High-speed Autonomous Quadrotor Navigation through Visual and Inertial Paths

The International Journal of Robotics Research
XX(X):1–??

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Tien Do, Luis C. Carrillo-Arce, and Stergios I. Roumeliotis[1]

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 potentially inertial data collected beforehand) and used to determine visual paths for the quadrotor to follow. In addition, during the actual navigation, the quadrotor employs both *wide-* and *short-baseline* RANSACs to efficiently determine its desired motion towards the next reference image and handle special motions such as rotations in place. In particular, when the quadrotor relies only on visual observations, it uses the 5pt and 2pt algorithms in the wide- and short-baseline RANSACs, respectively. On the other hand, when information about the gravity direction is available, significant gains in speed are realized by using the 3pt+1 and 1pt+1 algorithms instead. 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-off-the-shelf 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.

Keywords

Quadrotor, autonomous navigation, visual servoing.

1 Introduction and Related Work

In order for a quadrotor to autonomously navigate within a GPS-denied area, it must be able to determine its current location using its onboard sensors 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, and potentially 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 efficiently 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) (Chaumette and Hutchinson 2006, 2007). 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 (Chaumette and Hutchinson 2006). Prior work on VS for quadrotors equipped with a downward-pointing camera has addressed the problem of landing on a known target (Lee et al. 2012; Bourquardez et al. 2009) and hovering over an arbitrary target (Azrad et al. 2010). Furthermore, for quadrotors equipped with a forward-facing camera, (Bills et al. 2011) classifies the environment into corridors, stairs, or "other" in order to determine the appropriate motion (turn, side-ways, or upward) necessary for the robot to continue its exploration.

In the context of navigating along a visual path, VS techniques have been recently applied to aerial vehicles (Nguyen et al. 2014; Courbon et al. 2010). In particular, in Nguyen et al. (2014) an extension of the "funnel"-lane concept of Chen and Birchfield (2009) 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

[1]MARS Lab, Dept. of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, USA

Corresponding author:
Tien Do, Dept. of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, 55455
Email: doxxx104@umn.edu

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 features' pixel coordinates between the reference and the current image.

In Courbon et al. (2010), the VS method of Courbon et al. (2008) 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., Diosi et al. (2011) and Goedemé et al. (2005)], Courbon et al. (2010) uses the PBVS algorithm described in Chaumette and Hutchinson (2007) 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 Nguyen et al. (2014) and Courbon et al. (2010) is that they cannot deal with rotations in place (often required for navigating through tight spaces), or, for the case of Courbon et al. (2010), 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 Nguyen et al. (2014), or no turns in Courbon et al. (2010), where the quadrotor was moving back and forth between two locations connected via a direct path.

To address these limitations, we present a PBVS-based quadrotor navigation algorithm that employs both a wide and and a short-baseline (WB and SB respectively) RANSAC to (i) distinguish between translational and rotational motions, and (ii) efficiently switch between reference images. In particular, the wide-baseline RANSAC estimates the relative orientation ${}^{I_c}_{I_d}\mathbf{R}$ and the unit vector of translation ${}^{I_c}\mathbf{t}_{I_d}$ between two frames $\{I_c\}$ and $\{I_d\}$ of the current and desired images, respectively. The short-baseline RANSAC estimates the relative orientation ${}^{I_c}_{I_d}\mathbf{R}$ between two frames, $\{I_c\}$ and $\{I_d\}$, under the assumption of very small baseline as compared to the depth of the scene. Once an initial relative pose estimate, from either the WB or the SB RANSAC, is determined, a least-squares iterative process is employed for refining it. Lastly, the desired 2.5D motion is extracted from the 5 dof relative pose and provided to the quadrotor's controller. As a result of this process, the quadrotor is able to reliably navigate over a wide range of motions comprising rotations in place under challenging conditions (e.g., lengthy featureless corridors, areas with numerous specular reflections). Additionally, the proposed method does not require recording the images by manually controlling the robot through the reference paths as is done in Nguyen et al. (2014); Courbon et al. (2010). Instead, one can easily define the desired paths by simply walking through the area of interest carrying the quadrotor. Lastly, we extend the optical-flow algorithm of Honegger et al. (2013) to reduce its sensitivity to lighting conditions and floor texture, and allow navigating through not-well-lit regions and over low-texture surfaces. The key contributions of this work are as follows:

1. We employ a WB [5pt (Nistér 2004)] and SB (2pt) RANSAC-based algorithm for (i) determining the type of motion (translational and rotational versus rotational only) that the quadrotor needs to undergo in order to reach the next location along its path, and (ii) switching to the next reference image.
2. When information about the gravity direction is available (e.g., from the IMU), we employ the 3pt+1 (Naroditsky et al. 2012) and 1pt+1 algorithm for WB and SB RANSAC, respectively, significantly improving the efficiency* of the proposed autonomous quadrotor navigation algorithm.
3. We extend the optical-flow algorithm of Honegger et al. (2013) 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.
4. We demonstrate the efficiency, accuracy, and robustness of the proposed algorithm under adverse lighting conditions onboard the Parrot Bebop quadrotor (Parrot-Inc) navigating through areas comprising lengthy corridors, tight turns, and stairs.

A preliminary version of this paper was presented in Do et al. (2015), where we introduced the concept of using both WB and SB RANSACs and demonstrated the robustness of this approach when navigating through tight spaces. A limitation of Do et al. (2015), however, was that due to the high processing requirements of the 5pt RANSAC (45 ms per image pair) the navigation algorithm could only run at 8 Hz on the quadrotor's processor. To address this issue in this work, we employ the gravity direction in the 3pt+1 RANSAC to increase the navigation algorithm's speed to 15 Hz. Furthermore, we improve the optical-flow algorithm previously used allowing the quadrotor to fly 2.5 times faster under a wide range of lighting conditions.

The rest of this paper is structured as follows: In Sect. 2, we describe the offline and online phase of our approach, as well as our extension to the PX4Flow optical-flow algorithm. In Sect. 3, we validate our method experimentally under challenging conditions. Finally, we provide our concluding remarks and outline our future research directions in Sect. 4.

2 Technical Approach

Our approach comprises two phases. In the first (offline) phase, a visual-graph (VG)-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

*In this case, the minimal solver remains the same, but we improve robustness to outliers since the algorithm requires only one, instead of two, point feature correspondences along with the gravity direction. Note that as compared to the 5pt, the 3pt+1 RANSAC requires fewer (17 versus 30) inliers for estimating the 5 dof transformation between two images, thus allowing operation in areas with only a few features.

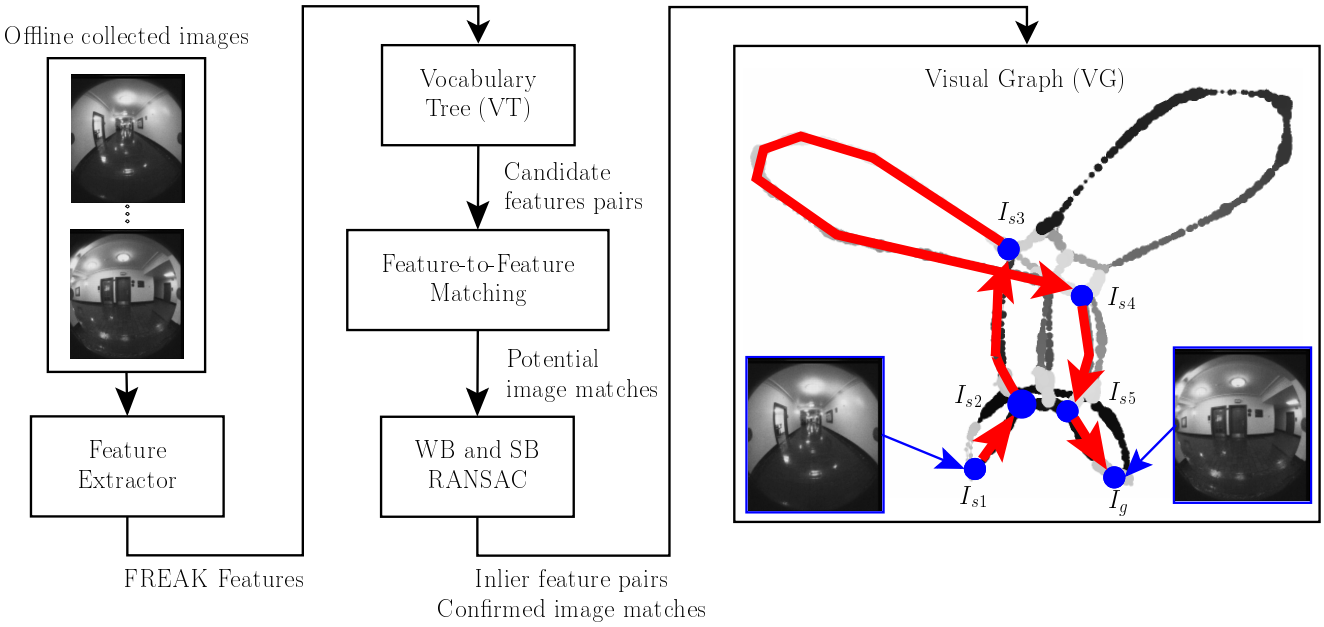

Figure 2. Offline phase: The area of interest is described as a visual graph (VG) whose nodes correspond to images, while edges link images containing a sufficient number of common features for reliably visually-servoing between them. In the VG, I_{s_1} and I_g denote the start and goal images, respectively, while $I_{s_2}, \ldots, I_{s_5}$ signify intermediate goal locations along the quadrotor's path specified by the user. The grayscale color of the VG codes the different map segments, while its thickness reflects the number of neighboring images each vertex image has.

Figure 1. The Parrot Bebop quadrotor equipped with a 180 deg wide field of view (WFOV) camera, an optical-flow sensor, and an ARM-based processor.

images captured by its onboard, forward-facing 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 (Nistér and Stewénius 2006) method for *relocalizing* inside the previously constructed VG when losing track of the reference image path.

Before presenting the details of our technical approach, we should note that the Parrot Bebop quadrotor used in this work (see Fig. 1) has an attitude observer/controller for stabilization, a forward-facing camera for visual navigation, a downward-pointing camera for estimating optical flow, and an ultrasonic sensor for measuring its distance to the ground. In particular, the observer processes gyroscope and accelerometer measurements, from the onboard inertial measurement unit (IMU), to estimate its roll and pitch angles, yaw-rate, and thrust, while the controller uses this information to regulate the corresponding commanded setpoints. Lastly, we note that despite the availability of metric information from the horizontal 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.

2.1 *Offline phase*

2.1.1 Map generation A person carrying the quadrotor, walks through the area of interest collecting images at 15 Hz. In addition, when available, we compute and save along each image the corresponding *gravitational direction* allowing us to run two different versions of RANSAC for WB (5pt or 3pt+1) and SB (2pt or 1pt+1). Subsequently, we extract FREAK feature points (Alahi et al. 2012) from each image and employ a VT to generate the visual map; the latter is represented as a 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 (or 17 when the gravity direction is available) passed the WB or SB 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 should be able to reliably navigate through in the image space towards its destination.

At this point, we should note that 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.

2.1.2 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 Cormen et al. (2001). 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., $I_{s_{i+1}} = I_{g_i}$, $i = 1, \ldots, n$). Note that, for the rest of the paper and in order to improve readability, we abuse the notation and have I_ℓ to represent both the ℓ^{th} image and the camera frame $\{I_\ell\}$ from which the image was taken. For the reader's convenience, we describe our nomenclature in Sect. B.

Once the visual path $\mathcal{P} = \{I_{s_1} = I_\xi, I_{\xi+1}, \ldots, I_g = I_{\xi+N}\}$ 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 $I_1^r = I_\xi$ and moves along the path matching FREAK features, using both the WB and SB RANSAC algorithms, until it finds the first image, $I_{\xi+m}$, $m \geq 1$, that either has more WB than SB RANSAC inliers, or the relative yaw angle between I_1^r and $I_{\xi+m}$ is greater than 10 deg. In the first case, we declare that the quadrotor is in translation, otherwise, in rotation and set $I_{\xi+m}$, as the next reference image I_2^r. The resulting path $\mathcal{P}_{pruned} = \{I_1^r, I_2^r, \ldots, I_n^r\}$ is provided to the quadrotor along with two additional pieces of information: (i) We specify which images correspond to rotation-only motion and compute the yaw angle between consecutive rotation-only images; (ii) We compute 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 Sect. 2.2.4), while the latter is used by the quadrotor for efficiently finding and matching its next reference image through the process described hereafter.

2.2 Online phase

2.2.1 System state determination Firstly, consider the case of WB; we are interested in computing the desired motion that will bring the quadrotor close to the reference image $I_k^r \in \mathcal{P}$. To do so, we seek to estimate the quadrotor's 5 dof transformation with respect to I_k^r, when only visual information is available, by extracting and matching features between its current, I_t, and reference, I_k^r, images. Specifically, given 5 pairs of feature matches between I_t and I_k^r, we employ the WB 5pt RANSAC (Nistér 2004) to compute the 5 dof transformation from I_t to I_k^r based on the epipolar constraint for the set of feature correspondences ($j = 1, \ldots, 5$):

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

where ${}^{I_k^r}\mathbf{b}_{f_j}$, ${}^{I_t}\mathbf{b}_{f_j}$ are the (unit) bearing vectors to feature f_j from frames $\{I_k^r\}$ and $\{I_t\}$, respectively; ${}^{I_k^r}\mathbf{t}_{I_t}$ is the unit translational vector of $\{I_t\}$ in $\{I_k^r\}$; and ${}^{I_k^r}_{I_t}\mathbf{R}$ is the rotational matrix describing the relative orientation between $\{I_t\}$ and $\{I_k^r\}$.

