High-Speed Autonomous Quadrotor Navigation through Image Paths

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

Abstract— **This paper addresses the problem of autonomous quadrotor navigation within indoor spaces. In particular, we focus on the case where a visual map of the area, represented as a graph of linked images, is constructed offline (from visual and inertial data collected beforehand) and used to determine visual paths for the quadrotor to follow. Moreover, during the actual navigation, the quadrotor employs both the 3+1pt and the 1+1pt RANSAC to efficiently determine its desired motion towards the next reference image, for both cases of sufficient and insufficient baseline (e.g., rotations in place). Lastly, we introduce an adaptive optical-flow algorithm that can accurately estimate the quadrotor's horizontal velocity under adverse conditions (e.g., when flying over dark, textureless floors) by progressively using information from more parts of the images. The speed and robustness of our algorithms are evaluated experimentally on a commercial quadrotor navigating in the presence of dynamic obstacles (i.e., people walking), along lengthy corridors, and through tight corners, as well as across building floors via poorly-lit staircases.**

I. INTRODUCTION AND RELATED WORK

In order for a quadrotor to autonomously navigate, it must be able to determine its current location and compute a path towards its destination. One way to solve this problem indoors is to create, typically offline, a *dense* 3D map and use it for both localization and path planning. The high computational cost and memory requirements of such an approach, however, limit its applicability to small-size areas. On the other hand, a building can be described as a *visual graph* using images, as well as inertial data, collected beforehand. Such a representation has many advantages, such as *scalability* (no metric global map needs to be constructed) and *ease of implementation* (the images can be collected by a person walking through the building with the quadrotor). Moreover, visual paths can be easily specified by a user by selecting the corresponding images along which the quadrotor needs to navigate, or by simply indicating the start and end images. The main challenge that such an approach poses, however, is designing algorithms for fast and reliably navigating the quadrotor along the visual path despite the lack of scale in the reference trajectory.

Controlling a robot to reach a specific destination defined in the image space can be achieved using visual servoing (VS) [7], [8]. 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 [7]. Prior work on VS for quadrotors equipped with a downward-pointing camera has addressed the problem of landing on a known target [21], [6] and hovering over an arbitrary target [4]. Furthermore, for quadrotors equipped with a forward-pointing camera, [5] 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 recently applied to aerial vehicles [24], [11]. In particular, in [24] an extension of the "funnel"-lane concept of [9] 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 move in order to match the reference image. Then, the desired motion of the quadrotor is computed as the convex combination of the heading and 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 [11], the VS method of [12] 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., [13] and [16]), [11] uses the PBVS algorithm described in [8] 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 [24] and [11] is that they cannot deal with rotations in place (often required for navigating through tight spaces), or, for the case of [11], 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 [24], or no turn in [11], where the quadrotor was moving back and forth between two locations connected via a direct path.

To address these limitations, in our previous work [14], we introduced a PBVS algorithm that uses both the 5pt[1] and the 2pt RANSAC to (i) distinguish between wide- and short-baseline motions, and (ii) efficiently switch between

[1]The 5pt RANSAC [25] 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. The 2pt RANSAC [14] 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.

reference images. Moreover, the resulting algorithm was able to reliably navigate the quadrotor over a wide range of motions comprising rotations in place under challenging conditions (e.g., featureless corridors and areas with numerous specular reflections). A key limitation of [14], however, was that the navigation algorithm could only run at 8 Hz on the quadrotor's processor. This was primarily due to the high processing requirements of the 5pt RANSAC (45 ms per image pair). Moreover, the optical-flow algorithm used in [14] was sensitive to lighting conditions and floor texture, which made navigating through not-well-lit regions, or over low-texture surfaces, challenging. To overcome these limitations, in this work we extend the approach of [14] in two ways:

1) We employ the 3+1pt [23], instead of the 5pt, RANSAC algorithm, which uses the gravity-direction correspondence between image pairs[2] to very efficiently estimate the 5 dof between them. Analogously, we replace the 2pt with the 1+1pt RANSAC.[3] As a result, we are able to determine the desired motion 80% faster than in [14].
2) We extend the optical-flow algorithm of [18] so as to gradually acquire and process additional information from a larger part of the image so as to compute a robust and accurate estimate of the quadrotor's horizontal velocity.

