Robotics and Autonomous Systems 62 (2014) 1657–1667

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Vistas and parallel tracking and mapping with Wall–Floor Features: Enabling autonomous flight in man-made environments

D.-N. Ta *, K. Ok, F. Dellaert

Center for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA, USA

HIGHLIGHTS

- We enable autonomous flight for a lightweight quad-rotor in indoor environments.
- Distant features, called vistas, are used for steering towards open spaces.
- Parallel tracking and mapping framework with odometry overcomes challenges in monoSLAM.
- Special Wall–Floor Features cope with feature-poor environments.
- Wall inference using Wall–Floor Features helps avoid lateral collisions.

ARTICLE INFO

Article history:
Available online 28 March 2014

Keywords:
Quadrotor
MAV
Autonomous navigation
Indoor
SLAM
Monocular
Parallel tracking and mapping
Vistas
Wall–Floor Features

ABSTRACT

We propose a solution towards the problem of autonomous flight in man-made indoor environments with a micro aerial vehicle (MAV), using a frontal camera, a downward-facing sonar, and odometry inputs. While steering an MAV towards distant features that we call *vistas*, we build a map of the environment in a parallel tracking and mapping fashion to infer the wall structure and avoid lateral collisions in real-time. Our framework overcomes the limitations of traditional monocular SLAM approaches that are prone to failure when operating in feature-poor environments and when the camera purely rotates. First, we overcome the common dependency on feature-rich environments by detecting *Wall–Floor Features (WFFs)*, a novel type of low-dimensional landmarks that are specifically designed for man-made environments to capture the geometric structure of the scene. We show that WFFs not only reveal the structure of the scene, but can also be tracked reliably. Second, we cope with difficult robot motions and environments by fusing the visual data with odometry measurements in a principled manner. This allows the robot to continue tracking when it purely rotates and when it temporarily navigates across a completely featureless environment. We demonstrate our results on a small commercially available quad-rotor platform flying in a typical feature-poor indoor environment.

Published by Elsevier B.V.

1. Introduction

We address the problem of vision-based autonomous navigation in man-made environments for Micro Aerial Vehicles (MAVs). We utilize a lightweight frontal camera, a downward-facing sonar for height measurements, and odometry inputs, while refraining from using heavy and power-hungry sensors that impose limitations on the MAVs. Research and applications for such lightweight MAVs have been growing significantly in recent years for both military and civilian services.

Man-made environments pose two key difficulties for vision-based methods. First, state-of-the-art monocular vision systems [1,2] rely on many corner-type features in the image to localize the camera and build a map. However, a typical indoor environment does not possess many distinct corner-type features, while the few that are present are mostly far-away, providing only little information to localize the camera reliably. Furthermore, these systems break down when the camera dominantly rotates towards an unknown region, due to the lack of motion parallax needed to triangulate new landmarks in the scene. Such rotation is common in robot navigation where the robot's frontal camera undergoes a yaw motion induced by a change in the robot's heading.

In this paper, we present a vision-based navigation system that steers the robot towards distant features, which we call *vistas*, while inferring the structure of the scene to avoid lateral collisions.

* Corresponding author. Tel.: +1 6174158387.
E-mail addresses: duynguyen@gatech.edu (D.-N. Ta), kyelok@gatech.edu (K. Ok), dellaert@cc.gatech.edu (F. Dellaert).

http://dx.doi.org/10.1016/j.robot.2014.03.010
0921-8890/Published by Elsevier B.V.

Such capability could not be achieved in previous vision-based MAVs, without dedicating additional sensors for this purpose (i.e. frontal and side-facing sonars [3]). We infer the scene structure by building a map of Wall–Floor Features, a special type of landmarks, that can cope with lack of corner-type features in man-made environments. Our map building system also overcomes the aforementioned limitations of monocular SLAM by fusing visual features with odometry inputs. We do this in a parallel tracking and mapping framework to achieve fast response to changes in the environment, while building a high-quality map of the scene. Contributions of our system, which is an extension of our previous work [4], are as follows (see Fig. 1):

Our first contribution is using vistas to determine the robot steering direction, enabling robust navigation. Our vistas are derived from first principles of what it means to be *distant*; hence, they are not hallway-specific like the previous work that depends on vanishing points detected from spurious edges [3] or hallway-specific cues [5]. Moreover, vistas are also derived from scale-space features and inherit the properties such that they are easily and reliably detected and tracked in many types of environments.

Our second contribution is a special type of landmarks, called *Wall–Floor Features*, that are suitable for mapping indoor environments and enabling autonomous exploration capabilities. In addition to vistas, for intelligent exploration schemes, the MAV needs some knowledge of the scene structure. We infer the structure from a map of compact and low-dimensional landmarks that are informative enough to capture the most important geometric information of the scene. Our novel landmarks lie on the perpendicular intersection of vertical lines on the wall and the horizontal floor plane. They encode the direction of the wall and can capture any type of corners whether straight, convex or concave.

Our third contribution is a parallel tracking and mapping framework that adopts the parallel-style execution of tracking and mapping introduced in the Parallel Tracking and Mapping (PTAM) system in [1], and extends its capabilities to deal with the lack of features in man-made environments and typical robot motions that fail monocular SLAM. While PTAM requires a large number of visual features to localize the camera, our system fuses visual features with odometry/IMU measurements to deal with periods of featureless scenes in the environment. Utilizing sensor fusion, our system can also deal with rotational motions towards unknown regions, where map building commonly fails due to the lack of motion parallax to triangulate new landmarks for the map. Unlike other PTAM–IMU fusion work that uses the actual PTAM system as a black box [6,7] and fails to prevent breakdowns for such cases, we incorporate the parallel-style execution in our own factor graph framework [8] to fuse sensor information in a principled manner; hence, our system gains robustness in these difficult situations.

We demonstrate our results on an inexpensive AR.Drone 2.0[1] quad-rotor. Our system successfully flies autonomously towards vistas while avoiding lateral collisions with wall inference. We evaluate our mapping results with loop closures on a pre-captured offline dataset using the ground-truth map, and compare its accuracy against PTAM and the quad-rotor's own estimate provided by the manufacturer's own firmware.

2. Related work

Successful MAV navigation systems neglect to address the power and payload limitations by using heavy and power-hungry sensors. For example, [9–14] use laser scanners to build a map of indoor environments for MAV navigation and exploration. Scherer et al. [15] use a laser scanner on a helicopter to detect and avoid different types of objects such as buildings, trees, and 6 mm wires in the city. Recently, Bry et al. [16] also use a laser scanner and a prebuilt map to enable aggressive flight and obstacle avoidance on a small fixed-wing airplane. These systems, however, are severely limited to short-term operations due to their heavy payload and high power usage. Furthermore, active sensors such as laser scanners are undesirable in many applications (e.g., military), due to the risk of cross-talk and ineligibility for covert operations. Therefore, we preclude the use of laser scanners and other heavy and power-hungry sensors.

Fig. 1. We use *vistas* (bottom left) to steer the robot and rely on a map of *Wall–Floor Features* (bottom right) in a parallel tracking and mapping framework to infer the scene structure and avoid lateral collisions. We present our results in an indoor setting using an AR.Drone (top).

Recent work in vision-based autonomous navigation neglects to provide exploration and planning capabilities achieved by building a map of the environment. While the system in [17] can navigate autonomously with a single camera, it relies on a prebuilt map. Bills et al. [3] enable autonomous navigation towards the end of the hallway by detecting the vanishing point from intersections of long lines along the corridors, but their system relies on supplementary sonar sensors to detect openings to the sides for autonomous exploration. Moreover, Murali and Birchfield [5] enable vision-based autonomous exploration on a ground robot by fusing many hallway-specific properties such as high entropy, symmetry, self-similarity, etc. to detect new hallway directions; however, their method is reactive and cannot support high-level planning algorithms. Based on map-building, our method can support exploration and planning algorithms to efficiently navigate towards unexplored regions.