If the gravity direction is known for both the current, ${}^{I_t}\mathbf{g}$, and the reference image, ${}^{I_k^r}\mathbf{g}$, then we employ the WB 3pt+1 RANSAC (Naroditsky et al. 2012) to compute the 5 dof motion from I_t to I_k^r based on the relation between the gravitational directions of the two images

$$ {}^{I_k^r}\mathbf{g} = {}^{I_k^r}_{I_t}\mathbf{R}\, {}^{I_t}\mathbf{g} \tag{2} $$

and only 3 pairs of feature matches satisfying (1). Note that since the minimal solver of Naroditsky et al. (2012) employs a 4th-order polynomial equation whose solution is known in closed-form, it is significantly faster than that of Nistér (2004) which is based on the analytical solution of a 10th-order polynomial equation. Furthermore, by requiring 3, instead of 5, feature pair matches, the 3pt+1-RANSAC requires a significantly lower number of hypotheses as compared to the 5pt RANSAC.

At this point, we should note that the motion estimate from (1), and potentially (2), is not reliable when the baseline between I_t and I_k^r is short. In particular, when the distance between 2 images is significantly smaller than the distance to the feature from either image (i.e., ${}^{I_k^r}d_{I_t} \ll {}^{I_t}d_{f_i}, {}^{I_k^r}d_{f_i}$) the 5 dof degenerates into a 3 dof rotation-only transformation [see Do et al. (2015) for more details] between I_t and I_k^r. This 3 dof transformation can be computed (see Appendix A.1) by either (2pt RANSAC) employing two pairs of feature correspondences satisfying

$$ {}^{I_k^r}\mathbf{b}_{f_i} = {}^{I_k^r}_{I_t}\mathbf{R}\, {}^{I_t}\mathbf{b}_{f_i} \tag{3} $$

or (1pt+1 RANSAC) only a single pair of such feature matches and the direction of gravity [see (2)].

Another issue of concern is that the appearance-based feature matching between I_t and I_k^r (i.e., WB RANSAC's input) is not always reliable (e.g., due to adverse lighting conditions and/or image blur). To address these challenges, we model our system as a hybrid automaton $\mathcal{H}$ as follows:

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

- $\mathcal{L}$ 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.
- $\mathcal{E}$ is the set of relations governing transitions between the states in $\mathcal{L} = \{\ell_0, \ell_1, \ell_2\}$.

Given $\mathcal{H}$, and in order to complete the reference visual path $\mathcal{P}$, the system must ideally iterate between two steps until the last element of $\mathcal{P}$ 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 Sect. 2.2.2); (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 $\mathcal{P}$ (see Sect. 2.2.3), and the system, by design, returns to state ℓ_0. In the event of an external disturbance, however, the system may reach state ℓ_2. In this case, a recovery procedure is executed to attempt to bring the system back to ℓ_0 or ℓ_1 (see Sect. 2.2.4).

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 $\mathcal{E} = \{e_0, e_1, e_2\}$ in the following 3 steps.

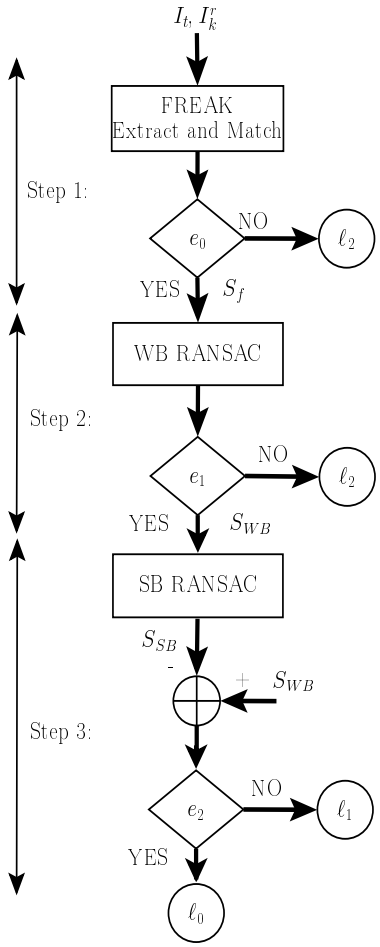

Figure 3. Online phase: Schematic diagram of the steps and transitions between the different states of the automaton $\mathcal{H}$.

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 $e_0 : |S_f| \geq \xi$ (where $\xi = 30$ or 17 for the 5pt and 3pt+1 RANSACs, respectively, and $|S_f|$ 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 WB (5pt or 3pt+1) RANSAC to compute the geometric relation $({}^{I_k^r}_{I_t}\mathbf{R}, {}^{I_k^r}\mathbf{t}_{I_t})$ between I_t and I_k^r, as well as the set of features whose reprojection error (Hartley and Zisserman 2004) is within a threshold ϵ_1 [the error tolerance for outlier rejection (Fischler and Bolles 1981)]. At this point, we require that the condition $e_1 : |S_{WB}| \geq \xi$, where $\xi = 30$ or 17 for the 5pt and 3pt+1 RANSACs, respectively, (i.e., the number of WB 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, as previously mentioned, when the baseline is short, the 5 dof degenerates into a 3 dof, rotation-only constraint that is satisfied by all the WB 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 SB (2pt or 1pt+1) RANSAC on the features $f \in S_{WB}$ to compute the rotation ${}^{I_k^r}_{I_t}\mathbf{R}$ between the two images and determine $S_{SB} = \{f \in S_{WB} \mid 1 - {}^{I_k^r}\mathbf{b}_f^T \, {}^{I_k^r}_{I_t}\mathbf{R} \, {}^{I_t}\mathbf{b}_f < \epsilon_2\}$, which is the subset of WB inliers that are also SB inliers. Lastly, and in order to compensate for the noise in the measurements and the randomness of RANSAC, instead of requiring $|S_{SB}| = |S_{WB}|$, we employ the condition $e_2 : \frac{|S_{SB}|}{|S_{WB}|} > 0.94$ 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.

2.2.2 *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's accuracy. Note that such features satisfy the SB RANSAC. To remove them, we define as ${S'}_{WB} = S_{WB} \setminus S_{SB}$, run again the WB RANSAC on the features $f \in {S'}_{WB}$, and use the winning hypothesis to initialize an efficient iterative least-squares algorithm (see Appendix A.2) to improve the accuracy of the estimated 5 dof motion between I_t and I_k^r.

Extracting the desired 2.5D motion At this point, we note that although the estimated relative pose 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 in place), 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,[†] and the desired (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 algorithm can measure) and obtain the desired motion vector:

$$\mathbf{r} = \begin{bmatrix} v_x^d \\ v_y^d \\ \psi^d \end{bmatrix} = \begin{bmatrix} t_x \, v_0 \\ t_y \, v_0 \\ {}^{I_k^r}\hat{\psi}_{I_t} \end{bmatrix} \tag{4}$$

Note also that when information (e.g., from ultrasound sensors) about closeby obstacles is available, we can modify $\mathbf{r}$ so that the quadrotor can smoothly avoid obtacles while maintaining its course [see Do et al. (2015) for more details]. After finalizing the desired motion $\mathbf{r}$, we provide it to the PID controller to compute the control actions.

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,

[†]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.

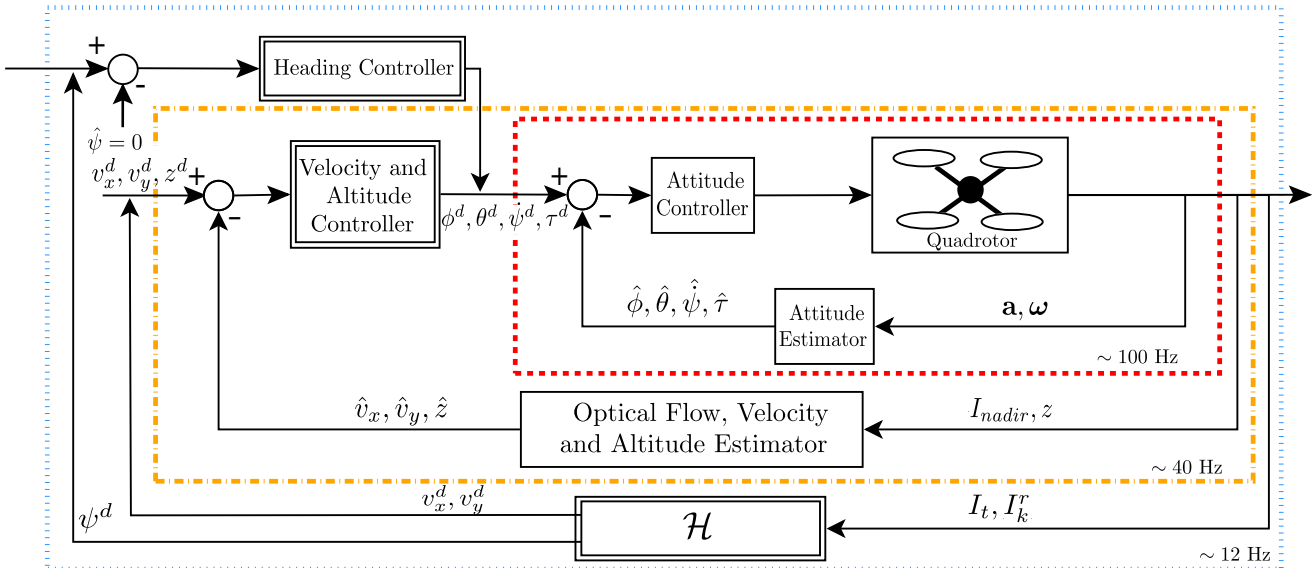

Figure 4. System block diagram. The double-line blocks denote components of our algorithm described in Sections 2.2.1 ($\mathcal{H}$) and 2.2.2 (Controllers).

linearized about the equilibrium (near-hover condition):

$$\begin{bmatrix} \dot{v}_x(t) \\ \dot{v}_y(t) \end{bmatrix} = g \begin{bmatrix} \theta(t) \\ -\phi(t) \end{bmatrix} \tag{5}$$

$$\ddot{z}(t) = \frac{1}{m}\tau(t) - g \tag{6}$$

where g is the magnitude of gravity, m is the quadrotor's mass, z is the quadrotor's altitude in the inertial frame, 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:

$$\begin{bmatrix} e_{v_x}(t) \\ e_{v_y}(t) \end{bmatrix} = \begin{bmatrix} v_x^d(t) - \hat{v}_x(t) \\ v_y^d(t) - \hat{v}_y(t) \end{bmatrix} \tag{7}$$

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:

$$e_z(t) = z^d(t) - \hat{z}(t) \tag{8}$$

Lastly, based on (6), (7), (8) and $\hat{\psi}$ in (4), we form a PID controller that computes the desired control input to the system as:

$$\mathbf{u}_k(t) = \begin{bmatrix} \theta^d(t) \\ \phi^d(t) \\ \tau^d(t) \\ \dot{\psi}^d(t) \end{bmatrix} = \begin{bmatrix} k_{p,v_x} e_{v_x}(t) + k_{i,v_x} \int e_{v_x}(t)dt \\ -k_{p,v_y} e_{v_y}(t) - k_{i,v_y} \int e_{v_y}(t)dt \\ k_{p,z} e_z(t) + k_{i,z} \int e_z(t)dt + k_{d,z} \dot{e}_z(t) \\ k_{p,\psi} \psi^d \end{bmatrix} \tag{9}$$

The gains k_p, k_i, and k_d that ensure high response, zero tracking error, and robustness were found as described in Franklin et al. (1997). Note also that, in order to keep the quadrotor in the near-hover condition, in practice, we bound the magnitude of the desired controller inputs for roll and pitch to be within 15 degree.

Fig. 4 depicts the 3-control-loop implementation of our navigation algorithm on the Parrot Bebop quadrotor. The outer loop takes as input I_t, I_k^r and determines the desired 2D velocity, v_x^d, v_y^d, and the yaw angle ψ^d at 12 Hz (see Sect. 2.2.2). The velocity and altitude controller (middle loop) takes as input z and I_{nadir} (image from the nadir-pointing camera at time t) and provides the roll, pitch, and thrust setpoints at 40 Hz to the attitude controller, which in turn, runs at 100 Hz.

2.2.3 Short baseline (ℓ_1) In case of short baseline, we detect if there is any rotational motion necessary for minimizing 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, ${}^{I_k^r}_{I_t}\mathbf{R}$, by employing the least-squares method of Horn (1987) on the features $f \in S_{SB}$ using as initial estimate the one from the minimal solver of the SB (2pt or 1pt+1) RANSAC. After extracting the yaw component, if $|{}^{I_k^r}\psi_{I_t}| > \tau_3$,[‡] we send the desired rotation-in-place motion $\mathbf{r}^T = [0 \;\; 0 \;\; {}^{I_k^r}\psi_{I_t}]^T$ to the controller to minimize the relative yaw between I_t, and I_k^r; else, we switch to the next reference image along the path $\mathcal{P}$.