The key advantages of the proposed PBVS algorithm are as follows: (i) By employing the closed-form solution for the minimal solver of the 3+1pt, the quadrotor is able to process visual information very efficiently, and thus navigate 2.5 times faster as compared to [14].[4] (ii) Our algorithm is capable of dealing with a wide range of lighting conditions and environments, (e.g., dark, low-texture floors), while maintaining a fast operational speed. Lastly, and, in order to demonstrate the improved efficiency, accuracy, and robustness of the proposed algorithm as compared to [14], we have implemented and tested it under adverse lighting condition, using the Parrot Bebop [2] in areas comprising lengthy corridors, tight turns, and stairs.

II. QUADROTOR AND OBJECTIVE

The Parrot Bebop quadrotor (see Fig. 1) has an attitude-stabilization controller, which take as input information from an observer that processes gyroscope and accelerometer measurements, from the onboard inertial measurement unit (IMU), to estimate its roll and pitch angles, yaw-rate, and thrust. Additionally, it carries a downward-pointing camera to estimate optical flow, and an ultrasonic sensor to measure the distance to the ground.[5] Furthermore, the Bebop has access to (i) a forward-facing wide field of view (WFOV) camera (ii) an onboard ARM processors for executing, in real-time, all image-processing and control algorithms necessary by the proposed PBVS method.

Fig. 1. The Parrot Bebop quadrotor equipped with a 180 deg WFOV camera, an optical-flow sensor, and an ARM-based processor.

As mentioned earlier, the objective of this work is to develop a robust algorithm that will allow the quadrotor to follow, at relatively high speed, visual paths, defined as sequences of pre-recorded images.

III. TECHNICAL APPROACH

As in [14], 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 path. Additionally, and in order to increase robustness, our navigation approach employs a vocabulary tree (VT)-based [26] method for relocalizing inside the previously constructed visual graph when losing track of the reference image path.

A. Offline phase

1) Map generation: Similar to [14], a person carrying a quadrotor, walks through the area of interest collecting images at 15 Hz. Note though that in contrast to [14], we concurrently compute and save along each image the corresponding *gravitational direction*. Subsequently, we extract FREAK image points [3] and employ a VT to generate the visual map which is represented as a visual graph (VG) whose nodes correspond to the recorded images (see Fig. 2). An edge between two images signifies that these were matched by the VT and at least 30 point-correspondences passed the 3+1pt or 1+1pt RANSAC. Furthermore, we assign *weights* to these edges inversely proportional to the number

[2]The gravity-direction is computed from the IMU, which is transformed to camera frame of reference by the extrinsic IMU-camera parameters. Those parameters are estimated using the algorithm in [22].

[3]In this case, the minimal solver remains the same, but we improve robustness since the algorithm requires only one, instead of two, point feature correspondences along with the gravity direction.

[4]Note also that as compared to the 5pt, the 3+1pt RANSAC requires fewer (25 versus 17) points for estimating the 5 dof transformation between two imgaes, thus allowing operation in areas with only few features.

[5]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 both unreliable and computationally expensive.