On the other hand, state-of-the-art vision-based map-building methods are insufficient for usage in indoor navigation. Many methods build 3D point-cloud maps [18,7,19,20] but in textureless indoor environments, the point-clouds are too sparse to reveal the 3D structure needed for path/motion planning. Although some [21,22] build a map from the edges in the environment, they neglect to infer the environment structure crucial for robot navigation. Moreover, state-of-the-art vision-based methods that reconstruct the indoor scene [23–25] either rely on the indoor Manhattan world assumption or require expensive multi-hypothesis inference methods [24,25]. Our method based on

[1] http://ardrone.parrot.com.

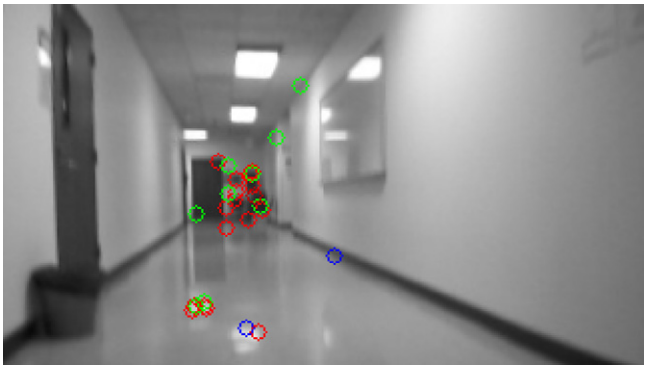


Fig. 2. PTAM is not suitable for robots with a frontal camera, especially in this type of indoor environment with a small number of features that are mostly far away along the robot's direction. Our framework for robot navigation overcomes PTAM's limitations by fusing visual and odometry measurements together in a principled manner.

Wall–Floor Features improves on previous work with the ability to work in textureless environments, using sparse yet informative scene representation, and not relying on the indoor Manhattan world assumption.

Furthermore, other vision methods that are derived from PTAM [1] to enable map building need to rely on a downward camera for building a map, lacking the ability to avoid obstacles [26,6,27,7,18,19]. When using with a frontal camera, these monocular SLAM methods fall short in indoor environments where sufficient features do not exist as shown in Fig. 2, and when the robot undergoes rotationally dominant motions. In these situations, one needs to rely on odometry or IMU measurements for the localization task, but the latest work in combining PTAM and odometry or IMU measurements does not integrate them properly. These systems [6,7] simply treat PTAM as a "black box" and combine results from PTAM with IMU measurements in an outer loop; hence, the visual and odometry/IMU measurements are not fused together in the same system, nor are the fused results propagated back to PTAM to prevent it from failing, e.g., when there is a lack of visual features in the scene, or when the robot rotates towards unknown regions.

3. Autonomous navigation towards vistas

One of the first tasks in autonomous navigation and exploration is to determine the direction towards open space. In this section, we derive from first principles a general approach that can potentially be applied to any type of environment to steer the robot.

3.1. Vista size change criterion

We use *vistas* to refer to landmarks that are far from the robot and can be used as a steering direction towards open space when exploring in an unknown environment.

One important property of vistas is that, due to their far distance to the robot, the size of their projection in the camera frame does not change significantly when flying towards them. This property is already well-known in perceptual psychology under the τ-theory [28] by David Lee, saying that the time-to-collision (TTC) to an object is the ratio τ of the object's image size to the rate of its size change. Some work has utilized this property to compute TTC using optical flow [29,30] or direct methods [31,5].

Using this property, we derive vistas from relative size change of scale-space features such as SIFT [32] or SURF [33]. The optimal sizes of these features are computed by fitting a 3D quadratic function to the feature responses in scale-space around the max response [32].

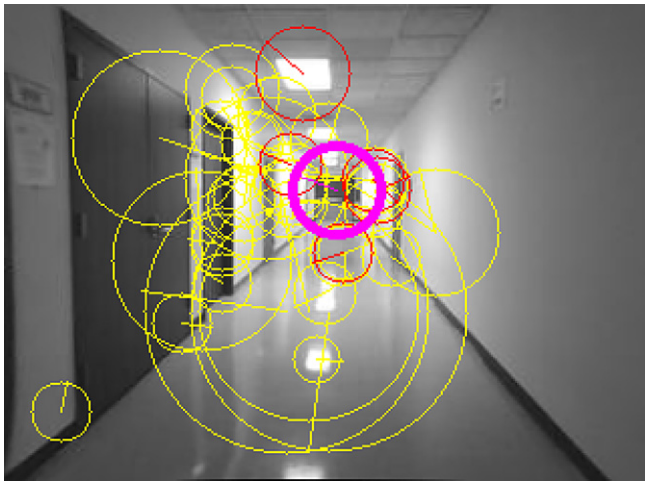


Fig. 3. Detected *vistas* (in red) and features that do not satisfy the *vista* criteria (in yellow) are shown. The closest*vista* to the mean of all the detected *vistas* (pink feature) is selected as the steering direction for the robot. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Let s_1, s_2 be feature sizes and Z_1, Z_2 be their distance from the camera at frames 1 and 2. Since $s_i = f\frac{S}{Z_i}$, where f is the camera focal length and S is the true size of landmark, we have $s_1/s_2 = Z_2/Z_1$. It can be easily shown that $\frac{\Delta s}{s_2} = -\frac{\Delta Z}{Z_1} = \frac{t_z}{Z_1}$ where $\Delta s = s_2 - s_1$ is the absolute size change of the feature and $t_z = -\Delta Z = Z_1 - Z_2$ is the amount of forward movement of the robot between two frames, easily obtained from integrating an IMU, using a motion model, or fusing optical flow and corner tracking on a bottom-looking camera, as already implemented on the AR.Drone [34].

Let $Z_{1\,\min}$ be the minimum safety distance to the landmark in camera frame 1 so that any landmarks with $Z_1 \geq Z_{1\,\min}$ can be considered vistas. The relative size change of vistas must satisfy

$$\frac{\Delta s}{s_2} \leq \frac{t_z}{Z_{1\,\min}}. \tag{1}$$

As shown in Fig. 3, this criterion leads to an efficient way to detect distant landmarks in the environment.

3.2. Vista rotation-predictability criterion

The minimum safety distance $Z_{1\,\min}$ of vistas in the previous section could be chosen arbitrarily as long as it is safe for the robot to avoid colliding with the wall at current speed. However, to ease the prediction and tracking of the vistas, we enforce another geometric property of distant landmarks that their projection in the image should be predictable using pure camera rotation, unaffected by the translation. We call this "rotation-predictability" criterion.

We derive this additional requirement for our vistas based on a well-known fact that if a point is at infinity, its projection in the camera image can be purely determined by the camera rotation. In our case, the camera translation between two consecutive frames is insignificant compared to the distance from the camera to the landmarks, hence has no effect on the landmark position in the image.

More specifically, let p_1 and p_2 be the 2D homogeneous forms of the landmark projections in camera frames 1 and 2. Also, let $K = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$ be the camera calibration matrix, and $X_2^1 = \{R, t\} \in \mathbb{SE}(3)$ be the odometry of the camera from frame 1 to frame 2. If the landmark P is at infinity or if the camera motion is under a pure

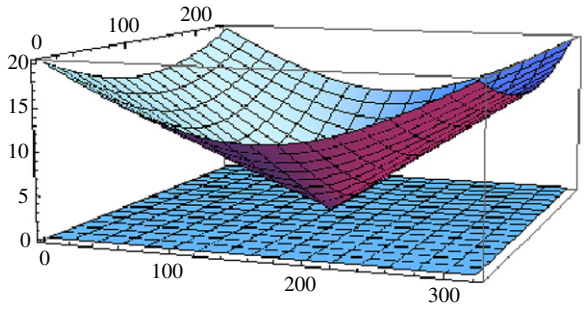


Fig. 4. Minimum $Z^r_{1\,\min}$ distances for rotation-predictable features for $t_x = t_y = 0,\ t_z = 0.1$. The horizontal xy-plane is in image pixels, and the vertical z-axis is the minimum Z_1 required at each pixel. Plot with camera calibration: $o_x = 160,\ o_y = 120,\ f_x = f_y = 210$.

rotation ($t = 0$), its projections p_1 and p_2 are related by the infinite homography $H = KR^2_1K^1$ between the two images [35]:

$$p_2 = p^r_2 \sim KR^2_1K^{-1}p_1,$$

where $R^2_1 = R^\top$, and $\sim$ denotes the equivalent up to a constant factor.