Alternatively, we can leverage the yaw angle (computed off-line - see Sect. 2.1.2) between the first and last rotation-only reference images to speed up the execution of this path segment. In this case, 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.

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

- It enters a featureless region.
- It enters a region where the result from the FREAK feature matching between I_t and I_k^r is unreliable.

[‡]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.

- It 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 the returned images to find the one that has at least 20 features in common with I_t that pass the WB 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.

At this point, we should note that during our experiments when flying within the same floor (see Sect. 3.3) the quadrotor enters state ℓ_2 on average one to two times, primarily due to external disturbances (e.g., people walking in front of it or high-speed airflow from the air-conditioning vents). The number of times that the quadrotor gets lost increases to five when travelling across floors (see Sect. 3.4). This is due to the additional challenges that navigating through dark, featureless staircases poses. Note though that in all cases where the 3pt+1 RANSAC was employed for WB estimation, the quadrotor was able to relocalize and successfully complete its path.

2.3 Optical Flow

In this section, we describe our extension of the PX4Flow algorithm of Honegger et al. (2013). The original algorithm first extracts a 64×64 patch in the center of the downward-pointing camera's image and computes the optical flow of each of the 25 8×8 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 maximum number of votes as the optical flow for that frame.

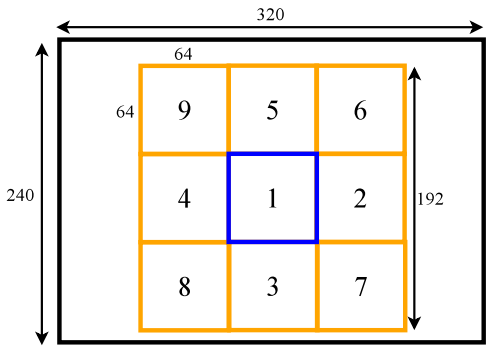

Figure 5. The image is divided into nine 64×64 pixel patches. The number indicates the order in which each patch is considered by the proposed optical-flow algorithm. Note that in Honegger et al. (2013), only the center patch (1) is used.

In poor lighting-conditions, however, or when flying over low-texture surfaces the patch in the center of the image may not have sufficient texture, or the minimum SAD for the chosen pixel blocks may be very large, leading to erroneous optical-flow estimation. To address this issue, we propose the following extension of the PX4Flow algorithm: First, we split the image of size 320×240 pixels into 9 patches each of pixel size 64×64 (see Fig. 5). Then, we start by computing the histogram of optical-flow based on the PX4Flow algorithm for the center patch of the image (i.e., patch 1 in Fig. 5) and count the number of valid pixel blocks. In particular, for a chosen pixel block $\mathbf{P}_b$, if the sum of horizontal and vertical gradients of the 4×4 patch centered in $\mathbf{P}_b$ (i.e., its textureness) is larger than a threshold γ_1 and the minimum SAD value in the search area is less than a threshold γ_2, $\mathbf{P}_b$ is considered to be a valid pixel block, where γ_1 and γ_2 are determined based on the camera's specifications. Subsequently, if the number of valid pixel blocks, among 25 chosen ones, is less than or equal to 20, we proceed to compute the optical flow from additional patches, and accumulate their histograms, following the patch order shown in Fig. 5. This process continues until the number of valid pixel blocks in the histogram is larger than 20. The reason behind this order of patch-selection (i.e., starting from the center and moving outward) is that the pixels far away from the center have more radial distortion. Furthermore, as evident from (10), they are affected by rotations more than the ones closest to the center:

$$\begin{aligned}\dot{u} &= \frac{-f_c v_x + u v_z}{\eta} - f_c\omega_y + v\omega_z + \frac{uv\omega_x - u^2\omega_y}{f_c} \\ \dot{v} &= \frac{-f_c v_y + v v_z}{z} + f_c\omega_x - u\omega_z + \frac{v^2\omega_x - uv\omega_y}{f_c}\end{aligned} \tag{10}$$

where f_c is the focal length of the camera, u and v are the coordinates of a pixel $\mathbf{p}$ w.r.t the center of the image, $\dot{u}$ and $\dot{v}$ are the optical-flow velocities of $\mathbf{p}$, (v_x, v_y, v_z) and $(\omega_x, \omega_y, \omega_z)$ are the linear and rotational velocities, respectively, of the downward-pointing camera in ego-centric coordinates, and z is the vertical coordinate of the 3D point to which $\mathbf{p}$ corresponds. Since our motions are mostly 2.5D, the distance of any 3D point in the camera's view is almost constant, and thus v_z is negligible. By employing this assumption, we can estimate v_x, v_y from the computed optical flow $\dot{u}, \dot{v}$, the rotational velocities $\omega_x, \omega_y, \omega_z$ estimated from the IMU, and the distance to the ground z measured by the ultrasonic sensor, as follows:

$$\begin{aligned}v_x &= \frac{z}{f_c}(-\dot{u} - f_c\omega_y + v\omega_z + \frac{uv\omega_x - u^2\omega_y}{f_c}) \\ v_y &= \frac{z}{f_c}(-\dot{v} + f_c\omega_x - u\omega_z + \frac{v^2\omega_x - uv\omega_y}{f_c})\end{aligned}$$

3 Experimental Results

In this section, we present experimental results for validating both our extension of the PX4FLow algorithm and the ability of the quadrotor to autonomously fly through image-defined paths. In the optical-flow experiment (see Sect. 3.2), we show the difference in performance between our proposed and the original approach using data acquired in a dark staircase inside the Walter Library. Next, in

Sect.s 3.3 and 3.4, respectively, we describe autonomous navigation experiments with two Parrot Bebop quadrotors in two scenarios: (i) A 75 m indoor area, within the same floor, to evaluate the algorithm's performance when moving relatively fast [the velocity, v_0 in (4), was set to 2 m/sec, which is the maximum velocity that can be measured by the optical-flow algorithm]; (ii) A 150 m indoor area that requires transitioning between two floors through two staircases. In this case, v_0 was set to 1.2 m/sec.

3.1 System setup

The Bebop carries an Invensense MPU-6050 MEMS IMU, a downward-pointing Aptina MT9V117 camera (53 deg FOV, set to 320×240 pixels resolution) used for optical flow, an ultrasonic sensor for measuring the distance to the ground, and a forward-facing Aptina MT9F002 camera (180 deg FOV, set to 300×264 pixels resolution) that we use for visual navigation. All computations are performed in real-time onboard Bebop's ARM Cortex A9 800 MHz dual-core processor.

3.2 Optical-flow experiment

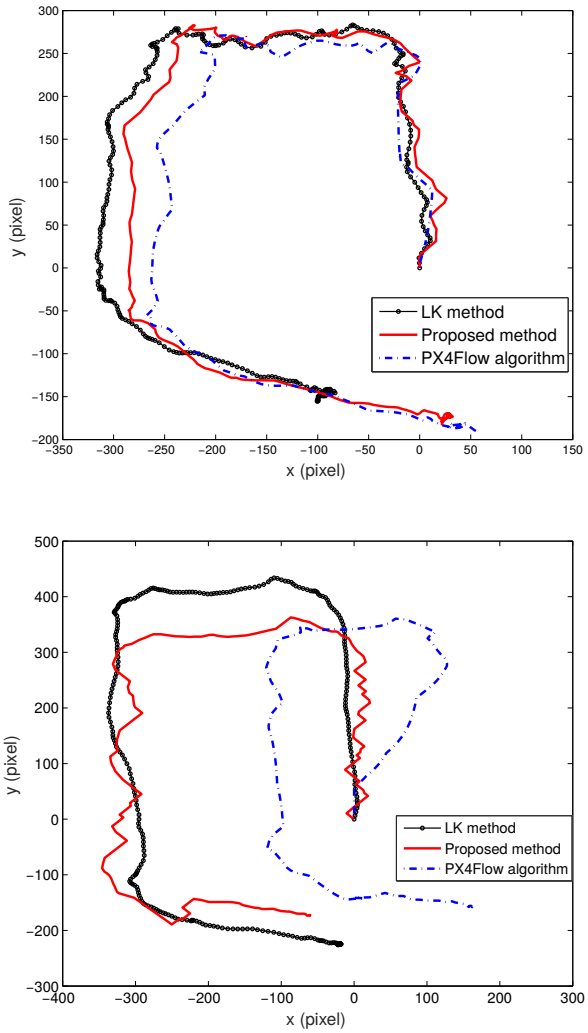

Figure 6. Comparison of different methods [Lucas-Kanade (LK), PX4Flow, and our proposed extension] for computing and integrating the optical flow over 2 datasets for slow (top) and fast (bottom) motions.

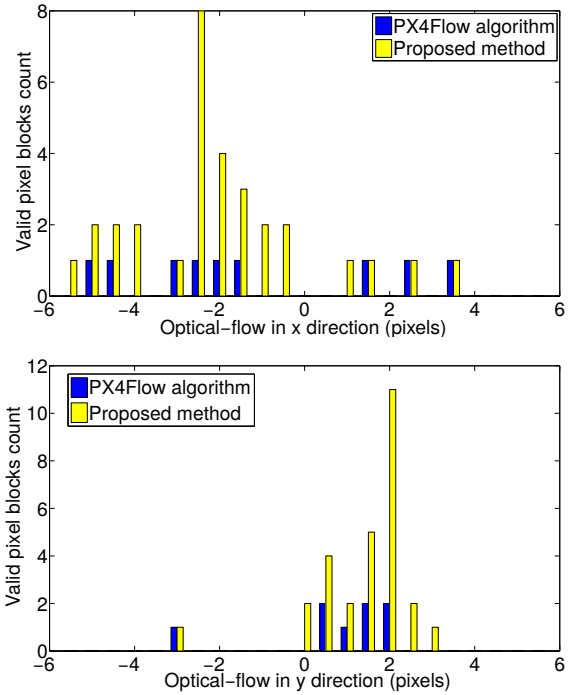

Figure 7. Histograms of the optical-flow in the x and y directions. The blue bars depict the histograms from the PX4Flow algorithm. The yellow bars indicate the histograms from our proposed extension to the PX4Flow algorithm.

We implemented the proposed optical-flow algorithm using ARM NEON and achieved an average time of 0.95 ms for images of size 320×240 pixels. Thus, for a frame rate of 60 Hz, the total time (per second of flight) for computing the optical-flow is approximately 57 ms which is sufficient for real-time operation. To demonstrate the robustness of our proposed extension, we collected 2 datasets of roughly 300 images using the Parrot Bebop at the Walter Library stairs; the first one is with a moderate speed (0.5 m/sec) while the second one is relatively faster (2.0 m/sec). Subsequenty, we integrated the flow estimates

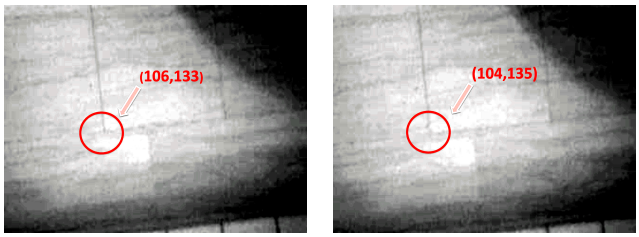

Figure 8. Representative consecutive image pairs from the Walter Library stairs. The red circle shows the same corner in the two images with their pixel coordinates.

from consecutive image pairs, for (i) the PX4Flow algorithm, (ii) our proposed extension, and (iii) the Lucas-Kanade (LK) method, which is considered as groundtruth. The resulting paths, in pixel space, are shown in Fig. 6. As evident, our algorithm is significantly more robust than the original PX4Flow algorithm under fast motions, while it has similar accuracy and processing requirements when moving slowly. In order to better understand where the gain in performance comes from, in Fig. 7, we show the histograms of the pixel blocks' displacements resulting from the image pair

shown in Fig. 8. Notice that the blue bars in Fig. 7, 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.

3.3 *Experiments Set 1 (within the same floor)*

Figure 9. Experiments Set 1: (Top) Blueprint of the experimental area with the reference trajectory (1-. . . -6-1) overlaid. (Bottom) Snapshots of the Bebop along its path (locations 1-6).

In these experiments, which took place in the Walter Library's basement, the two quadrotors used 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. 9 shows the blueprint of the experimental area with the reference visual path (red bold line) overlaid, as well as snapshots of the quadrotor in flight. Figs. 11 and 10 show examples of matched feature inliers between the current and reference images for the WB and SB cases, respectively. In both instances, the 3p+1 and 1p+1 RANSAC inlier ratios (0.675 and 0.92 in Figs. 11 and 10, respectively), were used to determine whether the configuration between the current, I_t, and reference, I_k^r, images corresponds to a WB or SB case.

During the experiment, and depending on the WB vs. SB choice, the Bebop was able to complete the reference trajectory (see Table 1) in a total of 97-240 sec (Tot. Time). Within this interval, the quadrotor performs mostly

Exp.	Len.	Tot. time	Trans. time	Avg. speed
3pt+1	75 m	97 sec	73 sec	1.0 m/s
3pt+1	75 m	130 sec	84 sec	0.9 m/s
3pt+1	75 m	133 sec	96 sec	0.8 m/s
5pt	75 m	210 sec	146 sec	0.5 m/s
5pt	75 m	240 sec	180 sec	0.4 m/s

Table 1. Performance comparison between the 3pt+1/1pt+1 and 5pt/2pt-based autonomous navigation algorithms for the single-floor experiments (Set 1).