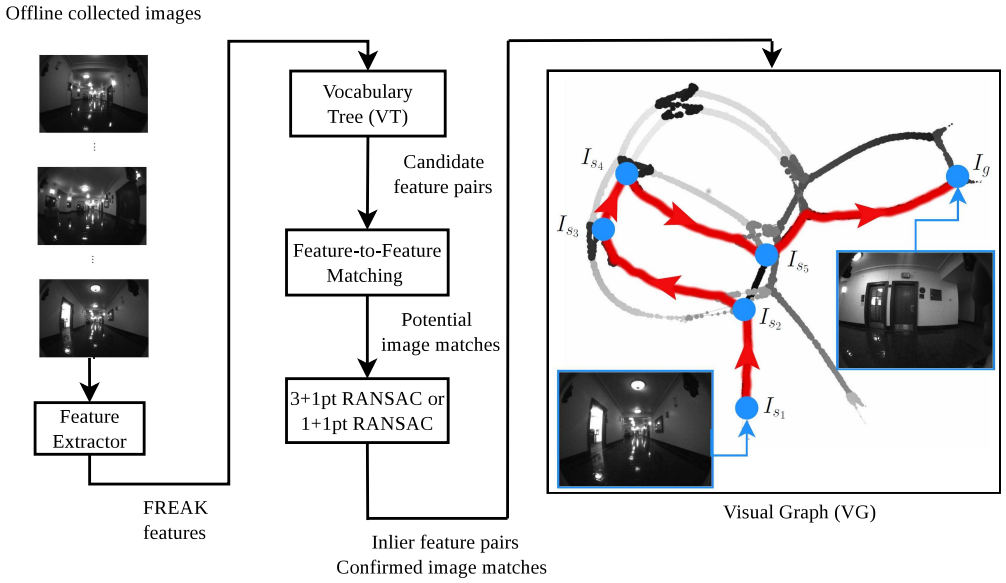

Fig. 2. Offline phase: The area of interest is described as a visual graph (VG) whose nodes correspond to images, while edges link images containing a sufficient number of common features for reliably visually-servoing between them. In the VG, I_{s1}, I_g denote the start and goal images, respectively,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

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.

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 or replacing subsets of images corresponding to new/altered regions of a building.

2) Path specification: The VG is used as in [14] 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 [10]. This process is easily extended to include intermediate locations (e.g., $I_{g_1}, I_{g_2} \ldots I_{g_n}$), by simply resetting as the start of the next path segment the end image of the previous one (e.g., $I_{s_{i+1}} = I_{g_i},\ i = 1 \ldots n$).

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 3+1pt and 1+1pt RANSAC algorithms until it finds the first image, I_{s+m}, $m \geq 1$, that either has more 3+1pt than 1+1pt 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 $\mathscr{P} = \{I_1^r, I_2^r, \ldots, I_n^r\}$ 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 and the direction of gravity. The former is useful in case the quadrotor gets lost (see Section III-B.4), while the latter is used by the quadrotor for efficiently finding and matching its next reference image through the process described hereafter.

B. Online phase

1) System state determination: Firstly, consider the case of sufficient baseline; we are interested in computing the desired motion that will bring the quadrotor close to the reference image $I_k^r \in \mathscr{P}$. To do so, we seek to estimate the quadrotor's 5 dof with respect to I_k^r by extracting and matching features between its current, I_t, and reference, I_k^r, images. Specifically, given three pairs of feature matches between I_t and I_k^r, we employ the 3+1pt minimal solver of [23] to compute the 5 dof transformation from I_t to I_k^r based on the relation between the gravitational directions of the two images:

$$ {}^{I_k^r}\mathbf{g} = {}^{I_k^r}_{I_t}\mathbf{R}\,{}^{I_t}\mathbf{g} \tag{1} $$

and the epipolar constraint for the 3 feature correspondences:

$$ {}^{I_k^r}\mathbf{b}_{f_j}^T \lfloor {}^{I_k^r}\mathbf{t}_{I_t} \times \rfloor {}^{I_k^r}_{I_t}\mathbf{R}\,{}^{I_t}\mathbf{b}_{f_j} = 0, \quad j = 1 \ldots 3 \tag{2} $$

Note, however, that the estimate from (1), (2) is not reliable when the baseline between I_t and I_k^r is short. Furthermore, the appearance-based feature matching between I_t and I_k^r (i.e., the 3+1pt RANSAC's input), is not always reliable (e.g., due to adverse lighting conditions and/or occlusions). 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:

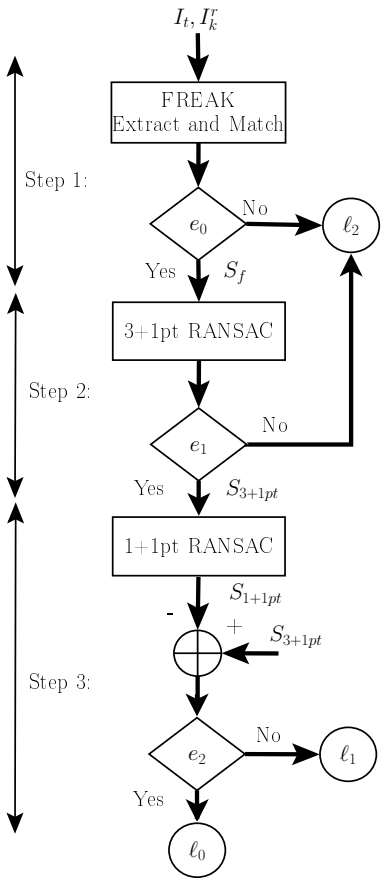

Fig. 3. Online: Schematic diagram of the steps and transitions between the different states of the automaton $\mathscr{H}$.