However, if the camera motion also involves a translation, i.e. $t \neq 0$, and the landmark is not at infinity, the relationship between p_1 and p_2 is:

$$\begin{aligned} p_2 = p^t_2 &\sim K(R^2_1Z_1K^{-1}p_1 + t^2_1) \\ &\sim p^r_2 + \frac{1}{Z_1}Kt^2_1, \end{aligned}$$

where $t^2_1 = -R^\top t$.

Consequently, the rotation-predictability criterion infers that p^t_2 must be well approximated by p^r_2. In this case, the effect of the camera translation t on p_2 is negligible and insensible by the camera, i.e., in homogeneous form, $\frac{1}{Z_1}Kt^2_1 \approx kp^r_2$, for some scalar $k \in \mathbb{R}$. To satisfy this constraint, we impose the condition that the non-homogeneous distance between p^t_2 and p^r_2 has to be less than 1 pixel, i.e.,

$$\left\| \frac{1}{z_{p^r_2}}p^r_2 - \frac{1}{z_{p^t_2}}p^t_2 \right\|^2 \leq 1,$$

where $z_{p^r_2}$ and $z_{p^t_2}$ are the third components of p^r_2 and p^t_2, respectively. Solving for this constraint leads to the minimum depth $Z^r_{1\,\min}$ of the landmark such that its image projection can be purely determined by the camera rotationas follows:

$$\begin{aligned} &Z^r_{1\,\min}(x, y, t) \\ &\quad = t_z + \sqrt{[f_xt_x + t_z(o_x - x)]^2 + [f_yt_y + t_z(o_y - y)]^2} \end{aligned} \tag{2}$$

where $t = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^\top$, and (x, y) is the non-homogeneous coordinate of p_1.

This formula shows that the minimum $Z^r_{1\,\min}$ distance of the landmark in the first camera view depends on its position (x, y) in the first image, and also the camera translation t. Fig. 4 shows the $Z^r_{1\,\min}$ required for each pixel landmark location in the image where the camera moves in z direction.

Note that at the Focus of Expansion (FoE), where the camera translation vector intersects with the camera image plane, the minimum $Z^r_{1\,\min}$ is very close to the camera. As a trivial example, when the camera moves forward without rotation, $R = I_{3\times3}$, the minimum distance for rotation-predictability criterion is $Z^r_{1\min} = t_z$; i.e., any point along the camera optical axis is not affected by the camera translation as long as it is in front of the second camera view.

Although such limitations exist in the FoE region, the rotation-predictability criterion is still useful to reject false vistas outside the region. Thus, we use $\max(Z^r_{1\,\min}, Z_{1\,\min})$ for the minimum distance in Eq. (1) to create the final criteria to track vistas on a frame to frame basis.

4. Parallel tracking and mapping framework for robot navigation

Despite the capability of vistas in steering a robot towards open space, vistas alone cannot grant fully autonomous exploration capabilities. In order to avoid lateral collisions, detect directions towards unexplored regions, and adopt an intelligent planning scheme, it is critical to obtain a map of the environment.

Our map-building and localization system tailors the well-known Parallel Mapping and Tracking (PTAM) system [1] for robot navigation. Although PTAM is originally designed for Augmented Reality applications, its parallel framework is desirable in robotics applications as well, in which the tracker guarantees a fast response to changes in the environment, while the mapper builds a high-quality map of the environment and performs computationally expensive tasks such as structure inference. These desirable characteristics of PTAM have proven the system superior over traditional monocular SLAM filtering methods in robotics applications [36].

However, similar to other feature-based monocular SLAM methods, PTAM suffers from pure camera rotations and the lack of features in the environment, as previously discussed. Our framework overcomes these limitations by fusing visual features with odometry inputs for robot navigation. We note that although we use a special type of visual features in this work to capture the structure of indoor environments, as detailed in Section 5, our framework presented here is generic enough to be applied for any types of visual features.

4.1. The framework

Fig. 5 shows the overall layout of our framework, consisting of a tracker and a mapper running in parallel. The *tracker* receives a stream of timestamped images and odometry measurements from the robot. Its task is to localize the current camera in real-time within the optimized map of local landmarks created and maintained by the mapper. Based on the localization results, it then decides if a new keyframe and odometry measurements should be added to the existing map to cover new parts of the scene. Performing these tasks in real-time, the tracker's localization results benefit time-critical tasks such as controlling the robot to follow a trajectory, or avoiding obstacles.

The *mapper* periodically receives a new keyframe image from the tracker, as well as odometry measurements between this new keyframe and the previously received keyframe. These odometry measurements are created by accumulating all odometry measurements of the intermediate frames between the two keyframes. The mapper's task then is to build a global map of visual landmarks in the environment, and simultaneously compute the optimal keyframe poses. After finishing an iteration, the mapper updates the tracker with the newly optimized local map and keyframe poses, which are then used by the tracker to localize the current camera pose in real-time. Due to the inherently slower pace of the mapper and the fact that it does not have to obey real-time constraints, other time-consuming tasks such as inferring the structure of the environment can also be added to the mapper thread, to provide contextual information of the environment.

4.2. The mapper

Unlike PTAM, which only uses visual measurements, we employ both visual measurements and odometry measurements between

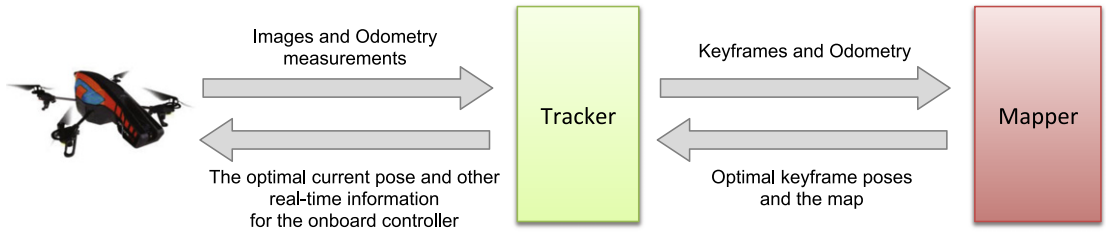


Fig. 5. Overall framework of our Parallel Tracking and Mapping system with odometry measurements.

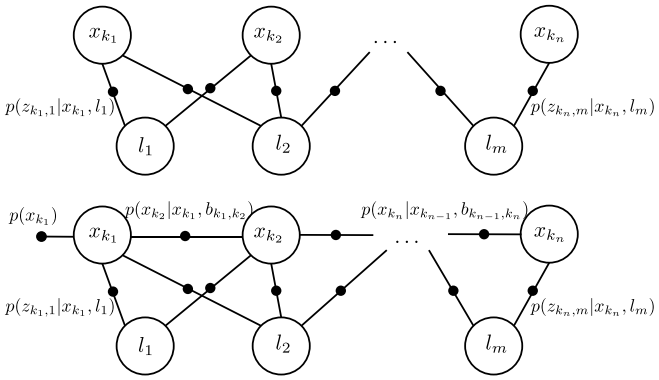


Fig. 6. The factor graphs of PTAM's mapper (top) and our mapper (bottom).

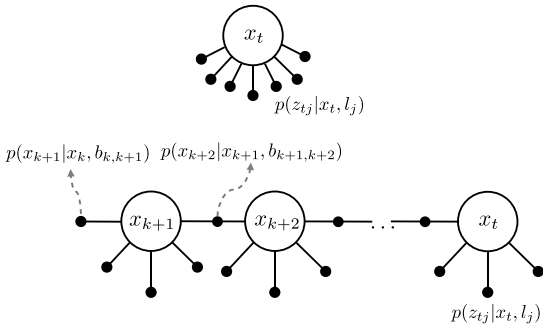


Fig. 7. The factor graphs of PTAM tracker (top) and our trackers (bottom).

keyframes to build the map. We denote the set of unknown camera poses and landmarks by $X = \{x_i\}_{i=1}^{n}$ and $L = \{l_j\}_{j=1}^{m}$ respectively, the set of all visual measurements by $Z = \{z_{ij}\}$, with z_{ij} the visual measurement of landmark j viewed from camera i, and the set of all odometry measurements $B = \{b_{ik}\}$, with b_{ik} the odometry measurement between the camera poses i and k.