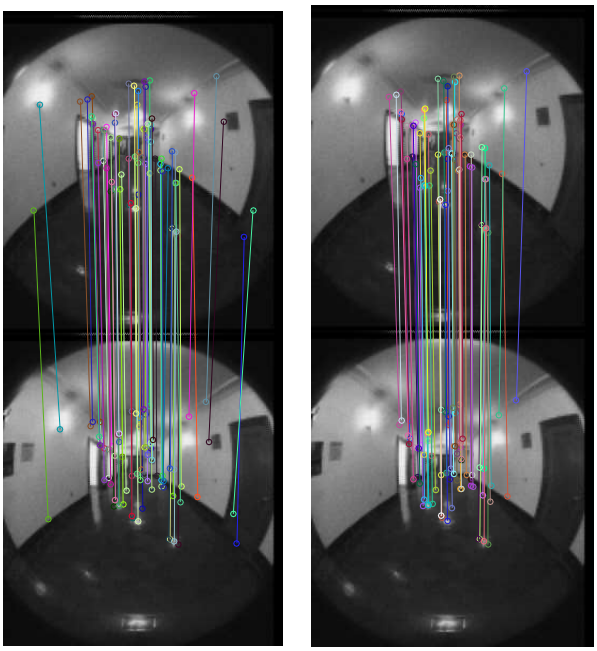

Figure 10. Short baseline case: 3pt+1 (left) and 1pt+1 (right) RANSAC-based matching between 2 pairs of images (top I_k^r, bottom I_t) from the Bebop's forward-facing camera.

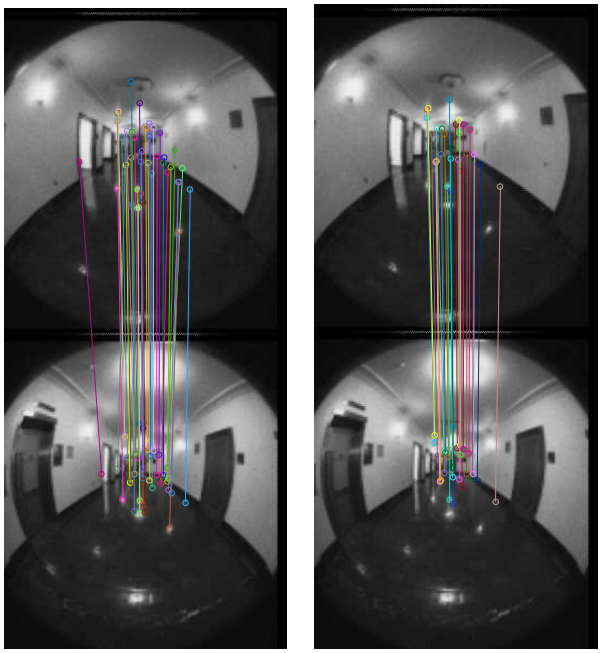

Figure 11. Wide baseline case: 3pt+1 (left) and 1pt+1 (right) RANSAC-based matching between 2 pairs of images (top I_k^r, bottom I_t) from the Bebop's forward-facing camera.

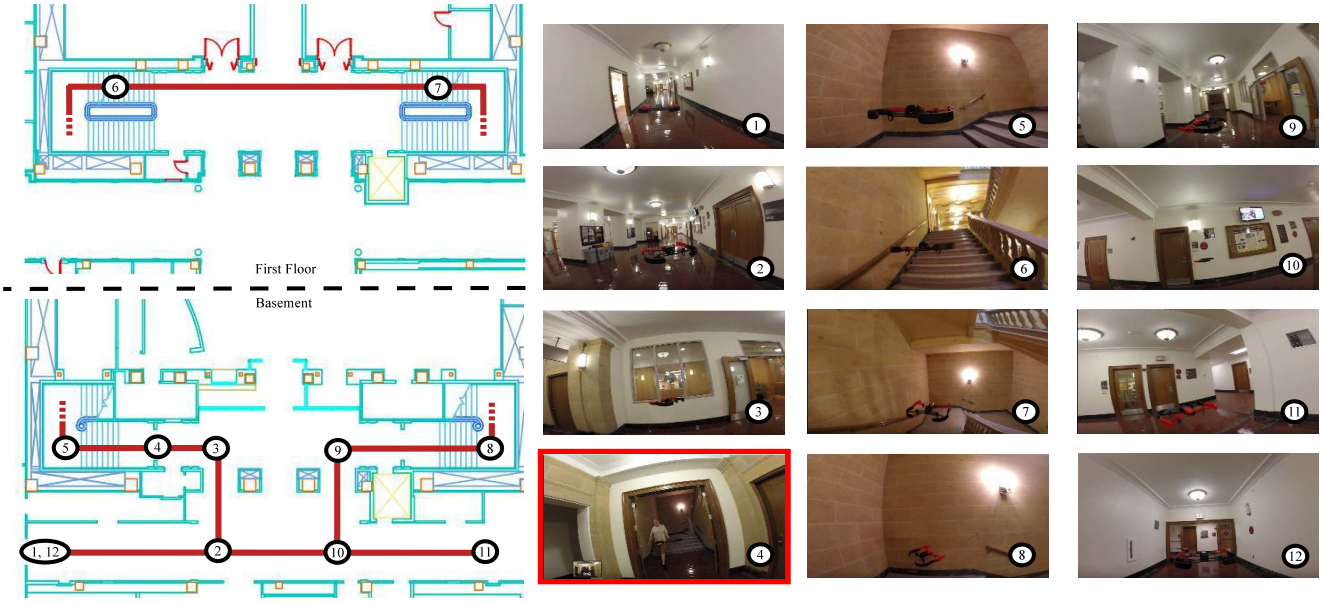

Figure 13. Experiment 2: Blueprint of the experimental area, the reference path, and the snapshot of the Bebop during flights. Note that the stairs area is shown on images 5-8.

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

Table 2. Execution time comparison between the 3pt+1 and 5pt minimal solvers and corresponding RANSACs on the Bebop's processor.

translational motion (trans. time) for 73-180 sec, at an average speed (avg. speed) of 1.0-0.4 m/s. Specifically, Table 1 summarizes the performance results, in terms of speed, from 5 experiments where two different options for WB and SB RANSAC were employed. In particular, in the first three experiments, the 3pt+1 along with 1pt+1 algorithm were used in the WB and SB RANSACs, respectively, achieving an up to 2.5 times speedup as compared to the case when the 5pt and 2pt algorithms were used instead. The significant gain in navigation speed is better explained by noting the substantial difference in the processing requirements between the 5pt and 3pt+1 minimal solvers, and thus among the corresponding RANSACs. These timing results for the Bebop's processor are listed in Table 2. As evident from these comparisons, employing the gravity direction in the motion-estimation algorithm leads to significant gains in speed during autonomous visual navigation.

3.4 *Experiments Set 2 (flight across two floors through stairs)*

This set of experiments took place in the Walter Library's basement and 1st floor, which are connected through 2 staircases (south-north), each comprising two flights of stairs. In this situation and due to the lengthier trajectory (150 m), the quadrotor is more likely to lose track of the reference image, and enter the lost mode (Fig. 12 shows an example of an online query image and the returned images, along with the number of inlier matches from the VT).

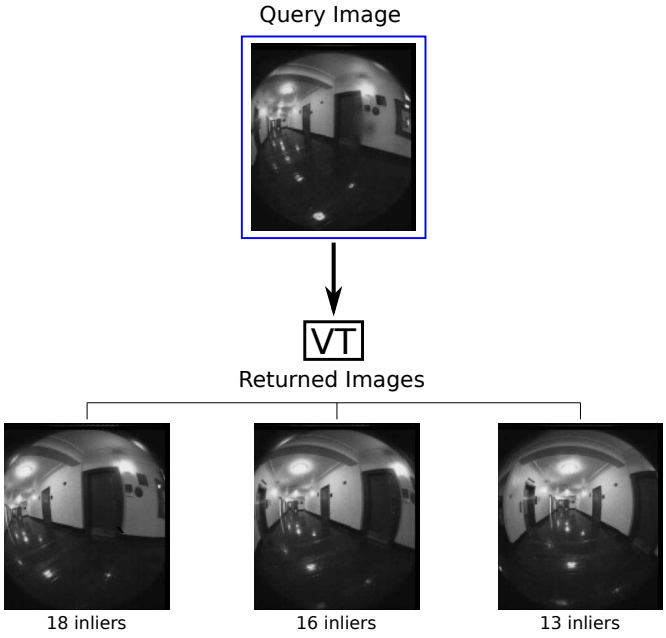

Figure 12. An example of an online image query when in lost mode. The top image is the one that the quadrotor acquired online but was not able to match with the reference image. The bottom 3 images are the ones returned by the VT after querying with the current quadrotor online image.

Furthermore, the quadrotor has to fly through two staircases, where the high rate of change of the ultrasonic sensor's height measurement caused the vision-only approach (5pt + 2pt RANSAC) to fail to complete the trajectory. The main challenge in this experiment was accurately estimating the optical flow. In particular, the stairs comprise textureless steps and part of the staircases (transitions between images 4-6 and 7-9 in Fig. 13) is featureless and quite dark as compared to the rest of the path. In addition, there was regular pedestrian traffic in the experimental area with people often walking in front of the quadrotor (see Fig. 14), disturbing its path-following process. Despite the adverse

Figure 14. Example images of people walking or standing in Bebop's path.

conditions, the quadrotor was able to successfully navigate through this path, and achieved the performance summarized in Table 3. Fig. 13 shows the blueprint of the experimental area with the reference visual path (red bold line) overlaid, as well as snapshots of the Bebop in flight. The videos of both sets of experiments are available at Do et al. (2016).

Exp.	Len.	Tot. time	Trans. time	Avg. speed
3pt+1	150 m	250 sec	154 sec	1.0 m/s
3pt+1	150 m	235 sec	181 sec	0.8 m/s
3pt+1	150 m	274 sec	213 sec	0.7 m/s

Table 3. Performance of the 3pt+1/1pt+1-based autonomous navigation algorithm for the multi-floor experiments (Set 2).

4 Conclusion and Future Work

In this paper, we presented a visual-servoing algorithm for autonomous quadrotor navigation 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 carrying the quadrotor. Specifically, the visual map is represented as a graph of images linked with edges whose weights (cost to traverse) are inversely proportional to their number of common features. Once the visual graph is constructed, and given as input the start, intermediate, and goal locations of the quadrotor, it automatically generates the desired path as a sequence of reference images. This information is then 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 WB and SB RANSAC algorithms online for (i) determining the type of desired motion (translation and rotation versus rotation in place) and (ii) selecting the next reference image, the quadrotor is able to reliably navigate through areas comprising lenthgy corridors, as well as narrow passages. Additionally, it is able to cope with static and moving obstacles and recover its path after losing track of its reference image. Moreover, we have shown that by employing information about the direction of gravity in the WB and SB RANSAC algorithms (i.e., using the 3pt+1/1pt+1 instead of the 5pt/2pt minimal solvers) we are able to realize significant gains in processing, and hence speed of navigation. Lastly, critical improvements in robustness, especially when flying over low-texture areas, were achieved by extending the PF4Flow optical-flow algorithm to progressively use larger parts of the downward-pointing camera's images for estimating the vehicle's horizontal velocity. The performance of the proposed autonomous navigation algorithm was assessed in two sets of experiments over two lengthy paths (75 m and 150 m), across two floors, and under challenging conditions (e.g., moving obstacles, specular reflections, featureless corridors, textureless stairs, dark areas) while running in real-time on the resource-constrained processor of a commercial-of-the-self, low-cost quadrotor.

As part of our future work, we plan to assess and improve the performance of our autonomous quadrotor navigation algorithm for the case where the images used for constructing the visual map were recorded by a different camera (e.g., from another type of quadrotor). Robustness to large changes in the appearance of areas of the building, as well as the lighting conditions are also within our future interests.

5 Acknowledgements

We would like to thank Zhengqi Li for implementing the ARM NEON optimized version of the optical-flow algorithm presented in this paper.

A Appendix

A.1 2pt/1pt+1 RANSAC minimal solver

Consider two unit bearing measurements ${}^{I_1}\mathbf{b}_{f_i}$, ${}^{I_2}\mathbf{b}_{f_i}$ to a feature f_i from two images, and assume that the motion between them is purely rotational, i.e.,

$$ {}^{I_2}\mathbf{b}_{f_i} = \mathbf{R}({}^{I_2}_{I_1}\bar{q})\,{}^{I_1}\mathbf{b}_{f_i}, \tag{11} $$

where ${}^{I_2}_{I_1}\bar{q}$ is the unit quaternion of rotation. When only visual information is available, finding ${}^{I_2}_{I_1}\bar{q}$ requires two feature matches between I_1 and I_2 to satisfy (11). We refer to this as the 2pt-minimal problem. On the other hand, when the direction of gravity is known for both images, i.e.,

$$ {}^{I_2}\hat{\mathbf{g}} = \mathbf{R}({}^{I_2}_{I_1}\bar{q})\,{}^{I_1}\hat{\mathbf{g}} \tag{12} $$

we only need one feature match to satisfy (11). We refer to this problem as the 1pt+1 minimal problem.