- $\mathscr{L}$ 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., failure in the appearance-based feature matching.
- $\mathbf{x}(t,k) = [I_t, I_k^r, \mathbf{r}(t,k)]$ 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 the reference visual path $\mathscr{P}$, the system must ideally iterate between two steps until the last element of $\mathscr{P}$ is reached: (i) When in ℓ_0, we compute the motion $\mathbf{r}$ and control the quadrotor so as to bring the system to state ℓ_1 (see Section III-B.2); (ii) When in 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 III-B.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 III-B.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 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 $e_0 : |S_f| \geq 17$, 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 3+1pt RANSAC to compute the geometric relation $({}^{I_k^r}_{I_t}\hat{\mathbf{R}}, {}^{I_k^r}\hat{\mathbf{t}}_{I_t})$ between I_t and I_k^r, as well as the set of features whose reprojection error [17] is within a threshold ε_1 (the error tolerance for outlier rejection [15]). At this point, we require that the condition $e_1 : |S_{3+1pt}| \geq 17$ (i.e., the number of 3+1pt RANSAC inliers) 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., ${}^{I_k^r}d_{I_t} \ll {}^{I_t}d_f, {}^{I_k^r}d_f \Leftrightarrow {}^{I_k^r}\mathbf{b}_f \simeq {}^{I_k^r}_{I_t}\mathbf{R}\,{}^{I_t}\mathbf{b}_f$), the 5 dof degenerates into a 3 dof, rotation-only constraint that is satisfied by all the 3+1pt inliers. Our algorithm uses this observation to determine if there is sufficient baseline between the current, I_t, and reference, I_k^r, images. In particular, we employ the 1+1pt RANSAC on the features $f \in S_{3+1pt}$ to compute the rotation ${}^{I_k^r}_{I_t}\breve{\mathbf{R}}$ between two images and determine $S_{1+1pt} = \{f \in S_{3+1pt} \mid 1 - {}^{I_k^r}\mathbf{b}_f^T\, {}^{I_k^r}_{I_t}\breve{\mathbf{R}}\, {}^{I_t}\mathbf{b}_f < \varepsilon_2\}$, which is the subset of 3+1pt inliers that are also 1+1pt inliers. Lastly, and in order to compensate for the noise in the measurements and the randomness of RANSAC, instead of requiring $|S_{1+1pt}| = |S_{3+1pt}|$, we employ the condition $e_2 : \frac{|S_{1+1pt}|}{|S_{3+1pt}|} > 0.94$ 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.

2) Wide baseline (ℓ_0):

a) 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 1+1pt RANSAC. To remove them, we define as $S'_{3+1pt} = S_{3+1pt} \setminus S_{1+1pt}$, run again the 3+1pt RANSAC on the features $f \in S'_{3+1pt}$, and use the winning hypothesis to initialize an iterative batch-least squares algorithm [20] to improve the accuracy of the estimated desired motion between I_t and I_k^r.

At this point, note that although the desired motion between I_t and I_k^r 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,[6] and the desired

[6]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.

(relative) yaw angle ${}^{I_k^r}\hat{\psi}_{I_t}$. 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 can measure) and provide our PID controller described in [14] with the following desired motion vector:

$$\mathbf{r} = \begin{bmatrix} v_x^d \\ v_y^d \\ \hat{\psi} \end{bmatrix} = \begin{bmatrix} t_x \ v_0 \\ t_y \ v_0 \\ {}^{I_k^r}\hat{\psi}_{I_t} \end{bmatrix} \tag{3}$$

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, ${}^{I_k^r}_{I_t}\check{\mathbf{R}}$, by employing the least-squares method of [19] on the features $f \in S_{1+1pt}$ using as initial estimate the one from the minimal solver of the 1+1pt RANSAC. After extracting the yaw component, if $|{}^{I_k^r}\check{\psi}_{I_t}| > \tau_3$,[7] we send the desired rotation-in-place motion $\mathbf{r}^T = [0 \ \ 0 \ \ {}^{I_k^r}\check{\psi}_{I_t}]^T$ 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 $\mathscr{P}$.