The map-building problem is then to recover the maximum a posteriori (MAP) estimate, given by

$$X^*, L^* = \operatorname*{argmax}_{X,L} p(X, L, Z, B) = \operatorname*{argmax}_{X,L} p(x_1) \prod_{i,k} p(x_i | x_k, b_{ik}) \prod_{i,j} p(z_{ij} | x_i, l_j).$$

This general map-building problem can be posed in terms of inference in a factor graph [8]. As shown in the bottom of Fig. 6, the camera poses x_i and the landmarks l_j are represented as variable nodes (white circles) in the graph. The factor nodes (black dots) in the graph represent the prior densities $p(x_1)$ on the variable nodes, the motion models $p(x_i | x_k, b_{ik})$ between two poses x_i and x_k given the odometry measurement b_{ik}, and the measurement likelihood models $p(z_{ij} | x_i, l_j)$ constraining a pose x_i and a landmark l_j, given the corresponding visual measurement z_{ij}. The measurement model can be derived for any type of landmark and their visual measurements, using any standard camera projection model. In Section 5 we give a detailed measurement model for the specific landmarks we use. For comparison, the factor graph of the original PTAM is shown at the top of Fig. 6. It has no odometry factors between camera poses.

This problem can be solved by techniques like bundle adjustment [37,1], smoothing and mapping [8], or incremental smoothing and mapping methods [38]. We use the state-of-the-art incremental smoothing and mapping algorithm iSAM2 [38] implemented in our GTSAM library[2] for the actual inference.

[2] https://borg.cc.gatech.edu/download.

4.3. The tracker

While our mapper is fairly similar to PTAM's, our tracker differs significantly. PTAM's tracker can robustly localize the current camera pose x_t within the known map (received from the mapper), by relying on the visual measurements z_{tj} of *many* landmarks l_j visible in the current image t. To do so, it solves the well-known camera resectioning problem, i.e., computing the optimal camera pose from measurements of known landmarks:

$$x_t^* = \operatorname*{argmax}_{x_t} p(x_t | \{z_{tj}, l_j\}_{j=1..m}) = \operatorname*{argmax}_{x_t} \prod_j p(z_{tj} | x_t, l_j).$$

The corresponding factor graph is very simple: there is only a single variable node x_t, and a single (unary) resectioning factor for each of the visual measurements, parameterized by the *corresponding* known landmark. The top of Fig. 7 shows the factor graph of PTAM's tracker.

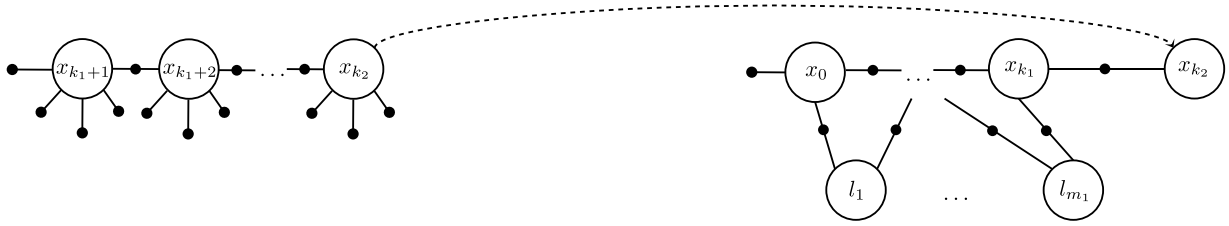
With the odometry measurements to compensate for the lack of visual measurements and landmarks in the environment, our tracker is more involved. We compute not only the current pose, but also all previous poses since the latest keyframe received from the mapper. To make things precise, let $X_{]k,t]} = \{x_i\}_{i=k+1}^{t}$ be those unknown camera poses, with k the index of the latest keyframe received. Also, let L^k be the set of known landmarks received from the mapper, $Z^k_{]k,t]}$ the visual measurements from frame $k+1$ to frame t, and $B_{[k,t[} = \{b_{i,i+1}\}_{i=k}^{t-1}$ the set of odometry measurements from frame k to t.

Our tracker computes:

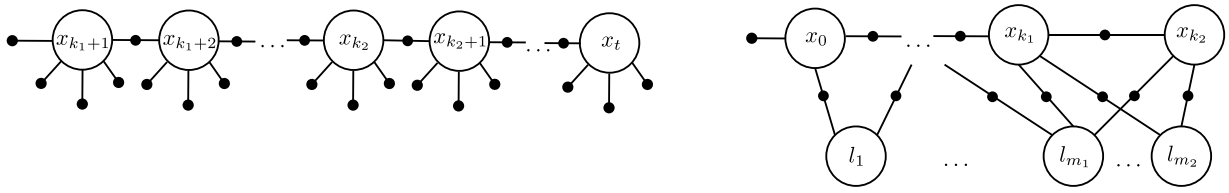
$$X^*_{]k,t]} = \operatorname*{argmax}_{X_{]k,t]}} p(X_{]k,t]} | x_k, L^k, Z^k_{]k,t]}, B_{[k,t[}) = \operatorname*{argmax}_{X_{]k,t]}} \prod_i p(x_i | x_{i-1}, b_{i-1,i}) \prod_{i,j} p(z_{ij} | x_i, l_j)$$

where $i \in]k, t]$ and where the index j ranges over landmarks in L^k observed in frame i. The corresponding factor graph is shown in the bottom of Fig. 7. We also use our GTSAM library mentioned above to optimize this graph.

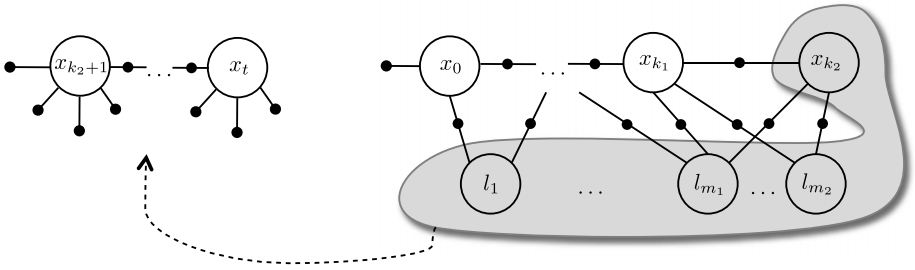
(a) Factor graphs of the tracker (left) and the mapper (right) before update.



(b) The tracker sends a new keyframe (and its odometry measurement from the previous keyframe) to the mapper.



(c) While the mapper is computing new landmarks and optimizing the graph, the tracker keeps on adding new poses and measurements.



(d) After receiving the new map and the latest keyframe pose from the mapper, the tracker removes the past poses and updates the remaining graph.

Fig. 8. The communication and updating process between our tracker and mapper.

At runtime, the tracker decides when to send a keyframe to the mapper, to request for new landmarks in the new parts of the scene. While waiting for the results from the mapper, it keeps on adding new camera poses and optimizing the graph as it receives new images and odometry measurements from the robot. In our experiments, the tracker simply sends a keyframe to the mapper after every 10 frames. Based on the speed of our robot and the field of view of its frontal camera, we found that this 10-frame sampling frequency is sufficient for the mapper to cover the space, while preventing redundant overlap between them.

Upon receiving a new keyframe pose and landmarks back from the mapper, the tracker removes from the graph all the past frames before the latest keyframe received, including the keyframe itself. This process is shown in Fig. 8. Based on the new optimal keyframe pose and landmarks, it then updates the prediction, redoes data association, and re-optimizes all the remaining poses in the graph using the new prediction as the initial value. The data association step might not be necessary if the keyframe and landmark updates from the mapper are not very different from their values currently used in the tracker. However, this is not the case at loop closure, as we will discuss next.