In summary, in both cases, we need 2 pairs of linearly independent unit vectors $(\mathbf{u}_1, \mathbf{u}_2)$, and $(\mathbf{v}_1, \mathbf{v}_2)$, to satisfy (11) or (12), [or equivalently (13) below] where $\mathbf{u}_1 = {}^{I_1}\mathbf{b}_{f_1}, \mathbf{v}_1 = {}^{I_2}\mathbf{b}_{f_1}$, while $\mathbf{u}_2 = {}^{I_1}\mathbf{b}_{f_2}, \mathbf{v}_2 = {}^{I_2}\mathbf{b}_{f_2}$ or

$\mathbf{u}_2 = {}^{I_1}\hat{\mathbf{g}}, \mathbf{v}_2 = {}^{I_2}\hat{\mathbf{g}}$. In what follows, we prove the following theorem:

Theorem 1. *Given two pairs of unit vectors* $(\mathbf{u}_1, \mathbf{u}_2)$, *and* $(\mathbf{v}_1, \mathit{capitallettersmathbf}v_2)$, *where* $\mathbf{u}_1$ *and* $\mathbf{u}_2$ *are linearly independent, satisfying:*

$$\mathbf{v}_i = \mathbf{R}(\bar{q})\mathbf{u}_i \quad i = 1, 2 \tag{13}$$

the unknown quaternion of rotation $\bar{q}^T = \begin{bmatrix}\mathbf{q} & q_4\end{bmatrix}^T$ *can be found as:*

if $(\mathbf{v}_1 - \mathbf{u}_1) \times (\mathbf{v}_2 - \mathbf{u}_2) \neq \mathbf{0}$ ***then***

$$\bar{q} = \gamma \begin{bmatrix}(\mathbf{v}_1 - \mathbf{u}_1) \times (\mathbf{v}_2 - \mathbf{u}_2) \\ (\mathbf{v}_1 + \mathbf{u}_1)^T(\mathbf{v}_2 - \mathbf{u}_2)\end{bmatrix}$$

else

if $\mathbf{v}_1^T(\mathbf{u}_1 \times \mathbf{u}_2) \neq 0$ ***then***

$$\bar{q} = \eta \begin{bmatrix}(\mathbf{v}_1 \times \mathbf{v}_2) \times (\mathbf{u}_1 \times \mathbf{u}_2) \\ (\mathbf{v}_1 \times \mathbf{v}_2)^T(\mathbf{v}_1 \times \mathbf{v}_2 + \mathbf{u}_1 \times \mathbf{u}_2)\end{bmatrix}$$

else

if $\mathbf{u}_1 = \mathbf{v}_1$ *and* $\mathbf{u}_2 = \mathbf{v}_2$ ***then***

$$\bar{q} = \begin{bmatrix}\mathbf{0} \\ 1\end{bmatrix}$$

else

$$\bar{q} = \tfrac{1}{\|\mathbf{v}_i + \mathbf{u}_i\|}\begin{bmatrix}\mathbf{v}_i + \mathbf{u}_i \\ 0\end{bmatrix}, \ \mathbf{v}_i + \mathbf{u}_i \neq \mathbf{0}, \ i \in \{1, 2\}$$

end if

end if

end if

Proof:

In what follows, we present an algebraic derivation of the preceding relations. A geometry-based proof of this result is shown in Reynolds (1998).

Employing the quaternion relations in Trawny and Roumeliotis (2005) we have:

$$\begin{aligned}
&\mathbf{v}_i = \mathbf{R}(\bar{q})\mathbf{u}_i \\
\Leftrightarrow &\begin{bmatrix}\mathbf{v}_i \\ 0\end{bmatrix} = \bar{q} \otimes \begin{bmatrix}\mathbf{u}_i \\ 0\end{bmatrix} \otimes \bar{q}^{-1} \\
\Leftrightarrow &\begin{bmatrix}\mathbf{v}_i \\ 0\end{bmatrix} \otimes \bar{q} - \bar{q} \otimes \begin{bmatrix}\mathbf{u}_i \\ 0\end{bmatrix} = \mathbf{0} \\
\Leftrightarrow &\left[\mathscr{L}\left(\begin{bmatrix}\mathbf{v}_i \\ 0\end{bmatrix}\right) - \mathscr{R}\left(\begin{bmatrix}\mathbf{u}_i \\ 0\end{bmatrix}\right)\right]\begin{bmatrix}\mathbf{q} \\ q_4\end{bmatrix} = \mathbf{0} \\
\Leftrightarrow &\left[\begin{bmatrix}-\lfloor \mathbf{v}_{i\times}\rfloor & \mathbf{v}_i \\ -\mathbf{v}_i^T & 0\end{bmatrix} - \begin{bmatrix}\lfloor \mathbf{u}_{i\times}\rfloor & \mathbf{u}_i \\ -\mathbf{u}_i^T & 0\end{bmatrix}\right]\begin{bmatrix}\mathbf{q} \\ q_4\end{bmatrix} = \mathbf{0} \\
\Leftrightarrow &\begin{bmatrix}-\lfloor \mathbf{v}_i + \mathbf{u}_{i\times}\rfloor & \mathbf{v}_i - \mathbf{u}_i \\ -(\mathbf{v}_i - \mathbf{u}_i)^T & 0\end{bmatrix}\begin{bmatrix}\mathbf{q} \\ q_4\end{bmatrix} = \mathbf{0}
\end{aligned}$$

$$\Leftrightarrow \begin{cases} -\lfloor \mathbf{v}_i + \mathbf{u}_{i\times}\rfloor \mathbf{q} + (\mathbf{v}_i - \mathbf{u}_i)q_4 = \mathbf{0} & (14) \\ (\mathbf{v}_i - \mathbf{u}_i)^T\mathbf{q} = 0 & (15)\end{cases}$$

A.1.1 Case 1 [C1]: $(\mathbf{v}_1 - \mathbf{u}_1) \times (\mathbf{v}_2 - \mathbf{u}_2) \neq \mathbf{0}$

From (15) for $i = 1, 2, \exists\, \gamma \neq 0$:

$$\mathbf{q} = \gamma(\mathbf{v}_1 - \mathbf{u}_1) \times (\mathbf{v}_2 - \mathbf{u}_2) \tag{16}$$

Substituting (16) in (14) for $i = 1$ yields:

$$\begin{aligned}
&(\mathbf{v}_1 - \mathbf{u}_1)q_4 = \gamma\lfloor \mathbf{v}_1 + \mathbf{u}_{1\times}\rfloor(\mathbf{v}_1 - \mathbf{u}_1) \times (\mathbf{v}_2 - \mathbf{u}_2) \\
&\Leftrightarrow (\mathbf{v}_1 - \mathbf{u}_1)q_4 = \gamma(\mathbf{v}_1 - \mathbf{u}_1)(\mathbf{v}_1 + \mathbf{u}_1)^T(\mathbf{v}_2 - \mathbf{u}_2) \quad (17) \\
&\Rightarrow q_4 = \gamma(\mathbf{v}_1 + \mathbf{u}_1)^T(\mathbf{v}_2 - \mathbf{u}_2) \quad (18)
\end{aligned}$$

where (17) is obtained by $\lfloor \mathbf{a}_\times\rfloor\lfloor \mathbf{b}_\times\rfloor = \mathbf{b}\mathbf{a}^T - (\mathbf{a}^T\mathbf{b})\mathbf{I}$, while q_4 in (18) is found by noting that $\mathbf{v}_1 \neq \mathbf{u}_1$, else **[C1]** will not hold. Note that, the solution will not change if we find q_4 by substituting $\mathbf{q}$ into (14) for $i = 2$ (instead of $i = 1$). In such case, we will get $q_4' = -\gamma(\mathbf{v}_2 + \mathbf{u}_2)^T(\mathbf{v}_1 - \mathbf{u}_1)$. Note though that:

$$\begin{aligned}
\frac{q_4}{\gamma} - \frac{q_4'}{\gamma} &= (\mathbf{v}_1 + \mathbf{u}_1)^T(\mathbf{v}_2 - \mathbf{u}_2) + (\mathbf{v}_2 + \mathbf{u}_2)^T(\mathbf{v}_1 - \mathbf{u}_1) \\
&= 2(\mathbf{v}_1^T\mathbf{v}_2 - \mathbf{u}_1^T\mathbf{u}_2) \\
&= 2(\mathbf{u}_1^T\mathbf{R}^T(\bar{q})\mathbf{R}(\bar{q})\mathbf{u}_2 - \mathbf{u}_1^T\mathbf{u}_2) \\
&= 0
\end{aligned}$$

and thus $q_4 = q_4'$. Hence, under **[C1]**, we obtain the quaternion solution:

$$\bar{q} = \begin{bmatrix}\mathbf{q} \\ q_4\end{bmatrix} = \gamma\begin{bmatrix}(\mathbf{v}_1 - \mathbf{u}_1) \times (\mathbf{v}_2 - \mathbf{u}_2) \\ (\mathbf{v}_1 + \mathbf{u}_1)^T(\mathbf{v}_2 - \mathbf{u}_2)\end{bmatrix} \tag{19}$$

where γ is the normalization constant that ensures unit length.

A.1.2 Case 2 [C2]: $(\mathbf{v}_1 - \mathbf{u}_1) \times (\mathbf{v}_2 - \mathbf{u}_2) = \mathbf{0}$

This condition means that $\exists\, \alpha \neq 0$ such that:

$$\begin{aligned}
&\mathbf{v}_1 - \mathbf{u}_1 = \alpha(\mathbf{v}_2 - \mathbf{u}_2), \\
\Leftrightarrow\ &\mathbf{R}(\bar{q})\mathbf{u}_1 - \mathbf{u}_1 = \alpha(\mathbf{R}(\bar{q})\mathbf{u}_2 - \mathbf{u}_2) \\
\Leftrightarrow\ &\mathbf{R}(\bar{q})(\mathbf{u}_1 - \alpha\mathbf{u}_2) = \mathbf{u}_1 - \alpha\mathbf{u}_2 \quad (20) \\
\Leftrightarrow\ &\mathbf{R}(\bar{q})^T(\mathbf{v}_1 - \alpha\mathbf{v}_2) = \mathbf{v}_1 - \alpha\mathbf{v}_2 \quad (21)
\end{aligned}$$

From (20), we conclude that $\mathbf{u}_1 - \alpha\mathbf{u}_2$ is an eigenvector corresponding to the eigenvalue 1 of the rotational matrix $\mathbf{R}(\bar{q})$, and thus, it is colinear with the unit vector of rotation $\mathbf{k}$ and the corresponding quaternion vector $\mathbf{q}$. As a consequence, $\mathbf{u}_1, \mathbf{u}_2$, and $\mathbf{q}$ are coplanar. Analogously, from (21), we conclude that $\mathbf{v}_1, \mathbf{v}_2$, and $\mathbf{q}$ are coplanar.

A.1.3 Subcase 2a [C2a]: $\mathbf{v}_1^T(\mathbf{u}_1 \times \mathbf{u}_2) \neq 0$

Under this configuration (see Fig. 15), the plane Π_u (formed by $\mathbf{u}_1, \mathbf{u}_2$) intersects the plane Π_v (formed by $\mathbf{v}_1, \mathbf{v}_2$) at a unique line, with the same direction as the vector

$$\mathbf{q} = \eta(\mathbf{v}_1 \times \mathbf{v}_2) \times (\mathbf{u}_1 \times \mathbf{u}_2), \quad \eta \neq 0 \tag{22}$$

Substituting $\mathbf{q}$ from (22) into (14) for $i = 1$ yields:

$$\begin{aligned}
&(\mathbf{v}_1 - \mathbf{u}_1)q_4 \\
&\quad = \eta\lfloor \mathbf{v}_1 + \mathbf{u}_{1\times}\rfloor\lfloor \mathbf{v}_1 \times \mathbf{v}_{2\times}\rfloor(\mathbf{u}_1 \times \mathbf{u}_2) \\
&\quad = \eta((\mathbf{v}_1 + \mathbf{u}_1)^T(\mathbf{u}_1 \times \mathbf{u}_2)(\mathbf{v}_1 \times \mathbf{v}_2) \\
&\qquad - (\mathbf{v}_1 + \mathbf{u}_1)^T(\mathbf{v}_1 \times \mathbf{v}_2)(\mathbf{u}_1 \times \mathbf{u}_2)) \quad (23) \\
&\quad = \eta\left(\mathbf{v}_1^T\lfloor \mathbf{u}_{1\times}\rfloor\mathbf{u}_2(\mathbf{v}_1 \times \mathbf{v}_2) - \mathbf{u}_1^T\lfloor \mathbf{v}_{1\times}\rfloor\mathbf{v}_2(\mathbf{u}_1 \times \mathbf{u}_2)\right) \quad (24)
\end{aligned}$$

where (23) is obtained using the identity $\lfloor \mathbf{a}_\times\rfloor\lfloor \mathbf{b}_\times\rfloor\mathbf{c} = (\mathbf{a}^T\mathbf{c})\mathbf{b} - (\mathbf{a}^T\mathbf{b})\mathbf{c}$, while (24) results from the property of the cross-product $\mathbf{a}^T(\mathbf{a} \times \mathbf{b}) = 0$. Next, we note that from **[C2]**:

$$\begin{aligned}
&(\mathbf{v}_1 - \mathbf{u}_1) \times (\mathbf{v}_2 - \mathbf{u}_2) = \mathbf{0} \\
\Rightarrow\ &\mathbf{v}_1^T\lfloor \mathbf{v}_1 - \mathbf{u}_{1\times}\rfloor(\mathbf{v}_2 - \mathbf{u}_2) = 0 \\
\Rightarrow\ &-\mathbf{v}_1^T\lfloor \mathbf{u}_{1\times}\rfloor(\mathbf{v}_2 - \mathbf{u}_2) = 0 \\
\Rightarrow\ &\mathbf{v}_1^T\lfloor \mathbf{u}_{1\times}\rfloor\mathbf{u}_2 = -\mathbf{u}_1^T\lfloor \mathbf{v}_{1\times}\rfloor\mathbf{v}_2 \quad (25)
\end{aligned}$$

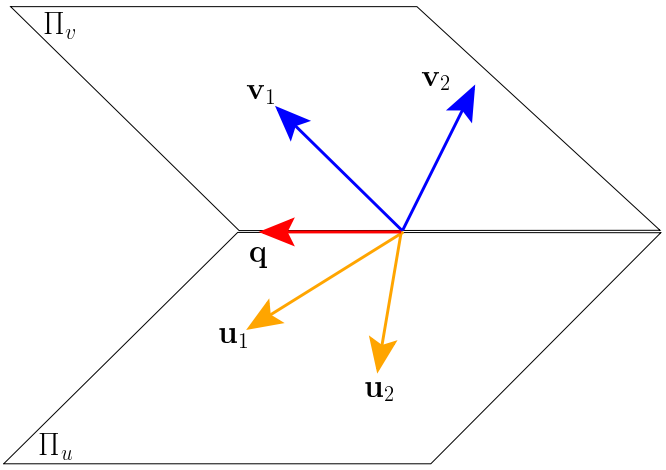

Figure 15. Geometric interpretation of case [**C2a**]: The plane Π_u, comprising vectors $\mathbf{u}_1, \mathbf{u}_2$, intersects the plane Π_v, comprising vectors $\mathbf{v}_1, \mathbf{v}_2$, at a line with direction $\mathbf{q}$.

