Onboard Monocular Vision for Landing of an MAV on a Landing Site Specified by a Single Reference Image

Shaowu Yang, Sebastian A. Scherer, Konstantin Schauwecker and Andreas Zell

***Abstract*— This paper presents a real-time monocular vision solution for MAVs to autonomously search for and land on an arbitrary landing site. The autonomous MAV is provided with only one single reference image of the landing site with an unknown size before initiating this task. To search for such landing sites, we extend a well-known visual SLAM algorithm that enables autonomous navigation of the MAV in unknown environments. A multi-scale ORB feature based method is implemented and integrated into the SLAM framework for landing site detection. We use a RANSAC-based method to locate the landing site within the map of the SLAM system, taking advantage of those map points associated with the detected landing site. We demonstrate the efficiency of the presented vision system in autonomous flight, and compare its accuracy with ground truth data provided by an external tracking system.**

Fig. 1: Our MAV navigating autonomously while searching for a textured landing site.

I. Introduction

Micro Aerial Vehicles (MAVs) are a growing research area that has attracted much attention in the robotics community in recent years. One focus has been on using onboard sensors such as cameras and laser scanners, which do not rely on any external signal, to facilitate their autonomous flights. These onboard sensors are important replacements for GPS sensors in environments where GPS is unavailable or not reliable, as e.g. indoors or in outdoor urban areas.

Compared with other sensors, cameras have a superior potential for environment perception, while still being lightweight, relatively low cost and energy efficient. Furthermore, unlike stereo cameras with small baselines, a monocular camera does not lose its functionality even for large working distances, if scale information is provided. Those advantages make monocular vision very attractive for research on autonomous flight of MAVs, which in general have very limited payload.

Autonomous landing is a basic but also challenging phase for autonomous flights of MAVs. When the exact position of a desired landing site is unknown, an MAV should be able to search for and locate it autonomously, and then land on it to finish an autonomous flight. Monocular visual simultaneous localization and mapping (SLAM) has brought more flexibility to autonomous navigation of MAVs in unknown environments [1]. In fact, it is especially well-suited for the autonomous landing task of an MAV: The problem of slow scale drift, which is inherent to every purely visual monocular SLAM system caused by the unobservability of the scale factor, can hardly cause much effect in such relatively small areas where the MAV is expected to land on.

In this paper we show that the rich information provided by a visual SLAM system can also benefit both the real time detection of a known landing site and its localization. Considering the limited computational power that is typically available onboard an MAV, those processes are normally difficult to be performed in parallel to visual autonomous navigation. We achieve autonomous navigation of our MAV by implementing a constant-time monocular visual SLAM framework, while simultaneously detecting an arbitrary landing site using ORB features [16], and estimating its global pose. The resulting monocular vision system enables the MAV to autonomously search for the landing site in unknown environments (as depicted in Fig. 1), and then land on it once it is found.

II. Related Work

Early research on autonomous navigation of Unmanned Aerial Vehicles (UAVs) mainly relied on pose estimation from GPS sensors, with fusing inertial navigation system (INS) data. Such systems works well for high altitude and long range tasks, but are not suitable in GPS-denied environments. In recent years, more effort has been focused on using computer vision to enable autonomous flight of UAVs. Computer vision methods do not depend on external signals. Moreover, they fit especially well to cases in which precise position control relative to other objects is required,

S. Yang, S. A. Scherer and K. Schauwecker are PhD students with the Department of Computer Science, University of Tübingen, Tübingen, Germany {shaowu.yang, sebastian.scherer, konstantin. schauwecker} @uni-tuebingen.de

A. Zell is full professor with the Department of Computer Science, University of Tübingen, Tübingen, Germany andreas.zell@uni-tuebingen.de

e.g. for the landing tasks of UAVs. Thus, they are highly appreciated for research towards full autonomy of UAVs.