4.4. Loop closing

While PTAM cannot close the loop, as mentioned in its original paper [1], our use of odometry factors allows us to employ a simple loop closing mechanism. Although feature-based loop closure detection schemes [39,40] are available, we employ a simple yet effective loop-closure detection method that works well for typical indoor environments with straight orthogonal hallway segments. Our method is based on the Small-Blurry-Image (SBI) technique used in PTAM for camera relocalization from tracking loss [21]. We implement the loop closure module in the mapper thread due to its high computational complexity.

In particular, to detect a loop closure, we first determine if current hallway segment is potentially a pre-visited segment. When the robot makes a turn around a corner, we know that it is entering a new segment. Thus, we can divide the trajectory into segments, and average the yaw angles of the robot during its stay on a segment and use it as the segment direction. If the current segment direction is similar to another segment in the past, a potential match is found.

To confirm a loop closure from a potential match, we find the keyframe in the old candidate segment that best matches with our

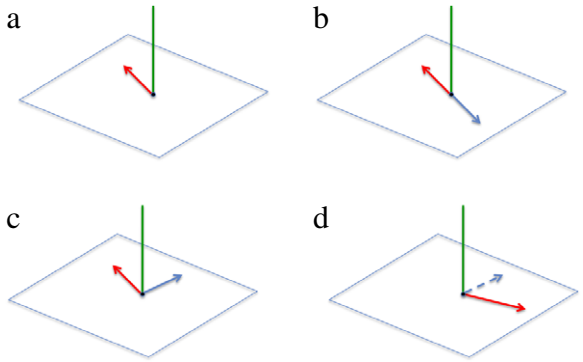


Fig. 9. (a) Our landmark encodes a vertical line position and a wall direction. (b) Two landmarks with opposite wall directions sharing one vertical line. (c) Two landmarks encoding an edge. (d) Two landmarks, one invisible.

current keyframe, and determine if their difference is small enough to consider them a loop closure. We do this by computing the SBIs, down-sampled and blurred images [21] for keyframes in all segments, and using the sum-of-square-differences (SSD) of pixels from the candidate keyframe's SBI and current SBI to measure the difference between them. In order to aid the matching, we also use an image alignment technique to optimize for the relative camera rotation that best aligns the two SBIs before calculating the SSD. Finally, if the best match has differences less than a threshold, we consider it a good match, and a loop closure is found.

When a loop closure is found, we add a loop-closure factor constraining the relative pose between the latest keyframe and the best match keyframe to the mapper. Since keyframes in the mapper regularly sample the space and the SBI match guarantees that the two frames are very close together, we simply predefine the values for this constrained relative pose measurement and use a reasonably large uncertainty as its noise model. We note that other traditional feature-based techniques, e.g. detecting and matching features between the two keyframes using fundamental matrix RANSAC, can also be used; however, their effectiveness is questionable with the lack of features and the regularity of indoor environments.

At loop closure, after receiving the latest keyframe and landmarks from the mapper, it is important for the tracker to redo the data associations for all visual measurements of previous frames in its graph and re-compute the initial values of those poses for optimization. This is because when closing the loop, the latest keyframe can be shifted very far back to an old place in the map, moving the attached frames together with it due to their odometry constraints. Hence, the visual measurements need to be re-associated with these old landmarks, and the initial values need to be recomputed to make sure the nonlinear optimization process converges at the correct local minima.

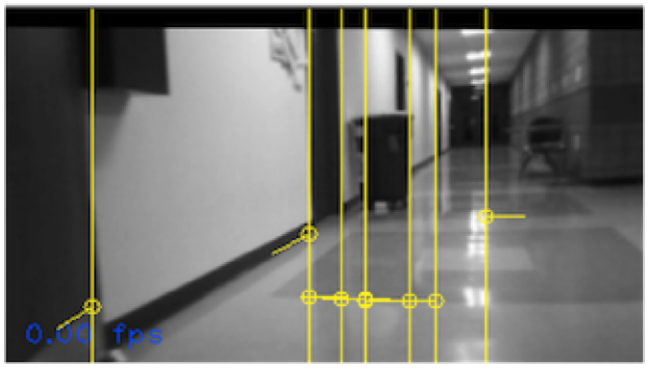
5. Wall–floor features

For the typical indoor environment in our experiment (Fig. 2), choosing the right type of visual landmarks is challenging. Corner-type features are easy to detect but most of them are far away at the end of the hallway and provide little information for localizing the camera. Many solutions have been proposed to deal with this type of environment; for example, encoding the floor–wall boundary in each column of the input image [41], categorizing all possible types of corners to generate all hypotheses of the environment structure [24], generating multiple hypotheses of wall–floor intersection lines from detected edges [25], or exploiting the floor–ceiling planar homology [23].

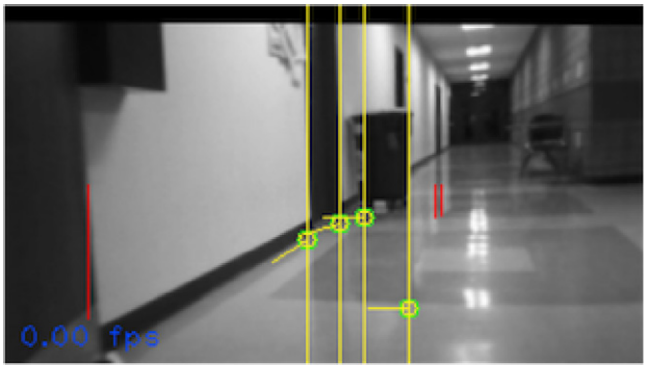
Our system overcomes such difficulties by adopting and improving the Wall–Floor Features introduced in our previous work [4]. These landmarks lie on the intersection of a vertical line on the wall and the intersecting floor plane. As shown in Fig. 9, each landmark encodes a single wall direction and their combinations can model any type of wall corner configurations. We represent each landmark as an element of the Lie-group $\mathbb{SE}(2)$, encoding its 2D position and direction on the floor.

To detect these landmarks in the image, instead of using steerable filters and gradient walk, as done in our previous work, we use the Histogram of Gradient (HoG) of a 7×7 patch around the pixel to gain more resilience to pixel intensity noises. Inspired by [42], our HoG is computed by accumulating the gradient magnitudes of each pixel in the patch into the corresponding histogram bin associated with the pixel gradient orientation.

More specifically, we first detect the vertical line in the landmark by rectifying the image using the rotation estimate from the on-board server, so that vertical lines in the 3D space are also vertical lines in the image, as shown in Fig. 10. After rectifying the image, we detect vertical lines as local maxima along the x-axis of the sum of horizontal image gradients as done in our previous work. Next, we compute HoG responses for points along the vertical line, starting from the middle of the image towards its bottom, up to the pixel where its HoG component in the horizontal gradient bin is less than a threshold, i.e. it is not a pixel on the vertical line anymore. We then mark this point as the end of the vertical line, and continue to search for another pixel, within a small distance from this end point, which has the maximum HoG component in a non-horizontal gradient bin. If this maximum non-horizontal bin HoG response is larger than a threshold, this pixel is considered a Wall–Floor Feature and the direction of the bin with the maximum



(a) Wall–Floor Features (circles) detected by the previous method in [4].



(b) Wall–Floor Features (green circles) detected by the new method using HoG. Red color denotes the vertical lines that have no features close to their lowest end points. The number of spurious features is reduced significantly.

Fig. 10. Detection results. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

HoG component encodes the direction of the wall–floor intersection line.

Fig. 10 shows our detection results. Unlike the previous method, which simply chooses the lowest point with the largest non-horizontal gradient as a feature, this new method can significantly reduce the number of false features by constraining the features to be close to the end point of a vertical line. However, as seen in Fig. 10, due to the reflection of vertical lines on the floor, some spurious features still exist. These features are very unstable and become undetectable very easily. We remove them by checking their uncertainty and visibility in the frames.

The projection of a landmark onto an image given the camera pose is also an element of $\mathbb{SE}(2)$ encoding its projected point and projected direction in the image. Hence, our likelihood function $p(z_{ij} \mid x_i, l_j)$ can be formulated from a Gaussian distribution on $\mathbb{SE}(2)$. However, the simple representation of $\mathbb{SE}(2)$ allows us to treat the measurement feature $z_{ij} \in \mathbb{SE}(2)$ and the predicted projection $p_{ij} \in \mathbb{SE}(2)$ of the landmark l_j given the camera pose x_i as two 3-vector (x, y, θ)'s, encoding their projection points (x, y)'s and directions θ's in the image. Our measurement model is simply $p(z_{ij} \mid x_i, l_j) = \mathcal{N}(p_{ij}; z_{ij}, \Sigma_m)$, where Σ_m is the covariance of the measurement noise.