Alternatively, we can leverage the yaw angle (computed off-line - see Section III-A.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 of the current path segment has been reached, or, determine the remaining rotation between the current image and the last rotation-only reference image.

4) Lost mode (ℓ_2): There are four possible cases that can cause the quadrotor to get lost:

- I_t enters a featureless region.
- I_t enters a region where the result from the FREAK feature matching between I_t and I_k^r is unreliable.
- I_t significantly deviates from its current path, in order to avoid an obstacle.
- Dynamic obstacles (e.g., people) obstruct its 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 20 features in common with I_t that pass the 3+1pt or 1+1pt 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 the last reference image where it computed good matches) for 0.5 sec and then attempts again to retrieve a good reference image I^r_{best}. This iterative process is repeated for 10 times before declaring that the quadrotor is lost, in which case, it autonomously lands.

[7]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.

C. Optical Flow

In this section, we describe our extension of the PX4Flow algorithm [18]. The original algorithm firstb extracts a 64x64 patch in the center of the downward-pointing camera's image and computes the optical flow of each of the 25 8x8 pixel blocks within the patch based on the sum of absolute differences (SAD) with a search area of half the window size (i.e., 5 pixels in the x, y directions of the second image). Then, subpixel refinement is applied to obtain better matching results with half pixel accuracy. Finally, the algorithm constructs two histograms of pixel displacements in the x and y directions based on the flow from the 25 blocks and picks the one with the highest voting value as the optical flow for that frame.

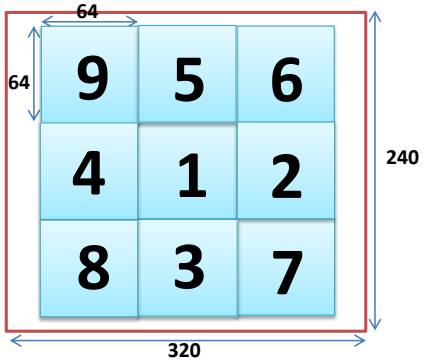

Fig. 4. The order of patch-selection. The largest (red) block corresponds to the whole image, while the 9 small blocks denote the 64x64 pixel patches. The assigned patches show the execution order of the adaptive optical-flow algorithm

In poor lighting or low-texture conditions, however, the patch in the center of the image may have less texture, or the minimum SAD for the chosen pixel may be very large, leading to erroneous optical flow estimation. To address this problem, we propose an extension of the PX4Flow algorithm which can estimate optical flow with displacement up to 5 pixels when considering translational motions.[8] In particular, our algorithm splits the image of size 320x240 into 9 patches of pixel size 64x64 (see Fig. 4). We begin by computing the histograms of optical-flow based on the PX4Flow algorithm for the center patch of the image (i.e., patch 1 in Fig. 4) and then, if the number of valid pixels,[9] among the total of 25 chosen pixel blocks, is less than or equal to 20, we continue to compute the optical flow, and accumulate histograms, in the patch order shown in Fig. 4. This process continues until the number of valid pixels in the histogram is larger than 20. The reason behind this order of patch-selection is that the pixels far away from the center have more radial distortion. Furthermore, as evident from (4), they are affected

[8]In practice, we can deal with small rotations but the main motion between consecutive frames needs to be translational.

[9]For a chosen pixel $\mathbf{p}$, if the sum of horizontal and vertical gradients of the 4x4 patch centered at $\mathbf{p}$ (i.e., textures of the image) is larger than a threshold τ_1 and the minimum SAD value in the search area is less than a threshold τ_2, $\mathbf{p}$ is considered to be a valid pixel.

by rotations more than the ones closest to the center:

$$\dot{x} = -\frac{v_x}{Z} + \frac{xv_z}{Z} + xy\omega_x - (1+x^2)\omega_y + y\omega_z$$
$$\dot{y} = -\frac{v_y}{Z} + \frac{yv_z}{Z} + (1+y^2)\omega_x - xy\omega_y - x\omega_z \quad (4)$$

where x and y are homogeneous coordinates of a pixel $\mathbf{p}$, $\dot{x}$ and $\dot{y}$ are the optical-flow velocities of $\mathbf{p}$, v_x, v_y and v_z are the linear velocities of the downward-pointing camera, ω_x, ω_y and ω_z are the angular velocities of the downward-pointing camera, and Z is the Z coordinate of the 3-D point to which $\mathbf{p}$ corresponds.