Employing (25), (24) can be written as:

$$(\mathbf{v}_1 - \mathbf{u}_1)q_4 = \eta\, \mathbf{v}_1^T \lfloor \mathbf{u}_{1\times} \rfloor \mathbf{u}_2 \left((\mathbf{v}_1 \times \mathbf{v}_2) + (\mathbf{u}_1 \times \mathbf{u}_2)\right) \tag{26}$$

Projecting both sides of (26) on $(\mathbf{v}_1 \times \mathbf{v}_2)$ yields:

$$(\mathbf{v}_1 \times \mathbf{v}_2)^T(\mathbf{v}_1 - \mathbf{u}_1)q_4 = \eta\, \mathbf{v}_1^T \lfloor \mathbf{u}_{1\times} \rfloor \mathbf{u}_2 \left(\|(\mathbf{v}_1 \times \mathbf{v}_2)\|^2 + (\mathbf{v}_1 \times \mathbf{v}_2)^T(\mathbf{u}_1 \times \mathbf{u}_2)\right) \tag{27}$$

$$\Rightarrow (-\mathbf{u}_1{}^T \lfloor \mathbf{v}_{1\times} \rfloor \mathbf{v}_2)\, q_4 = \eta\, \mathbf{v}_1^T \lfloor \mathbf{u}_{1\times} \rfloor \mathbf{u}_2 \left(\|(\mathbf{v}_1 \times \mathbf{v}_2)\|^2 + (\mathbf{v}_1 \times \mathbf{v}_2)^T(\mathbf{u}_1 \times \mathbf{u}_2)\right) \tag{28}$$

$$\Rightarrow q_4 = \eta\, (\mathbf{v}_1 \times \mathbf{v}_2)^T (\mathbf{v}_1 \times \mathbf{v}_2 + \mathbf{u}_1 \times \mathbf{u}_2) \tag{29}$$

where to arrive to (29), we employ again (25).

Summarizing (22) and (29), the solution for [**C2a**] is:

$$\bar{q} = \eta \begin{bmatrix} (\mathbf{v}_1 \times \mathbf{v}_2) \times (\mathbf{u}_1 \times \mathbf{u}_2) \\ (\mathbf{v}_1 \times \mathbf{v}_2)^T(\mathbf{v}_1 \times \mathbf{v}_2 + \mathbf{u}_1 \times \mathbf{u}_2) \end{bmatrix} \tag{30}$$

where η is the normalization constant that ensures unit length.

A.1.4 Subcase 2b [C2b]: $\mathbf{v}_1^T(\mathbf{u}_1 \times \mathbf{u}_2) = 0$

This condition means that $\mathbf{u}_1, \mathbf{u}_2, \mathbf{v}_1, \mathbf{v}_2$, and $\mathbf{q}$ are all coplanar (see Fig. 16). We assume that $\mathbf{q} \neq \mathbf{0}$, otherwise $\bar{q}$ is

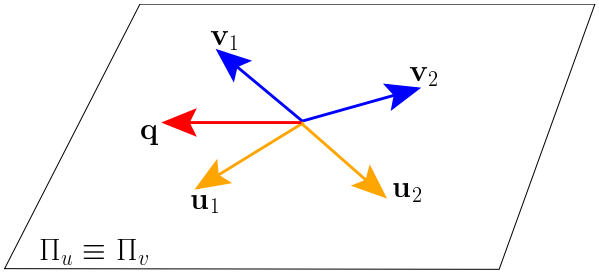

Figure 16. Geometric interpretation of case [**C2b**]: The planes Π_u and Π_v coincide, thus the 5 vectors $\mathbf{u}_1, \mathbf{u}_2, \mathbf{v}_1, \mathbf{v}_2$, and $\mathbf{q}$ are coplanar.

the unit quaternion. In such case, we first show that $q_4 = 0$. Specifically, we use the Rodriquez formula for expressing the rotation matrix in terms of the quaternion (Trawny and Roumeliotis 2005) and expand (13) as follows:

$$\begin{aligned} \mathbf{v}_i &= \mathbf{R}(\bar{q})\mathbf{u}_i \\ &= ((2q_4^2 - 1)\mathbf{I} - 2q_4\lfloor \mathbf{q}_\times \rfloor + 2\mathbf{q}\mathbf{q}^T)\,\mathbf{u}_i \\ &= (2q_4^2 - 1)\mathbf{u}_i + 2(\mathbf{q}^T\mathbf{u}_i)\mathbf{q} - 2q_4(\mathbf{q} \times \mathbf{u}_i) \end{aligned} \tag{31}$$

Note that $(\mathbf{q} \times \mathbf{u}_i)$ is perpendicular to all other vectors appearing in (31). Thus, projecting both sides of (31) on $(\mathbf{q} \times \mathbf{u}_i)$ yields $q_4 = 0$. Substituting q_4 back in (31) results in:

$$\mathbf{u}_i + \mathbf{v}_i = 2(\mathbf{q}^T\mathbf{u}_i)\mathbf{q}, \quad i = 1, 2 \tag{32}$$

If $\mathbf{u}_1 + \mathbf{v}_1 = \mathbf{u}_2 + \mathbf{v}_2 = \mathbf{0}$, we employ the assumption $\mathbf{q} \neq \mathbf{0}$, (32) leads to:

$$\mathbf{q}^T\mathbf{u}_i = 0, \quad i = 1, 2 \tag{33}$$

Note that since $\mathbf{u}_1, \mathbf{u}_2$, and $\mathbf{q}$ are coplanar, (33) infers that $\mathbf{u}_1$ and $\mathbf{u}_2$ are colinear. This contradicts the linear independent assumption of $\mathbf{u}_1$ and $\mathbf{u}_2$.

Now, we consider the case where $\mathbf{u}_i + \mathbf{v}_i \neq \mathbf{0}$ and $\mathbf{u}_j + \mathbf{v}_j = \mathbf{0}$, where $i \neq j$ and $i, j \in \{1, 2\}$. From (32), we have:

$$\begin{aligned} &\mathbf{q} = \rho(\mathbf{u}_i + \mathbf{v}_i) \quad \rho \neq 0 \\ \Rightarrow\ &\mathbf{q} = \frac{1}{\|\mathbf{u}_i + \mathbf{v}_i\|}(\mathbf{u}_i + \mathbf{v}_i) \end{aligned} \tag{34}$$

Note that this choice of $\mathbf{q}$ in (34) also satisfies $\mathbf{q}^T\mathbf{u}_j = 0$, or equivalently:

$$\begin{aligned} (\mathbf{u}_i + \mathbf{v}_i)^T\mathbf{u}_j &= \mathbf{u}_i^T\mathbf{u}_j + \mathbf{v}_i^T\mathbf{u}_j \\ &= \mathbf{u}_i^T\mathbf{R}^T(\bar{q})\mathbf{R}(\bar{q})\mathbf{u}_j + \mathbf{v}_i^T\mathbf{u}_j \\ &= \mathbf{v}_i^T\mathbf{v}_j - \mathbf{v}_i^T\mathbf{v}_j = 0 \end{aligned} \tag{35}$$

where (35) is obtained by noting that $\mathbf{u}_j = -\mathbf{v}_j$. Thus with $\mathbf{u}_i + \mathbf{v}_i \neq \mathbf{0}$, we have:

$$\bar{q} = \frac{1}{\|\mathbf{u}_i + \mathbf{v}_i\|} \begin{bmatrix} \mathbf{u}_i + \mathbf{v}_i \\ 0 \end{bmatrix} \tag{36}$$

Finally, when $\mathbf{u}_i + \mathbf{v}_i \neq \mathbf{0}$ and $\mathbf{u}_j + \mathbf{v}_j \neq \mathbf{0}$, (32) shows that:

$$(\mathbf{u}_i + \mathbf{v}_i) \times (\mathbf{u}_j + \mathbf{v}_j) = \mathbf{0}$$

or equivalently, $\mathbf{q}$, $\mathbf{u}_i + \mathbf{v}_i$, and $\mathbf{u}_j + \mathbf{v}_j$ are all colinear. Therefore, regardless of the choice of i, j, they yield the same solution:

$$\bar{q} = \frac{1}{\|\mathbf{u}_i + \mathbf{v}_i\|} \begin{bmatrix} \mathbf{u}_i + \mathbf{v}_i \\ 0 \end{bmatrix} = \frac{1}{\|\mathbf{u}_j + \mathbf{v}_j\|} \begin{bmatrix} \mathbf{u}_j + \mathbf{v}_j \\ 0 \end{bmatrix} \tag{37}$$

A.2 Motion estimation from 2 views

In this section, we describe an efficient Gauss-Newton method to determine the 5 dof transformation between 2 views given the bearing measurements to common features.

A.2.1 Problem formulation

Consider n features common to the images I_1, I_2, where each feature f_i's 3D position is expressed w.r.t frames $\{I_1\}$ and $\{I_2\}$ as ${}^{I_1}\mathbf{p}_{f_i}$ and ${}^{I_2}\mathbf{p}_{f_i}$, respectively. The measurement model in frame $\{I_j\}$ $(j = 1, 2)$ is the 2D projection of each feature f_i $(i = 1, \ldots, n)$

with additive zero-mean Gaussian noise:

$$ {}^{j}\mathbf{z}_i = \Pi({}^{I_j}\mathbf{p}_{f_i}) + {}^{j}\mathbf{n}_i \tag{38} $$

where $\Pi(\begin{bmatrix} x & y & z \end{bmatrix}^T) = \begin{bmatrix} \frac{x}{z} & \frac{y}{z} \end{bmatrix}^T$, and ${}^{j}\mathbf{n}_i \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. Next, we remove the unobservable scale (in this case the distance ${}^{I_2}d_{I_1}$ between the two images) by employing the following geometric constraint:

$$ \begin{aligned} {}^{I_2}\mathbf{p}_{f_i} &= \mathbf{R}(\bar{q})\,{}^{I_1}\mathbf{p}_{f_i} + {}^{I_2}\mathbf{p}_{I_1} \\ &= \mathbf{R}(\bar{q})\,{}^{I_1}\mathbf{p}_{f_i} + {}^{I_2}d_{I_1}\mathbf{t} \\ \Rightarrow \frac{1}{{}^{I_2}d_{I_1}}\,{}^{I_2}\mathbf{p}_{f_i} &= \mathbf{R}(\bar{q})\left(\frac{1}{{}^{I_2}d_{I_1}}\,{}^{I_1}\mathbf{p}_{f_i}\right) + \mathbf{t} \\ \Rightarrow {}^{I_2}\mathbf{f}_i &= \mathbf{R}(\bar{q})\,{}^{I_1}\mathbf{f}_i + \mathbf{t} \end{aligned} \tag{39} $$

where $\bar{q}$ and $\mathbf{t}$ are quaternion of rotation and unit vector of translational direction, respectively, from $\{I_1\}$ to $\{I_2\}$, while we have defined:

$$ {}^{I_2}\mathbf{f}_i \triangleq \frac{1}{{}^{I_2}d_{I_1}}\,{}^{I_2}\mathbf{p}_{f_i}, \text{ and } {}^{I_1}\mathbf{f}_i \triangleq \frac{1}{{}^{I_2}d_{I_1}}\,{}^{I_1}\mathbf{p}_{f_i} $$

Equation (39) describes the 5 dof constraint between the feature i's scaled 3D positions in 2 views.