Since the landmarks lie on the floor, their 3D world positions can be initialized efficiently from a single image by back-projecting the detected image locations onto the floor using the predicted camera orientation and the metric height measurement. This is another advantage of using these features over the traditional corner-type features, which depths have to be initialized by maintaining a set of hypotheses until converged [2], or using the inverse-depth parameterization technique with non-Gaussian priors [43]. As a result, this combination of features on the floor and the height measurements also provides us with the non-drifting absolute scale information, a well-known challenge for pure monocular SLAM systems.

6. Experiments

We evaluate our system for (1) offline map-building with loop closures and (2) online autonomous navigation following a vista and avoiding collisions. The first experiment evaluates the quality of our parallel tracking and mapping with odometry framework, while the second one assesses our system's ability for autonomous navigation and exploration. For both experiments, we fly an AR.Drone 2.0 quad-rotor in the hallway shown in Fig. 11, at one meter above the ground.

6.1. Platform

The quad-rotor is equipped with a 30 fps HD frontal camera, a 60 fps QVGA bottom camera, an ultrasound height sensor, and an IMU system with a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer. An on-board server program on the quad-rotor filters measurements from the bottom camera, the height sensor, and the IMU system to estimate the pose of the quad-rotor [34].

We developed an additional on-board program to obtain images from the frontal camera and stream JPEG compressed 320 × 180 grayscale images to the tracker running on a MacbookPro 6.2 laptop with a 2.4 GHz Intel Core i5 CPU and 4 GB DDR3 RAM. Our on-board program also receives estimated robot poses from the on-board server and creates odometry measurements between the frames, using their relative pose. However, these odometry measurements are not completely reliable over time, as indicated by the inaccurate trajectory estimates from the quad-rotor shown in Fig. 11. They systematically underestimate the traveled distance, perhaps due to the lack of features on the floor corrupting the motion estimation scheme that largely depends on the features from the bottom-facing camera [34].

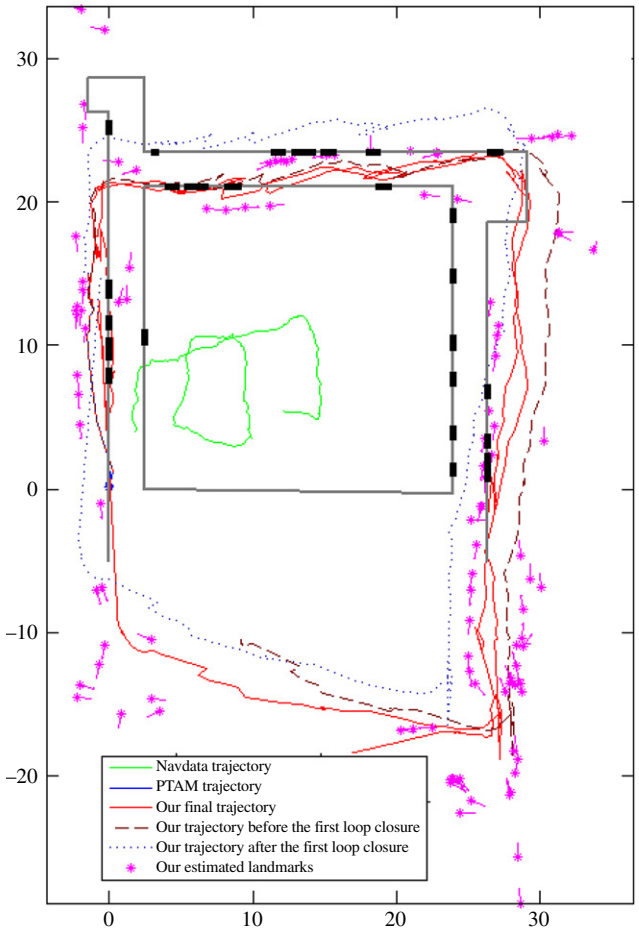


Fig. 11. Comparison between the ground-truth map of the environment and the estimated trajectories from the quad-rotor's on-board program, PTAM, and our system. Our map is manually-aligned to fit with the ground-truth map. The Wall–Floor Features are shown in magenta and the ground-truth floor layout is in gray (walls) and black (doors). Our estimate roughly matches with the ground-truth map even with an unreliable odometry input and a small number of visual features.

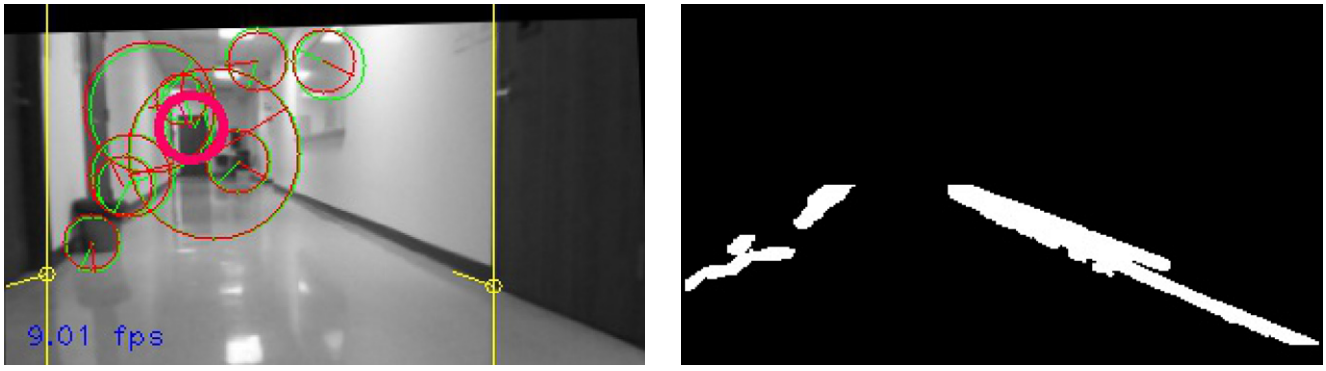
6.2. Map-building results

We evaluate the quality of our map by running our system offline on a set of video frames and sensor data recorded from the quad-rotor, flying manually twice around the square hallway for loop closures (see Appendix A). We compare our results with the original PTAM implementation provided by Klein et al. [1], and also with the hand-measured ground-truth of the test environment.

We first run PTAM with our dataset to analyze its performance in feature-poor environments. We initialize the system as required [1] using the last frame when the quad-rotor is on the floor and the first frame when it stabilizes in the air. Since the quad-rotor is flying about a meter high, we set the initial scale of the system, determined by the baseline between these two frames, as 1 m.

Fig. 11 shows the trajectory of the quad-rotor estimated by PTAM. As we predicted, PTAM cannot estimate the motion and shows that the quad-rotor does not move. This is because there are not enough corner features in the scene, and the few that are available are mostly at the end of the hallway, providing little information about the robot's forward movement; these features have almost no motion parallax as they are close to the focus-of-expansion point on the image. PTAM also breaks down when the quad-rotor starts to turn around the first corner, failing to initialize new landmarks with no motion parallax.

Using the same dataset, we show the results of our system in Fig. 11, where the estimated trajectory is in red and the landmarks



(a) Original frame with vistas (red) and detected Wall–Floor Features (yellow). (b) Wall inference mask, showing wall–floor intersection lines in white.

Fig. 12. Wall inference results in a key-frame. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

in magenta. Due to drift and unreliability in sensor readings during AR.Drone's take-off sequence [34], we only start our system once the drone stabilizes in the air. Since the entire map depends on the first robot pose at the system start, which is arbitrary due to the drift, we manually rotate our map to match the ground-truth map orientation.

