quadrotor micro-air vehicle using color-based tracking algorithm". *Journal of System Design and Dynamics*, 4(2):255–268, Mar. 2010.
6. C. Bills, J. Chen, and A. Saxena. "Autonomous mav flight in indoor environments using single image perspective cues". In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 5776 – 5783, Shanghai, China, May 9–13 2011.

7. O. Bourquardez, R. Mahony, and F. C. N. Guenard. "Image-based visual servo control of the translation kinematics of a quadrotor aerial vehicle". *IEEE Transactions on Robotics*, 25(3):743–749, Jun. 2009.
8. F. Chaumette and S. Hutchinson. "Visual servo control, part i: basic approaches". *IEEE Robotics and Automation Magazine*, 13(4):82–90, Dec 2006.
9. F. Chaumette and S. Hutchinson. "Visual servo control, part ii: advanced approaches". *IEEE Robotics and Automation Magazine*, 14(1):109–118, Apr. 2007.
10. Z. Chen and R. T. Birchfield. "Qualitative vision-based path following". *IEEE Transactions on Robotics*, 25(3):749–754, Jun. 2009.
11. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, Cambridge, MA, 2001.
12. J. Courbon, Y. Mezouar, N. Guenard, and P. Martinet. "Vision-based navigation of unmanned aerial vehicle". *Control Engineering Practice*, 18(7):789–799, Jul. 2010.
13. J. Courbon, Y. Mezouar, and P. Martinet. "Indoor navigation of a non-holonomic mobile robot using a visual memory". *Autonomous Robots*, 25(3):253–266, Jul. 2008.
14. A. Diosi, S. Šegvić, A. Remazeilles, and F. Chaumette. "Experimental evaluation of autonomous driving based on visual memory and image-based visual servoing ". *IEEE Transactions on Robotics*, 12(3):870–883, Sept. 2011.
15. T. Do, L. C. Carrillo-Arce, and S. I. Roumeliotis. "Autonomous flights through image-defined paths", http://mars.cs.umn.edu/publications.html. Technical report, Mar. 2015.
16. M. Fischler and R. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". *Communications of the ACM*, 24(6):381–395, Jun. 1981.
17. G. F. Franklin, J. D. Powell, and M. L. Workman. *Digital control of dynamic systems*. Addison-Wesley, Reading, MA, 1997.
18. T. Goedemé, T. Tuytelaars, L. V. Gool, G. Vanacker, and M. Nuttin. "Feature based omnidirectional sparse visual path following". In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1806–1811, Alberta, Canada, Aug. 2–6 2005.
19. C. X. Guo, D. G. Kottas, R. C. DuToit, A. Ahmed, R. Li, and S. I. Roumeliotis. "Efficient visual-inertial navigation using a rolling-shutter camera with inaccurate timestamps". In *Proc. of the Robotics: Science and Systems Conference*, Berkeley, CA, Jul. 12–16 2014.
20. R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
21. D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys. "An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications". In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 1736–1741, Karlsruhe, Germany, May 6–16 2013.
22. B. Horn. "Closed-form solution of absolute orientation using unit quaternions". *Journal of the Optical Society of America A*, 4(4):629–642, Apr. 1987.
23. B. Horn. "Relative orientation". *International Journal of Computer Vision*, 4(1):59–78, Jan. 1990.
24. D. Lee, T. Ryan, and H. J. Kim. "Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing". In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 971–976, Saint Paul, MN, May 14–18 2012.
25. G. L. Mariottini and S. I. Roumeliotis. "Active vision-based robot localization and navigation in a visual memory". In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 6192–6198, Shanghai, China, May 9–13 2011.
26. T. Nguyen, G. K. I. Mann, and R. G. Gosine. "Vision-based qualitative path-following control of quadrotor aerial vehicle". In *Proc. of the IEEE International Conference on Unmanned Aircraft Systems*, pages 412–417, Orlando, FL, May. 27–30 2014.
27. D. Nistér. "An efficient solution to the five-point relative pose problem". *IEEE Transactions on Patter Analysis and Machine Intelligence*, 26(6):756–770, Jun. 2004.
28. D. Nistér and H. Stewénius. "Scalable recognition with a vocabulary tree". In *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, New York, NY, Jun. 17–22 2006.