In [19], the landing task of a helicopter is solved by using image moments for object recognition, while the estimation of the relative position with respect to the landing pad still relies on precise height information provided by differential GPS. Garcia-Pardo et al. [8] present a strategy to find a safe landing area by searching the image for a circular area in which all the pixels have a level of contrast below a given threshold. The vision system developed in [6] allows a remote user to define target areas as waypoints or a landing area for a UAV from a high resolution aerial or satellite image. In this work, a Scale Invariant Feature Transform (SIFT) based image-matching algorithm is implemented to find the natural landmarks, and an optical-flow-based method is used for the detection of a safe landing area.

Recently, more vision solutions for autonomous navigation and landing have been presented, due to the fast growing interest in MAVs, and especially quadrotors. Mahony et al. [12] provide a tutorial introduction to modelling, pose estimation and control of such multi-rotor MAVs. Meier et al. [14] present a new self-developed quadrotor system capable of autonomous flight with onboard pose estimation from vision and an Inertial Measurement Unit (IMU), while relying on artificial visual markers. Previous work in [24] features an onboard monocular vision solution for autonomous takeoff, hovering and landing of an MAV based on a circular landing pad. Those works, achieving autonomous flight of MAVs, still depend on pose estimates from artificial landmarks, and are thus not flexible enough for long-term autonomy.

One way to be independent of artificial landmarks is to implement visual odometry or visual SLAM method on MAVs. Fraundorfer et al. [7] extended the system in [14] with autonomous mapping and exploration based on stereo cameras. In [2], [23], Parallel Tracking and Mapping (PTAM) [9] is implemented as a monocular visual SLAM framework for autonomous navigation of MAVs in unknown and GPS-denied environments. Achtelik [1] also use PTAM to provide position estimates for an MAV, while fusing data from an air pressure sensor and accelerometers to estimate the unknown metric scale factor of the monocular vision system. In [22], a modified PTAM, which integrates depth information as presented in [21], is used for position control of an MAV based on stereo vision.

In our work, we also implement our visual SLAM framework based on PTAM, to enable autonomous navigation of an MAV, because of its robustness and its ability to generate an accurate map with a large number of map points from the environment. To land an MAV on an arbitrary landing site, we implement an ORB-feature-based method for landing site detection, running in parallel with the visual SLAM. Furthermore, based on the existing map points, we show that it is possible to robustly estimate the 3D pose of the detected landing site even if the size of it is unknown. This is different from those methods that estimate a relative pose of an MAV with respect to a landing site only based on observations from the landing site itself. An example of such methods is the work in [13], which estimates the 3D pose of a camera for the control of UAVs by tracking a planar object with a known size. Since our pose estimation for MAV position control is provided by a SLAM system, high frequency landing site tracking and pose estimation become unnecessary without losing the final landing accuracy.

III. Visual SLAM for Autonomous Navigation

The visual SLAM framework we use for autonomous navigation of our MAV is based on PTAM. In order to overcome the lack of a scale factor, we implemented an automatic initialization method for PTAM, which can cope with cluttered environments and provides a high accuracy. Additionally, we modify the mapping thread of PTAM to achieve a nearly constant processing time during navigation.

A. Basic Functionality of PTAM

The original PTAM implementation can produce detailed environmental maps with a large number of landmarks, which can be used for accurately tracking the pose of a monocular camera at a high frequency. In order to achieve real-time operation, a main idea proposed in PTAM is to split tracking and mapping into two separate threads, which can be processed in parallel on a dual-core computer. One thread is responsible for tracking the camera motion relative to the current map. The other thread extends this map, which consists of 3D point features that are organized in keyframes, and refines it using bundle adjustment.

In the thread responsible for tracking the camera pose, the FAST corner detector [18] is applied to each image at four pyramid levels, and all map points are projected to the current image coordinate frame, based on a prior pose estimate. The map points located inside the image after this projection are then used for tracking: To locate those points in the current camera image, a fixed-range image search around their predicted positions is performed. During this search, only the FAST corner locations are evaluated for finding the best matches. In our work, those FAST corners will further be used for feature-based object detection, without increasing the computation time in this thread.