Our system successfully finishes the sequence with a proper loop closure. With the aid of the odometry measurements, our system does not fail when the robot yaws around the corner, or when it cannot detect any visual features in the environment. For example, during the last hallway segment before the loop closure at the bottom of Fig. 11, it relies sorely on the incorrect odometry measurements obtained from the quad-rotor's on-board program and underestimates the amount of its forward motion. This is evident in the estimated trajectory before the first loop closure, shown in dashed lines in Fig. 11. Due to the underestimated forward motion, the estimation results show that it is at the middle of the bottom hallway, when it has already turned around the corner. However, with the loop closure in our system, this is corrected, as shown in the trajectory after the first loop closure in Fig. 11. Our final map and trajectory roughly match the ground-truth map of the environment, even with the odometry input that drifts significantly.

6.3. Autonomous flight results

In these experiments, we run both our tracker and mapper online on the Macbook Pro, building the map and inferring the environment structure from images and sensor data streamed over Wifi from the on-board programs. To control the quad-rotor, our tracker also sends back control commands over the network to the low-level controller running on-board [34]. Although the tracker runs very fast, network latency is still a challenging problem for real-time control. We mitigate this issue by setting the maximum flying speed of the quad-rotor slow enough for our off-board controller to cope with changes in the environment in the presence of network latency. Our system can be modified to more robustly deal with network latency and this will be discussed in future work.

We steer the robot towards vistas which prevents frontal head-on collision with obstacles. However, to also avoid lateral collisions with the wall, we post-process the map of Wall–Floor Features, to infer the wall locations as follows.

Wall inference: To infer the local wall locations, we accumulate evidence of walls in a local occupancy grid on the floor. Given the estimated maps of our Wall–Floor Features and the key-frame poses, we first project each landmark as a corresponding line on the key-frame images and compute HoG descriptors, as described in Section 5, for pixels within a fixed distance from these lines. Pixels with large HoG component in the line orientation bin are masked as shown in Fig. 12 as possible wall–floor intersection lines and contribute their votes to the corresponding grid cells on the floor.

Avoiding lateral collisions: Given the local occupancy grid centered at the current robot position, we infer the distance to the walls on the sides of the robot by attempting to equate the distance to the left and the right by changing the MAV's yaw and roll rates. This strategy prevents side collisions and navigates the robot in the middle of the environment.

Given the binarized local occupancy grid centered at the current robot position, we can infer the directions of the walls on the sides of the robot and compute the distances to these walls to navigate along the middle of the hallway. We first find two lines that pass through the robot center and are orthogonal to the possible left and right walls in the local grid by minimizing the variances of projections of nonzero grid cells on each of the lines. More specifically, let $b = [\cos\theta \ \sin\theta]^\top$ be the line direction as a unit vector, o be the local grid center, and $v_i = u_i - o$ be a vector from the grid center to a nonzero grid cell u_i on one side of the robot, we find θ that minimizes the variance of $d = v^\top b$. This variance is computed as: $f(\theta) = E[d^2] - (E[d])^2$ and can be minimized in closed-form by solving for $\frac{\partial f}{\partial \theta} = 0$, leading to a quadratic equation of $\tan(\theta)$. Finally, knowing the minimum distances d_l, d_r from the robot to the left and right walls, we compute the offset of the robot to the middle of the hallway by calculating $(d_l - d_r)/2$. The obtained offset is used in a simple PID controller to keep the robot on the middle of the path.

With the closed-form formula, the algorithm can produce fast accurate results while the robot heading is relatively parallel to the side walls. However, when turning corners, where multiple wall directions appear on one side of the robot, our method produces invalid results. We solve this problem by simply enforcing a threshold on the maximum variance of projections.

Combining these strategies, we obtain a system that can avoid collisions in both forward and side directions, fly towards unexplored open areas, while also simultaneously localizing and building a map of the environment. We achieve avoiding any collisions 8 out of 10 experiments in the hallway shown in Fig. 12. Repeated failures to detect landmarks and the instability of the AR.Drone in the narrow hallway are main reasons for the failure cases.

In terms of speed, our tracker runs favorably at 10–12 fps, while the mapper runs at 1–2 fps with the majority of time spent on waiting for new keyframes and odometry measurements from the tracker. Comparing to iSAM2 [38], which is also very fast due to its incremental computational nature, our framework allows the extra resource in the mapper to be utilized for other time-consuming wall inference tasks without affecting the tracker speed.

7. Discussion and future work

We have presented a vision-based system that enables autonomous navigation strategies on an MAV in feature-poor indoor environments, which could not be achieved in previous work in the absence of heavy and power-hungry sensors. We are able to steer the robot towards open areas with vistas and avoid lateral collisions by inferring the local scene structure with our map of Wall–Floor Features. We adopt a parallel tracking and mapping framework to achieve fast responses to changes in the environment while maintaining the computationally expensive wall inference and map-building tasks in a parallel process.

We have shown that a typical monocular SLAM system is not suitable for robot navigation with a frontal camera, because (1) features in front of the robot do not provide enough information for localization when the robot moves forward, and (2) there is not enough motion parallax to triangulate new landmarks in the scene when the robot rotates to change its heading. In fact, motions typically performed by robots, mainly moving forward and purely rotating to change direction, are challenging for any monocular visual SLAM system in general. The lack of visual landmarks in typical indoor environments adds to the challenge for feature-based systems.

By fusing in the odometry measurements, we have shown that our framework can overcome the key difficulties in frontal monocular SLAM for robot navigation. The new framework also enables a simple loop-closing mechanism to correct the robot trajectory and the map.

Although our method of combining vistas and Wall–Floor Features advances autonomous navigation and exploration capabilities of MAVs, there remain some limitations as future work. The most critical limitation is the lack of ability to autonomously turn corners for exploring a large area. Our current strategy of steering towards vistas detected in the frontal camera of the robot does not work well with finding openings to the sides and turning to lateral directions. We have some preliminary results suggesting that the wall inference strategy can be further extended to find gaps in the walls to mark as potential corners to turn towards. However, this strategy requires further work to achieve reliable performance.

Some improvements can also be made to make the complete system less sensitive to thresholds and be more robust to other lighting conditions. Our wall-inference is sensitive to thresholding values, and requires fine-tuning for the specific lighting condition in the environment. In addition, our criteria for vistas (1) and (2) may have some practical limitations without perfect projective camera imaging. For example, the rolling shutter effect of the low-quality camera on the AR.Drone may affect the size of the features and violate (1).

Another direction of our future work is to further leverage the benefits of the parallel tracking and mapping framework for robotics applications. In our current implementation, the tracker and the mapper both run off-board but communicate with each other over TCP/IP. In the future, we plan to implement the tracker on the quad-rotor itself so that its results can be immediately accessed by the on-board controller, eliminating the effects of network latency on real-time control. The mapper can also be improved to utilize other visual information in the environment, using state-of-the-art structure inference methods.

Acknowledgment

This work is supported by an ARO MURI grant, award number W911NF-11-1-0046.

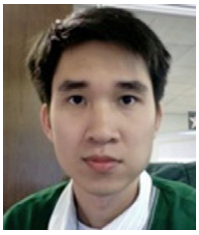
Appendix A. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.robot.2014.03.010.