Next, we seek to find the optimal solution $\mathbf{y} = \begin{bmatrix} \bar{q}^T & {}^{I_2}\mathbf{p}_{I_1}^T & {}^{I_1}\mathbf{p}_{f_1}^T \cdots {}^{I_1}\mathbf{p}_{f_n}^T \end{bmatrix}^T$, that minimizes the total reprojection error:

$$ \begin{aligned} \mathbb{C}(\mathbf{y}) &= \sum_{i=1}^{n} \left(\left\| {}^{1}\mathbf{z}_i - \Pi({}^{I_1}\mathbf{p}_{f_i}) \right\|^2 + \left\| {}^{2}\mathbf{z}_i - \Pi({}^{I_2}\mathbf{p}_{f_i}) \right\|^2 \right) \\ &= \sum_{i=1}^{n} \Bigg(\left\| {}^{1}\mathbf{z}_i - \Pi({}^{I_1}\mathbf{p}_{f_i}) \right\|^2 + \\ &\qquad \left\| {}^{2}\mathbf{z}_i - \Pi(\mathbf{R}(\bar{q})\,{}^{I_1}\mathbf{p}_{f_i} + {}^{I_2}d_{I_1}\mathbf{t}) \right\|^2 \Bigg) \\ &= \sum_{i=1}^{n} \Bigg(\left\| {}^{1}\mathbf{z}_i - \Pi(\frac{1}{{}^{I_2}d_{I_1}}\,{}^{I_1}\mathbf{p}_{f_i}) \right\|^2 + \\ &\qquad \left\| {}^{2}\mathbf{z}_i - \Pi\left(\mathbf{R}(\bar{q})(\frac{1}{{}^{I_2}d_{I_1}}\,{}^{I_1}\mathbf{p}_{f_i}) + \mathbf{t}\right) \right\|^2 \Bigg) \qquad (40) \\ &= \sum_{i=1}^{n} \Bigg(\left\| {}^{1}\mathbf{z}_i - \Pi({}^{I_1}\mathbf{f}_i) \right\|^2 + \\ &\qquad \left\| {}^{2}\mathbf{z}_i - \Pi\left(\mathbf{R}(\bar{q})\,{}^{I_1}\mathbf{f}_i + \mathbf{t}\right) \right\|^2 \Bigg) \qquad (41) \end{aligned} $$

where (40) is obtained by noting that the perspective projection is scale-invariant (i.e., $\Pi(\lambda\mathbf{x}) = \Pi(\mathbf{x})$, $\forall\lambda \neq 0$), and (41) results from the definition of ${}^{I_1}\mathbf{f}_i$ and (39).

Denoting $\mathbf{x} = \begin{bmatrix} \bar{q}^T & \mathbf{t}^T & {}^{I_1}\mathbf{f}_1^T \cdots {}^{I_1}\mathbf{f}_n^T \end{bmatrix}^T$, we have:

$$ \begin{aligned} &\mathbf{y}^* = \text{argmin}\ \mathbb{C}(\mathbf{y}),\ \text{subject to}\ \|\bar{q}\| = 1 \\ \Leftrightarrow\ &\mathbf{x}^* = \text{argmin}\ \mathbb{C}(\mathbf{x}),\ \text{subject to}\ \|\bar{q}\| = 1, \|\mathbf{t}\| = 1 \end{aligned} \tag{42} $$

To solve the non-linear least square (LS) problem (42), we employ iterative Gauss-Newton minimization: At every iteration k, $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} \oplus \delta\mathbf{x}^{(\mathbf{k})}$, where the update operation $\oplus$ is defined in Appendix A.2.2.

A.2.2 Solution In what follows, we first describe the pertubation model $\mathbf{x} = \hat{\mathbf{x}} \oplus \delta\mathbf{x}$ that we use, where $\mathbf{x}$ is the true state vector, $\hat{\mathbf{x}} = \begin{bmatrix} \hat{\bar{q}}^T & \hat{\mathbf{t}}^T & {}^{I_1}\hat{\mathbf{f}}_1^T \cdots {}^{I_1}\hat{\mathbf{f}}_n^T \end{bmatrix}^T$ is the estimate, and $\delta\mathbf{x}$ is the state pertubation corresponds to the estimate $\hat{\mathbf{x}}$ and their pertubation models. Recalling that the state vector comprises 3 main elements (orientation expressed as a unit quaternion, unit vector of translation, and scaled feature positions w.r.t $\{I_1\}$), we define the pertubation model for each state quantity as follows:

Unit-quaternion pertubation:

$$ \bar{q} = \delta\bar{q} \otimes \hat{\bar{q}} \tag{43} $$

$$ \Rightarrow\ \mathbf{R}(\bar{q}) \simeq (\mathbf{I} - \lfloor \delta\boldsymbol{\theta}\, \times \rfloor)\mathbf{R}(\hat{\bar{q}}) \tag{44} $$

with $\delta\bar{q} = (1 + \frac{\|\delta\boldsymbol{\theta}\|^2}{4})^{-\frac{1}{2}} \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta}^T & 1 \end{bmatrix}^T$, $\delta\boldsymbol{\theta}$ is the small angle-axis pertubation (i.e., $\|\delta\boldsymbol{\theta}\| \simeq 0$), where the last equation is obtained using the small-angle approximation [see Trawny and Roumeliotis (2005) for details].

Unit-vector of translation pertubation:

$$ \mathbf{t} = \mathbf{R}(\hat{\mathbf{t}}^{\perp}, \alpha)\mathbf{R}(\hat{\mathbf{t}}^{\perp\perp}, \beta)\hat{\mathbf{t}} \tag{45} $$

where $\begin{bmatrix} \hat{\mathbf{t}} & \hat{\mathbf{t}}^{\perp} & \hat{\mathbf{t}}^{\perp\perp} \end{bmatrix}$ forms a rotation matrix, and α, β are small pertubation angles. Using the small-angle approximation, in (45) we obtain:

$$ \begin{aligned} \mathbf{t} &\simeq (\mathbf{I} - \alpha\lfloor \hat{\mathbf{t}}^{\perp}\, \times \rfloor)(\mathbf{I} - \beta\lfloor \hat{\mathbf{t}}^{\perp\perp}\, \times \rfloor)\hat{\mathbf{t}} \\ &\simeq (\mathbf{I} - \alpha\lfloor \hat{\mathbf{t}}^{\perp}\, \times \rfloor - \beta\lfloor \hat{\mathbf{t}}^{\perp\perp}\, \times \rfloor)\hat{\mathbf{t}} \qquad (46) \\ &\simeq \hat{\mathbf{t}} + \alpha\hat{\mathbf{t}}^{\perp\perp} - \beta\hat{\mathbf{t}}^{\perp} \qquad (47) \end{aligned} $$

where in (46) we dropped the second-order term $\alpha\beta\lfloor \hat{\mathbf{t}}^{\perp}\, \times \rfloor \lfloor \hat{\mathbf{t}}^{\perp\perp}\, \times \rfloor$, while (47) is obtained by using the right-hand rule for the cross-product of $\hat{\mathbf{t}}$, $\hat{\mathbf{t}}^{\perp}$, and $\hat{\mathbf{t}}^{\perp\perp}$.

Feature position pertubation:

$$ {}^{I_1}\mathbf{f}_i = {}^{I_1}\hat{\mathbf{f}}_i + \delta\mathbf{f}_i \tag{48} $$

which is simply an additive model (no constraint is imposed on this element of the state vector).

Measurement function: Based on the pertubation models in (44), (47), and (48), we define the following pertubation state vector:

$$ \delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{r}^T & \delta\mathbf{f}_1^T \cdots \delta\mathbf{f}_n^T \end{bmatrix}^T = \begin{bmatrix} \delta\boldsymbol{\theta}^T & \alpha & \beta & \delta\mathbf{f}_1^T \cdots \delta\mathbf{f}_n^T \end{bmatrix}^T \tag{49} $$

Next, we linearize the measurement of feature i in the first image I_1, using Taylor series expansion around the estimate $\hat{\mathbf{x}}^{(k)}$ at iteration k to obtain:

$$ \begin{aligned} {}^{1}\mathbf{z}_i &= \Pi({}^{I_1}\mathbf{f}_i) + {}^{1}\mathbf{n}_i \\ &\simeq \Pi({}^{I_1}\hat{\mathbf{f}}_i^{(k)}) + \left(\frac{\partial\Pi}{\partial\,{}^{I_1}\mathbf{f}_i}({}^{I_1}\hat{\mathbf{f}}_i^{(k)})\right)\delta\mathbf{f}_i^{(k)} + {}^{1}\mathbf{n}_i \\ &\simeq \Pi({}^{I_1}\hat{\mathbf{f}}_i^{(k)}) + \mathbf{H}_{1i}^{(k)}\delta\mathbf{f}_i^{(k)} + {}^{1}\mathbf{n}_i \end{aligned} \tag{50} $$

where

$$ \mathbf{H}_{1i}^{(k)} = \frac{\partial\Pi}{\partial\,{}^{I_1}\mathbf{f}_i}(\begin{bmatrix} x & y & z \end{bmatrix}^T) = \begin{bmatrix} \frac{1}{z} & 0 & \frac{-x}{z^2} \\ 0 & \frac{1}{z} & \frac{-y}{z^2} \end{bmatrix} $$

Subsequently, from the geometric constraint (39), and the pertubation models (44), (47), and (48), we find the pertubation for the feature's position in $\{I_2\}$, denoted as

$\delta^{I_2}\mathbf{f}_i$, as follows:

$$\begin{aligned}
&{}^{I_2}\mathbf{f}_i = \mathbf{R}(\bar{q})^{I_1}\mathbf{f}_i + \mathbf{t} \\
\Rightarrow &{}^{I_2}\mathbf{f}_i \simeq (\mathbf{I} - \lfloor \delta\boldsymbol{\theta}_\times \rfloor)\mathbf{R}(\hat{\bar{q}})({}^{I_1}\hat{\mathbf{f}}_i + \delta\mathbf{f}_i) \\
&\quad + \hat{\mathbf{t}} + \alpha\hat{\mathbf{t}}^{\perp\perp} - \beta\hat{\mathbf{t}}^{\perp} \\
\Rightarrow &{}^{I_2}\mathbf{f}_i - \mathbf{R}(\hat{\bar{q}})^{I_1}\hat{\mathbf{f}}_i - \hat{\mathbf{t}} \simeq -\lfloor \delta\boldsymbol{\theta}_\times \rfloor \mathbf{R}(\hat{\bar{q}})^{I_1}\hat{\mathbf{f}}_i + \mathbf{R}(\hat{\bar{q}})\delta\mathbf{f}_i \\
&\quad + \alpha\hat{\mathbf{t}}^{\perp\perp} - \beta\hat{\mathbf{t}}^{\perp} \\
\Rightarrow &\delta^{I_2}\mathbf{f}_i \simeq \mathbf{R}(\hat{\bar{q}})\delta\mathbf{f}_i + \begin{bmatrix} \lfloor \mathbf{R}(\hat{\bar{q}})^{I_1}\hat{\mathbf{f}}_{i\times} \rfloor & \hat{\mathbf{t}}^{\perp\perp} & -\hat{\mathbf{t}}^{\perp} \end{bmatrix} \delta\mathbf{r}
\end{aligned} \tag{51}$$

where to reach (51), we have defined:

$$\delta^{I_2}\mathbf{f}_i = {}^{I_2}\mathbf{f}_i - {}^{I_2}\hat{\mathbf{f}}_i = {}^{I_2}\mathbf{f}_i - (\mathbf{R}(\hat{\bar{q}})^{I_1}\hat{\mathbf{f}}_i + \hat{\mathbf{t}})$$

Based on (50) and (51), we linearize the measurement of feature i in the second image I_2 as follows:

$$\begin{aligned}
{}^2\mathbf{z}_i &\simeq \Pi({}^{I_2}\hat{\mathbf{f}}_i^{(k)}) + \mathbf{H}_{2i}^{(k)}\delta^{I_2}\mathbf{f}_i^{(k)} + {}^2\mathbf{n}_i \\
&\simeq \Pi({}^{I_2}\hat{\mathbf{f}}_i^{(k)}) + \mathbf{H}_{2i}^{(k)}\mathbf{R}(\hat{\bar{q}}^{(k)})\delta\mathbf{f}_i^{(k)} \\
&\quad + \mathbf{H}_{2i}^{(k)}\begin{bmatrix} \lfloor \mathbf{R}(\hat{\bar{q}}^{(k)})^{I_1}\hat{\mathbf{f}}_i^{(k)}{}_\times \rfloor & \hat{\mathbf{t}}^{(\mathbf{k})\perp\perp} & -\hat{\mathbf{t}}^{(\mathbf{k})\perp} \end{bmatrix}\delta\mathbf{r}^{(k)} \\
&\quad {}^2\mathbf{n}_i \\
&\simeq \Pi({}^{I_2}\hat{\mathbf{f}}_i^{(k)}) + \mathbf{J}_{fi}^{(k)}\delta\mathbf{f}_i^{(k)} + \mathbf{J}_{ri}^{(k)}\delta\mathbf{r}^{(k)} + {}^2\mathbf{n}_i
\end{aligned} \tag{52}$$

where in (52) we have defined:

$$\begin{aligned}
\mathbf{H}_{2i}^{(k)} &= \frac{\partial\Pi}{\partial {}^{I_1}\mathbf{f}_i}({}^{I_1}\hat{\mathbf{f}}_i^{(k)}) \\
\mathbf{J}_{fi}^{(k)} &= \mathbf{H}_{2i}^{(k)}\mathbf{R}(\hat{\bar{q}}^{(k)}) \\
\mathbf{J}_{ri}^{(k)} &= \mathbf{H}_{2i}^{(k)}\begin{bmatrix} \lfloor \mathbf{R}(\hat{\bar{q}}^{(k)})^{I_1}\hat{\mathbf{f}}_i^{(k)}{}_\times \rfloor & \hat{\mathbf{t}}^{(\mathbf{k})\perp\perp} & -\hat{\mathbf{t}}^{(\mathbf{k})\perp} \end{bmatrix}
\end{aligned}$$