The mapping thread integrates new keyframes into the map when requested by the tracking thread, and creates new map points by triangulating FAST corner matches between the new keyframe and its closest neighbors. Local bundle adjustment and global bundle adjustment are continuously performed to refine the map for the rest of the time. Since the map points are actually landmarks of the real-world scene, we will take advantage of their known 3D position for our landing site pose estimation.

B. Automatic Initialization of PTAM

Since there exists a common scale ambiguity inherent to monocular camera systems, PTAM naturally requires additional metric scale information. Since PTAM was originally intended for augmented reality applications [9], an accurate metric scale was not necessary, thus only a coarse scale estimate is applied to the triangulation of the initialization

Fig. 2: A scene when PTAM is initialised. *Top left*, original image. *Top right*, detected circular pattern, labelled with a orange cross. *Bottom left*, vision features in different levels. *Bottom right*, chosen map points.

phase. We deal with this initialization issue by implementing the monocular solution presented in [24], which can robustly estimate the camera pose based on the image projection of a helicopter landing pad pattern, which also works in cluttered environments. Using this method, we can achieve accurate automatic initialization of PTAM during the takoff phase of our MAV, without requiring any additional sensors. Fig. 2 shows an example scene and related results of PTAM, when initialized with this method.

1) Pose Estimation from a Circular Pattern: In [24], we estimated the 6DOF camera pose based on the perspective projection of a typical helicopter landing pad, which consists of a letter "H" surrounded by a circle with a known diameter.

This pad is detected with a method similar to the one presented in [20]. Using adaptive thresholding, we obtain a binarized image that is used to find connected components with a run-based two-scan labeling algorithm. The components are then classified using an artificial neural network. Finally a geometric constraint is applied, enforcing that the letter "H" must be surrounded by a circle. This allows us to detect the pad robustly in real-time with a high frequency.

After applying a Canny edge detector to the image pattern associated with the above pad, we can retrieve the ellipse that corresponds to the projection of the circle in the pad. At this point, we can obtain a 5DOF pose of the camera coordinate frame $\mathcal{C}$ with respect to the world coordinate frame $\mathcal{W}$, which is defined by the pad and obtained by using a computational geometry method based on the known quadratic equation of the projected ellipse. During this step, we also integrate IMU data to eliminate the remaining geometric ambiguity. Finally, fitting an ellipse to the projected contour of the letter "H" provides us with the last DOF of the camera pose, i.e. its yaw angle.

2) Initializing PTAM During Takeoff: Once we obtain an estimate of the camera pose with a height larger than a threshold h_i, then this pose estimate and the image associated with it are sent to PTAM for initialization. If more than a minimum number of FAST features with non-maximum suppression are detected on all four pyramid levels of this image, then we use them to initialize the map of PTAM. We obtain the 3D position of those feature points by assuming that they all lie on the ground plane and by projecting them from their image coordinates to the $z = 0$ plane in the world coordinate frame $\mathcal{W}$. In this way, the world coordinate frame defined in PTAM coincides with $\mathcal{W}$.

C. Using PTAM with Constant Computation Time

Bundle adjustment, which is used for map managing, is the most computationally intensive task in PTAM. To enable PTAM to achieve a nearly constant computation time, we only retain its local bundle adjustment and abandon the global bundle adjustment, since it is very computationally expensive and can stop the mapping thread from adding enough keyframes to facilitate successful tracking. However, we still keep the complete map during exploration.

IV. Landing Site Detection and Pose Estimation

To search for an arbitrary landing site during autonomous navigation of our MAV, we implemented a feature-based object detection scheme. Using one pre-set reference image of the designated landing site, a set of feature matches between the reference image and the currently visible scene can be established. Then the landing site is detected by using a robust RANSAC-based method to estimate the corresponding homography. Because some of the map points produced by PTAM can be associated with the matched features, we can use the 3D position estimates of those map points to estimate the global 3D pose of the landing site, even though there exists no absolute scale information for the landing site. The above process is integrated in the mapping thread of PTAM, as shown in Fig. 3.