References

[1] G. Klein, D. Murray, Parallel tracking and mapping for small AR workspaces, in: IEEE and ACM Intl. Sym. on Mixed and Augmented Reality, ISMAR, Nara, Japan, 2007, pp. 225–234.
[2] A. Davison, I. Reid, N. Molton, O. Stasse, MonoSLAM: real-time single camera SLAM, IEEE Trans. Pattern Anal. Mach. Intell. 29 (6) (2007) 1052–1067.
[3] C. Bills, J. Chen, A. Saxena, Autonomous MAV flight in indoor environments using single image perspective cues, in: Proceedings of the 2011 IEEE International Conference on Robotics and Automation, 2011, ICRA 2011, 2011.
[4] K. Ok, D.-N. Ta, F. Dellaert, Vistas and wall-floor intersection features—enabling autonomous flight in man-made environments, in: Workshop on Visual Control of Mobile Robots ViCoMoR, 2012.
[5] V. Murali, S. Birchfield, Autonomous exploration using rapid perception of low-resolution image information, Auton. Robots (2012) 1–14.
[6] S. Weiss, R. Siegwart, Real-time metric state estimation for modular vision-inertial systems, in: IEEE Intl. Conf. on Robotics and Automation, ICRA, IEEE, 2011, pp. 4531–4537.
[7] M. Achtelik, M. Achtelik, S. Weiss, R. Siegwart, Onboard IMU and monocular vision based control for MAVs in unknown in and outdoor environments, in: Proc. of the IEEE International Conference on Robotics and Automation, ICRA, 2011.
[8] F. Dellaert, M. Kaess, Square root SAM: simultaneous localization and mapping via square root information smoothing, Int. J. Robot. Res. 25 (12) (2006) 1181–1203.
[9] A. Bachrach, R. He, N. Roy, Autonomous flight in unknown indoor environments, Int. J. Micro Air Vehicles (ISSN: 1756-8293) 1 (4) (2009) 217–228.
[10] A. Bachrach, S. Prentice, R. He, N. Roy, RANGE—robust autonomous navigation in GPS-denied environments, J. Field Robot. 28 (5) (2011) 644–666.
[11] M. Achtelik, A. Bachrach, R. He, S. Prentice, N. Roy, Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments, in: SPIE Defense, Security, and Sensing, International Society for Optics and Photonics, 2009, 733219–733219.
[12] S. Grzonka, G. Grisetti, W. Burgard, Towards a navigation system for autonomous indoor flying, in: IEEE International Conference on Robotics and Automation, 2009, ICRA'09, ISBN: 1050-4729, 2009, pp. 2878–2883.
[13] S. Grzonka, G. Grisetti, W. Burgard, A fully autonomous indoor quadrotor, IEEE Trans. Robot. 99 (2012) 1–11.
[14] S. Shen, N. Michael, V. Kumar, Obtaining liftoff indoors: autonomous navigation in confined indoor environments, IEEE Robot. Autom. Mag. 20 (4) (2013) 40–48.
[15] S. Scherer, S. Singh, L. Chamberlain, S. Saripalli, Flying fast and low among obstacles, in: 2007 IEEE International Conference on Robotics and Automation, ISBN: 1424406013, 2007, pp. 2023–2029.
[16] A. Bry, A. Bachrach, N. Roy, State estimation for aggressive flight in GPS-denied environments using onboard sensing, in: IEEE Intl. Conf. on Robotics and Automation, ICRA, IEEE, 2012, pp. 1–8.
[17] A. Wendel, A. Irschara, H. Bischof, Natural landmark-based monocular localization for MAVs, in: ICRA, IEEE, 2011, pp. 5792–5799.
[18] M. Blösch, S. Weiss, D. Scaramuzza, R. Siegwart, Vision based mav navigation in unknown and unstructured environments, in: 2010 IEEE International Conference on Robotics and Automation, ICRA, 2010, pp. 21–28.
[19] S. Weiss, M. Achtelik, L. Kneip, D. Scaramuzza, R. Siegwart, Intuitive 3D maps for MAV terrain exploration and obstacle avoidance, J. Intell. Robot. Syst. 61 (2011) 473–493.
[20] S. Shen, Y. Mulgaonkar, N. Michael, V. Kumar, Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor, in: Robotics: Science and Systems, RSS, 2013.
[21] G. Klein, D. Murray, Improving the agility of keyframe-based SLAM, in: European Conf. on Computer Vision, ECCV, Marseille, France, 2008.
[22] E. Eade, T. Drummond, Edge landmarks in monocular SLAM, in: Proc. British Machine Vision Conf., 2006.
[23] A. Flint, D. Murray, I. Reid, Manhattan scene understanding using monocular, stereo, and 3D features, in: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 2228–2235.
[24] D. Lee, M. Hebert, T. Kanade, Geometric reasoning for single image structure recovery (2009).
[25] G. Tsai, C. Xu, J. Liu, B. Kuipers, Real-time indoor scene understanding using Bayesian filtering with motion cues, in: International Conference on Computer Vision, 2011.
[26] S. Weiss, D. Scaramuzza, R. Siegwart, Monocular-SLAM–based navigation for autonomous micro helicopters in GPS-denied environments, J. Field Robot. 28 (6) (2011) 854–874.
[27] J. Engel, J. Sturm, D. Cremers, Camera-based navigation of a low-cost quadrocopter, in: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, IROS, Vol. 320, 2012, p. 240.
[28] D. Lee, et al., A theory of visual control of braking based on information about time-to-collision, Perception 5 (4) (1976) 437–459.
[29] N. Ancona, T. Poggio, Optical flow from 1-D correlation: application to a simple time-to-crash detector, Int. J. Comput. Vis. 14 (2) (1995) 131–146.
[30] D. Coombs, M. Herman, T. Hong, M. Nashman, Real-time obstacle avoidance using central flow divergence and peripheral flow, in: Fifth International Conference on Computer Vision, 1995. Proceedings, IEEE, 1995, pp. 276–283.

[31] B. Horn, Y. Fang, I. Masaki, Time to contact relative to a planar surface, in: 2007 IEEE Intelligent Vehicles Symposium, IEEE, 2007, pp. 68–74.
[32] D. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[33] H. Bay, T. Tuytelaars, L.V. Gool, SURF: speeded up robust features, in: European Conf. on Computer Vision, ECCV, 2006.
[34] P. Bristeau, F. Callou, D. Vissière, N. Petit, The navigation and control technology inside the AR.Drone micro UAV, in: World Congress, vol. 18, 2011, pp. 1477–1484.
[35] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2000.
[36] H. Strasdat, J.M.M. Montiel, A.J. Davison, Real-time monocular SLAM: why filter? in: IEEE Intl. Conf. on Robotics and Automation, ICRA, 2010, pp. 2657–2664.
[37] B. Triggs, P. McLauchlan, R. Hartley, A. Fitzgibbon, Bundle adjustment—a modern synthesis, in: W. Triggs, A. Zisserman, R. Szeliski (Eds.), Vision Algorithms: Theory and Practice, in: LNCS, vol. 1883, Springer-Verlag, 2000, pp. 298–372.
[38] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, F. Dellaert, iSAM2: incremental smoothing and mapping using the Bayes tree, Int. J. Robot. Res. 31 (2012) 217–236.
[39] A. Angeli, S. Doncieux, J.-A. Meyer, D. Filliat, Real-time visual loop-closure detection, in: IEEE Intl. Conf. on Robotics and Automation, ICRA, IEEE, 2008, pp. 1842–1847.
[40] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, J.D. Tardos, An image-to-map loop closing method for monocular SLAM, in: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, IROS, Nice, France, 2008.
[41] E. Delage, H. Lee, A. Ng, A dynamic Bayesian network model for autonomous 3D reconstruction from a single indoor image, in: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2, IEEE, 2006, pp. 2418–2428.
[42] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Conf. on Computer Vision and Pattern Recognition, CVPR, 2005, pp. 886–893.
[43] J. Montiel, J. Civera, A. Davison, Unified inverse depth parametrization for monocular SLAM, in: Robotics: Science and Systems, RSS, 2006.



Duy-Nguyen Ta is a Ph.D. student in the School of Interactive Computing, College of Computing, Georgia Institute of Technology. He received an M.Eng. degree in computer engineering from National University of Singapore in 2007, and the B.Eng. degree in computer engineering from the HCM City University of Technology, Viet Nam, in 2003. His research focuses on computer vision techniques for robot navigation and augmented reality.

Kyel Ok is currently a second year M.S. Student in Computer Science at Georgia Institute of Technology. He received his BASc. degree in Mechatronics Engineering from University of Waterloo in 2011. His research interests focus on state estimation, SLAM, and vision techniques for autonomous MAVs.

Frank Dellaert (M'96) received the Ph.D. degree in computer science from Carnegie Mellon University in 2001, an M.S. degree in computer science and engineering from Case Western Reserve University in 1995, and the equivalent of an M.S. degree in electrical engineering from the Catholic University of Leuven, Belgium, in 1989. He is currently a Professor in the College of Computing at the Georgia Institute of Technology. His research focuses on probabilistic methods in robotics and vision. Prof. Dellaert has published more than 180 articles in journals and refereed conference proceedings, as well as several book chapters. He also serves as Associate Editor for IEEE TPAMI.