Employing the linearizations of (50) and (52), we transform the non-linear LS problem (42) into a linear LS problem. Specifically, at each iteration k, we seek to find the $\delta\mathbf{x}^{(k)}$ that minimizes the following LS cost function:

$$\begin{aligned}
\mathbb{C}'(\delta\mathbf{x}^{(k)}) &= \sum_{i=1}^{n}\Bigg(\left\|{}^1\mathbf{z}_i - \Pi({}^{I_1}\hat{\mathbf{f}}_i^{(k)}) - \mathbf{H}_{1i}^{(k)}\delta\mathbf{f}_i^{(k)})\right\|^2 + \\
&\qquad \left\|{}^2\mathbf{z}_i - \Pi({}^{I_2}\hat{\mathbf{f}}_i^{(k)}) - \mathbf{J}_{fi}^{(k)}\delta\mathbf{f}_i^{(k)} - \mathbf{J}_{pi}^{(k)}\delta\mathbf{r}^{(k)}\right\|^2\Bigg) \\
&= \sum_{i=1}^{n}\left(\left\|\begin{bmatrix}\delta^1\mathbf{z}_i \\ \delta^2\mathbf{z}_i\end{bmatrix} - \begin{bmatrix}\mathbf{H}_{1i}^{(k)} & \mathbf{0} \\ \mathbf{J}_{fi}^{(k)} & \mathbf{J}_{ri}^{(k)}\end{bmatrix}\begin{bmatrix}\delta\mathbf{f}_i^{(k)} \\ \delta\mathbf{r}^{(k)}\end{bmatrix}\right\|^2\right)
\end{aligned} \tag{53}$$

where $\delta^j\mathbf{z}_i = {}^j\mathbf{z}_i - \Pi({}^{I_j}\hat{\mathbf{f}}_i^{(k)})$.

Computing $\delta\mathbf{x}^{(k)} = \operatorname{argmin}\mathbb{C}'(\delta\mathbf{x}^{(k)})$ is equivalent to finding the LS solution of the following over-determined ($n > 5$) linear system of equations:

$$\begin{bmatrix}\delta^1\mathbf{z}_1 \\ \delta^2\mathbf{z}_1 \\ \delta^1\mathbf{z}_2 \\ \delta^2\mathbf{z}_2 \\ \vdots \\ \delta^1\mathbf{z}_n \\ \delta^2\mathbf{z}_n\end{bmatrix} = \begin{bmatrix}\mathbf{H}_{11}^{(k)} & \mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_{f1}^{(k)} & \mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{J}_{r1}^{(k)} \\ \mathbf{0} & \mathbf{H}_{12}^{(k)} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{f2}^{(k)} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{J}_{r2}^{(k)} \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ \mathbf{0} & \cdots & \cdots & \cdots & \mathbf{H}_{1n}^{(k)} & \mathbf{0} \\ \mathbf{0} & \cdots & \cdots & \cdots & \mathbf{J}_{fn}^{(k)} & \mathbf{J}_{rn}^{(k)}\end{bmatrix}\begin{bmatrix}\delta\mathbf{f}_1^{(k)} \\ \delta\mathbf{f}_2^{(k)} \\ \vdots \\ \delta\mathbf{f}_n^{(k)} \\ \delta\mathbf{r}^{(k)}\end{bmatrix} \tag{54}$$

To efficiently solve (54), we take advantage of its sparse structure as described in the following steps:

Step 1: Solve for $\delta\mathbf{r}^{(k)}$ We will first eliminate all the terms $\delta\mathbf{f}_i^{(k)}$, $i = 1, \ldots, n$. To do so, we define the following matrix:

$$\mathbf{U} = \begin{bmatrix}\mathbf{u}_1^{T(k)} & \mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{u}_2^{T(k)} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ \mathbf{0} & \cdots & \cdots & \cdots & \mathbf{0} & \mathbf{u}_n^{T(k)}\end{bmatrix} \tag{55}$$

whose each 1×4 block element is defined as:

$$\mathbf{u}_i^{(k)} = \begin{bmatrix}\mathbf{u}_{i(1:2)}^{(k)} \\ \mathbf{u}_{i(3:4)}^{(k)}\end{bmatrix} = \begin{bmatrix} z_i\mathbf{J}_{fi(1:2,1:2)}^{T(k)}\begin{bmatrix}\mathbf{J}_{fi(2,:)}^{(k)}{}^{I_1}\hat{\mathbf{f}}_i \\ -\mathbf{J}_{fi(1,:)}^{(k)}{}^{I_1}\hat{\mathbf{f}}_i\end{bmatrix} \\ -\mathbf{J}_{fi(2,:)}^{(k)}{}^{I_1}\hat{\mathbf{f}}_i \\ \mathbf{J}_{fi(1,:)}^{(k)}{}^{I_1}\hat{\mathbf{f}}_i \end{bmatrix} \tag{56}$$

Hence, each $\mathbf{u}_i^T$ has the following property:

$$\mathbf{u}_i^{T(k)}\begin{bmatrix}\mathbf{H}_{1i}^{(k)} \\ \mathbf{J}_{fi}^{(k)}\end{bmatrix} = \mathbf{0}^T \Leftrightarrow \begin{bmatrix}\frac{1}{z_i} & 0 \\ 0 & \frac{1}{z_i} \\ \frac{-x_i}{z_i^2} & \frac{-y_i}{z_i^2}\end{bmatrix}\mathbf{J}_{fi}^{T}{}^{(k)}\begin{bmatrix}\mathbf{u}_{i(1:2)}^{(k)} \\ \mathbf{u}_{i(3:4)}^{(k)}\end{bmatrix} = \mathbf{0} \tag{57}$$

where ${}^{I_1}\hat{\mathbf{f}}_i^{(k)} = \begin{bmatrix}x_i & y_i & z_i\end{bmatrix}^T$. Multiplying both sides of (54) with $\mathbf{U}$ yields:

$$\begin{bmatrix}\mathbf{u}_{1(1:2)}^{T(k)}\delta^1\mathbf{z}_1 - \mathbf{u}_{1(3:4)}^{T(k)}\delta^2\mathbf{z}_1 \\ \mathbf{u}_{2(1:2)}^{T(k)}\delta^1\mathbf{z}_2 - \mathbf{u}_{2(3:4)}^{T(k)}\delta^2\mathbf{z}_2 \\ \vdots \\ \mathbf{u}_{n(1:2)}^{T(k)}\delta^1\mathbf{z}_n - \mathbf{u}_{n(3:4)}^{T(k)}\delta^2\mathbf{z}_n\end{bmatrix} = \begin{bmatrix}\mathbf{u}_{1(3:4)}^{T(k)}\mathbf{J}_{r1}^{(k)} \\ \mathbf{u}_{2(3:4)}^{T(k)}\mathbf{J}_{r2}^{(k)} \\ \vdots \\ \mathbf{u}_{n(3:4)}^{T(k)}\mathbf{J}_{rn}^{(k)}\end{bmatrix}\delta\mathbf{r}^{(k)} \tag{58}$$

Note that through this analytical process ($\mathbf{U}$ is computed in closed-form), we reduced the dimension of the problem we need to solve from $4n \times (3n + 5)$ for (54) to $n \times 5$ for (58), which is very efficient to solve for $\delta\mathbf{r}^{(k)}$.

Step 2: Solve for each $\delta\mathbf{f}_i^{(k)}$ Given $\delta\mathbf{r}^{(k)}$, we can separately solve for each $\delta\mathbf{f}_i^{(k)}$ by employing the following 4×3 system of equations resulting from the two block rows of (54) corresponding to each feature i:

$$\begin{bmatrix}\delta^1\mathbf{z}_i \\ \delta^2\mathbf{z}_i - \mathbf{J}_{fi}^{(k)}\delta\mathbf{r}^{(k)}\end{bmatrix} = \begin{bmatrix}\mathbf{H}_{1i}^{(k)} \\ \mathbf{J}_{fi}^{(k)}\end{bmatrix}\delta\mathbf{f}_i^{(k)} \tag{59}$$

Again, we can easily take advantage of the structure of (59) and employ in place Givens rotations (Golub and Van-Loan 2012) to transform it to an upper triangular system of size 3×3 and solve it efficiently.

After obtaining $\delta\mathbf{x}^{(k)}$, we employ (43), (45), and (48) to update $\hat{\mathbf{x}}^{(k+1)} = \hat{\mathbf{x}}^{(k)} \oplus \delta\mathbf{x}^{(k)}$ while ensuring unity of the quaternion and the translational vector; then, we repeat this process till convergence.

B Nomenclature

I_l	Image number l in the visual graph
I_{s_j}	Image set point j from the user
I_k^r	Reference image k along the visual path
$\{I_k^r\}$	Camera frame when image I_k^r is acquired
I_t	Current image taken at time instant t
$\{I_t\}$	Camera frame when image I_t is acquired
${}^{I_k^r}\mathbf{p}_{f_i}$	Position of feature f_i in frame $\{I_k^r\}$
${}^{I_k^r}\mathbf{b}_{f_i}$	Unit bearing vector of feature f_i in frame $\{I_k^r\}$
${}^{I_t}\mathbf{p}_{f_i}$	Position of feature f_i in frame $\{I_t\}$
${}^{I_t}\mathbf{b}_{f_i}$	Unit bearing vector of feature f_i in frame $\{I_t\}$
${}^{I_k^r}\mathbf{p}_{I_t}$	Position of frame $\{I_t\}$ in frame $\{I_k^r\}$
${}^{I_k^r}\mathbf{t}_{I_t}$	Unit translational vector of frame $\{I_t\}$ in frame $\{I_k^r\}$
${}^{I_k^r}_{I_t}\mathbf{R}$	Rotational matrix describing the orientation of $\{I_t\}$ in frame $\{I_k^r\}$

References

Alahi, A., Ortiz, R., and Vandergheynst, P. (2012). "FREAK: Fast retina keypoint". In *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 510–517, Providence, RI.

Azrad, S., Kendoul, F., and Nonami, K. (2010). "Visual servoing of quadrotor micro-air vehicle using color-based tracking algorithm". *Journal of System Design and Dynamics*, 4(2):255–268.

Bills, C., Chen, J., and Saxena, A. (2011). "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.

Bourquardez, O., Mahony, R., Guenard, N., and Chaumette, F. (2009). "Image-based visual servo control of the translation kinematics of a quadrotor aerial vehicle". *IEEE Transactions on Robotics*, 25(3):743–749.

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

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

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

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

Courbon, J., Mezouar, Y., Guenard, N., and Martinet, P. (2010). "Vision-based navigation of unmanned aerial vehicle". *Control Engineering Practice*, 18(7):789–799.

Courbon, J., Mezouar, Y., and Martinet, P. (2008). "Indoor navigation of a non-holonomic mobile robot using a visual memory". *Autonomous Robots*, 25(3):253–266.

Diosi, A., Šegvić, S., Remazeilles, A., and Chaumette, F. (2011). "Experimental evaluation of autonomous driving based on visual memory and image-based visual servoing ". *IEEE Transactions on Robotics*, 12(3):870–883.

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

Do, T., Carrillo-Arce, L. C., and Roumeliotis, S. I. (2016). "Autonomous flights through image-defined paths (videos)". http://mars.cs.umn.edu/research/quadrotor_project.php.

Fischler, M. and Bolles, R. (1981). "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". *Communications of the ACM*, 24(6):381–395.

Franklin, G. F., Powell, J. D., and Workman, M. L. (1997). *Digital control of dynamic systems*. Addison-Wesley, Reading, MA.

Goedemé, T., Tuytelaars, T., Gool, L. V., Vanacker, G., and Nuttin, M. (2005). "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.

Golub, G. and Van-Loan, C. (2012). *Matrix Computations*. JHU Press.

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

Honegger, D., Meier, L., Tanskanen, P., and Pollefeys, M. (2013). "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.

Horn, B. (1987). "Closed-form solution of absolute orientation using unit quaternions". *Journal of the Optical Society of America A*, 4(4):629–642.

Lee, D., Ryan, T., and Kim, H. J. (2012). "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.

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

Nguyen, T., Mann, G. K. I., and Gosine, R. G. (2014). "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.

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

Nistér, D. and Stewénius, H. (2006). "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.

Parrot-Inc. Bebop Drone. http://www.parrot.com/products/bebop-drone/.

Reynolds, R. G. (1998). "Quaternion parameterization and a simple algorithm for global attitude estimation". *Journal of Guidance, Control, and Dynamics*, 21(4):669–671.

Trawny, N. and Roumeliotis, S. I. (2005). "Indirect Kalman filter for 3d attitude estimation". https://www-users.cs.umn.edu/~trawny/Publications/Quaternions_3D.