IV. Experimental Results

In this section, we present experimental results for validating both the extension of PX4FLow algorithm and the ability of the Bebop to fly through image-defined paths. In the optical-flow experiment, we show the difference in performance between our proposed and the original approach using data taken in a dark stair case inside Walter library. Next, we performed experiments with the Parrot Bebop quadrotor in two scenarios: (i) a 75 m indoor area without stairs to evaluate the algorithm's performance under maximum allowable speed (2.0 m/s). (ii) a 150 m indoor area with the available of stairs under more conservative maximum speed (1.2 m/s).

A. System setup

The Bebop 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.

B. Optical-flow experiment

We implemented the proposed adaptive optical-flow algorithm using ARM NEON and achieved an average time of 0.948 ms for images of size of 320x240. Thus, for a frame rate of 60Hz, the total time per second for computing the optical-flow is approximately 57 ms which is sufficient for real-time operation.

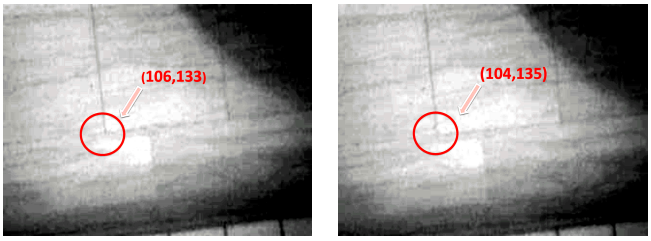

Fig. 5. Representative consecutive image pairs from the Walter library stairs. The red circle shows the same corner in the two images with their pixel coordinates.

Our test dataset of 1000 images was taken from the Bebop at the Walter library stairs. The approximate motion is shown as a black dashed line in Fig. 7. We then used both the PX4Flow and our algorithm, to estimate the quadrotor's horizontal velocity between consecutive image pairs. The path resulting from integrating those velocities over time is shown in Fig. 7. As evident, our algorithm is able to give significantly more accurate results compared to the PX4Flow. In order to better understand where the gain in performance comes from, we further show the histograms of the pixel blocks' displacement (see Fig. 6) resulting from the image pair shown in Fig. 5. The blue bars in Fig. 6, which depict histograms from the PX4Flow algorithm, do not have a distinct flow peak, suggesting inaccurate optical flow estimation. In contrast, the histograms from our proposed algorithm, shown as yellow bars, have a clear peak, resulting from the additional optical-flow information collected from the extra image blocks used.

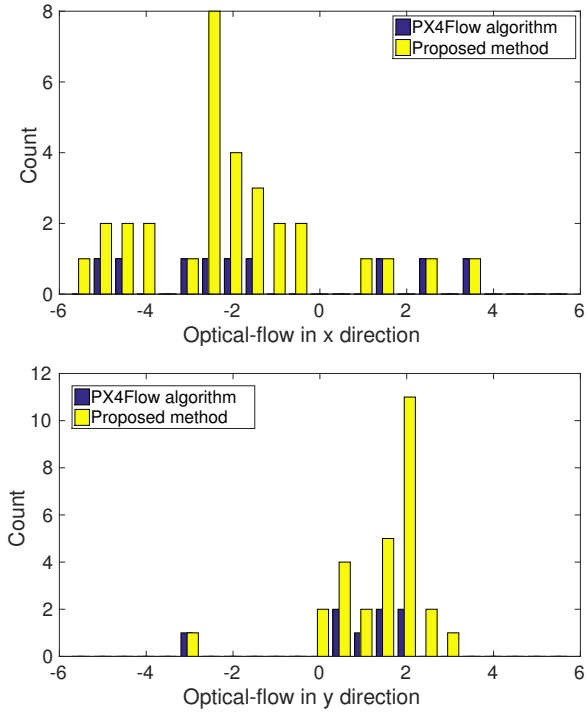

Fig. 6. Histograms of the optical-flow in the x and y directions. The blue bars depict the histograms from the PX4Flow algorithm. The yellow bars indicate the histograms from the proposed optical-flow algorithm.

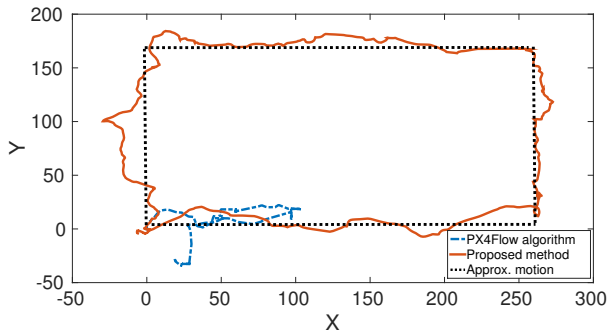

Fig. 7. The Bebop's position estimated from the PX4Flow algorithm and our adaptive optical-flow algorithm.

C. Experiment 1 (within the same floor)

This experiment took place in the Walter Library's basement which is very similar to the long experiment in [14], 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. Fig. 8 shows the blueprint of the experimental area depicting the reference visual path (red bold line), and snapshots of the quadrotor in flight. During the experiment, the Bebop was able to complete the reference trajectory in 97 sec, at an average speed of 1.0 m/s.

Fig. 8. Experiment 1: Blueprint of the experimental area, reference (1-6-1), and snapshots of the Bebop along its paths (1-6).

Table 1 illustrates the difference in performance between the combination of the 5pt & 2pt in our previous work [14] and the 3+1pt & 1+1pt in this paper, while Table 2 shows the execution time on the same Bebop's processor.

Exp.	Length	Total time	Trans. time	Avg. speed
5pt	75 m	240 sec	180 sec	0.4 m/s
3+1pt	75 m	97 sec	73 sec	1.0 m/s

Table 1: Comparison in performance between [14] and the proposed method.

Impl.	Solver	RANSAC	Freq.
5pt	1.3 ms	45 ms	8 Hz
3+1pt	0.13 ms	5 ms	15 Hz

Table 2: Comparison in execution time on the same processor between [14] and the proposed method.

As evident, by employing the proposed combination of the 3+1pt and 1+1pt (instead of the 5pt and 2pt) RANSAC, the quadrotor is able to follow the desired path 2.5 times faster than [14].

D. Experiment 2 (stairs connecting two floors)

This experiment took place in the Walter Library's basement and 1st floor which include stairs connecting them. In addition to the longer trajectory in translation (150 m), the quadrotor also has to deal with the challenging staircase area, where the rate of change in the ultrasonic sensor measurement is high. The main difficulty in this experiment was accurately estimating the optical-flow. Note that the stairs comprise textureless steps that pose a significant challenge to the optical-flow algorithm. Furthermore, part of the stairs (4-7 and 11-14, see Fig. 9) is featureless and quite dark compared to the rest of the path. Despite the adverse conditions, the quadrotor was able to successfully navigate through this path in 250 sec. Fig. 9 shows the blueprint of the experimental area with the reference visual path (red bold line), and snapshots of the Bebop in flight. The videos of both experiments are available at [1].

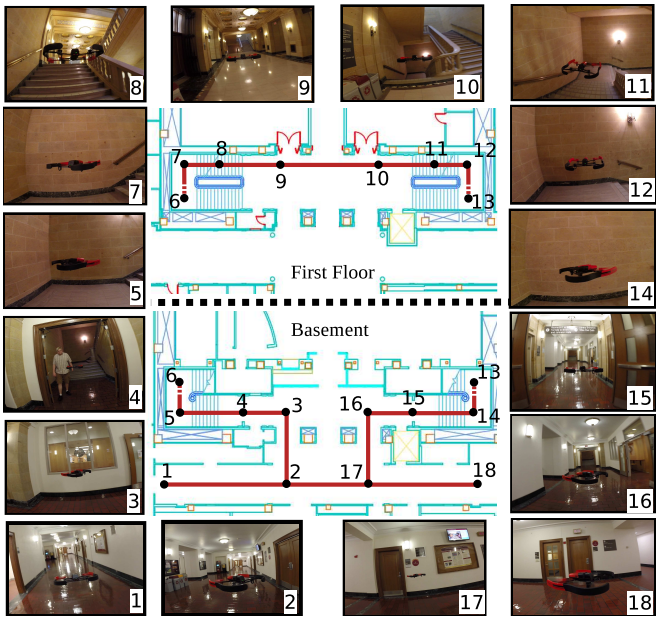

Fig. 9. Experiment 2: Blueprint of the experimental area, the reference path, and the snapshot of the Bebop during flights. Note that the stairs' parts are (3-9) and (10-15).

V. Conclusion and Future Work

In this paper, we presented a visual-servoing algorithm that allows quadrotors to autonomously navigate within a previously-mapped area. Similar to the previous work [14], the map is constructed offline from images collected by

a user walking though the area of interest and carrying a quadrotor. Specifically, 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, intermediate 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; it then 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.

In the online phase, instead of employing a mixture of 2pt and 5pt RANSAC as in [14], we use the combination of 1+1pt and 3+1pt RANSAC for determining the type of motion required (rotational, translational with close-by or faraway scene), and for switching to the next reference image. Additionally, we also extend the optical-flow algorithm used in [14] to adaptively process more information in order to improve the accuracy of horizontal velocity estimate. By exploiting the gravitational direction correspondence between an image pair, the entire algorithm can be executed at higher rate, resulting in 2.5 times faster in navigation speed. Finally, the improved optical-flow algorithm aids the quadrotor the ability to fly through adverse downward environments, hence, navigation task is achieved through an indoor path comprising dark, textureless floor and staircases.

VI. Appendix

A. 1+1pt RANSAC minimal solver

Consider bearing measurements ${}^{I_1}\mathbf{b}_f$, ${}^{I_2}\mathbf{b}_f$ to a feature correspondence, the unit gravitational vectors ${}^{I_1}\hat{\mathbf{g}}$, ${}^{I_2}\hat{\mathbf{g}}$ from two images, and assume that the motion between them is purely rotational, thus

$$\begin{aligned} {}^{I_2}\mathbf{b}_f &= \mathbf{R}({}^{I_2}_{I_1}\bar{q})^{I_1}\mathbf{b}_f \\ {}^{I_2}\hat{\mathbf{g}} &= \mathbf{R}({}^{I_2}_{I_1}\bar{q})^{I_1}\hat{\mathbf{g}} \end{aligned} \tag{5}$$

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

$${}^{I_2}_{I_1}\bar{q} = \gamma \begin{bmatrix} ({}^{I_2}\mathbf{b}_f - {}^{I_1}\mathbf{b}_f) \times ({}^{I_2}\hat{\mathbf{g}} - {}^{I_q}\hat{\mathbf{g}}) \\ ({}^{I_2}\hat{\mathbf{g}} - {}^{I_1}\hat{\mathbf{g}})^T ({}^{I_2}\mathbf{b}_f + {}^{I_1}\mathbf{b}_f) \end{bmatrix}$$

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/icra2016/quadrotor.html.

[2] Bebop Drone. http://www.parrot.com/products/bebop-drone/.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] F. Chaumette and S. Hutchinson. "Visual servo control, part i: basic approaches". *IEEE Robotics and Automation Magazine*, 13(4):82–90, Dec 2006.

[8] F. Chaumette and S. Hutchinson. "Visual servo control, part ii: advanced approaches". *IEEE Robotics and Automation Magazine*, 14(1):109–118, Apr. 2007.

[9] Z. Chen and R. T. Birchfield. "Qualitative vision-based path following". *IEEE Transactions on Robotics*, 25(3):749–754, Jun. 2009.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, Cambridge, MA, 2001.

[11] 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.

[12] 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.

[13] 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.

[14] T. Do, L. C. Carrillo-Arce, and S. I. Roumeliotis. "Autonomous flights through image-defined paths". In *Proc. of the International Symposium on Robotics Research (ISRR)*, Sestri Levante, Italy, Sept. 2015.

[15] 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.

[16] 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.

[17] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[18] 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.

[19] 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.

[20] B. Horn. "Relative orientation". *International Journal of Computer Vision*, 4(1):59–78, Jan. 1990.

[21] 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.

[22] F. M. Mirzaei and S. I. Roumeliotis. "A Kalman filter-based algorithm for imu-camera calibration: Observability analysis and performance evaluation". *IEEE Transactions on Robotics*, 24(5):1143–1156, Oct. 2008.

[23] O. Naroditsky, X. S. Zhou, J. Gallier, S. I. Roumeliotis, and K. Daniilidis. "Two efficient solutions for visual odometry using directional correspondence". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):818–824, Apr. 2012.

[24] 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.

[25] D. Nistér. "An efficient solution to the five-point relative pose problem". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, Jun. 2004.

[26] 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.