A. Brief Overview of ORB

Rublee et al. [16] proposed the ORB (Oriented FAST and Rotated BRIEF) feature based on the FAST keypoint detector

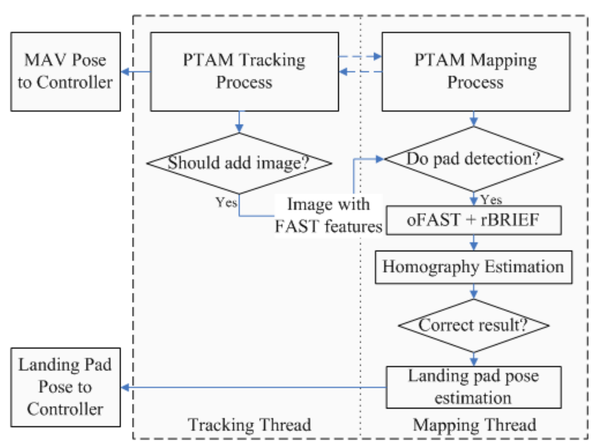


Fig. 3: Landing site detection and pose estimation integrated in PTAM framework.

and the BRIEF descriptor [5], both of which are known for their high computational efficiency. BRIEF uses a binary string constructed from a set of binary intensity tests as an efficient point feature descriptor. Because BRIEF was not designed to be aware of the orientation of a feature point, it is notably lacking rotation invariance [16], which is, however, important for feature-matching-based object detection.

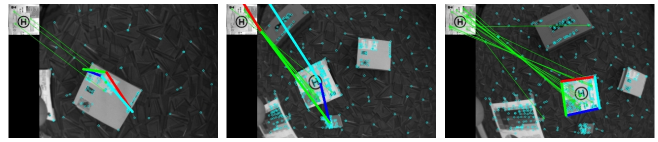
To cope with this issue, Rublee et al. proposed to compute an orientation component for each FAST interest point (oFAST) by using the so-called intensity centroid, which is computed from image moments. BRIEF descriptors for those points are then efficiently rotated according to the orientation component, and thus form the steered BRIEF descriptor. Furthermore, a learning method is developed for choosing a good subset of binary tests, in order to increase the feature variance and reduce correlation among the binary tests, both of which are important for a discriminative feature. The resulting descriptor is named rBRIEF.

B. Applying Multi-Scale ORB to the PTAM Framework

We chose ORB as the feature descriptor for our landing site detection because of its low time cost and high discrimination capability for feature matching. ORB achieves scale invariance by applying the FAST detector to a scale-space pyramid of the original image. Since in the tracking thread of PTAM, FAST points have been detected in four-level pyramid images of the current scene, it is straightforward for us to use those points for further feature description. We chose such a multi-scale method in order to avoid the computation of further pyramid levels, as a compromise between matching performance and time cost. In the mapping thread, we compute orientation components of the FAST points to obtain oFAST features, and use rBRIEF for feature description. We perform both of these operations individually at pyramid level 0 and 1, resulting in two sets of descriptors $\{D_i^c |\ i = 0, 1\}$, each with a size n_i. We discard higher pyramid levels, since at higher levels, a landing site appears too small for us to obtain useful features for matching. For the reference image of the landing site, the number of pyramid levels and the scale factor for producing the pyramid images can vary according to the requirements of scale invariance and available computation time. In this paper we apply a three level pyramid with a scale factor of 1.2 to the reference image, obtaining the reference descriptor sets $\{D_i^r |\ i = 0, 1, 2\}$. A Gaussian blur is applied to each pyramid level before feature detection.

C. Landing Site Detection by Feature Matching

1) Feature Matching: We use a standard feature matching scheme to obtain a set of good feature matches from $\{D_i^r |\ i = 0, 1, 2\}$ to $\{D_i^c |\ i = 0, 1\}$, for estimating the homography $\mathbf{H}_{rc}$ between the reference image of the landing site and the current image frame. For finding all possible matches, we employ a brute-force matcher without cross checking, implemented in OpenCV [4]. It finds the k descriptors with the closest normalized Hamming distances in $\{D_i^c |\ i = 0, 1\}$ for each descriptor in $\{D_i^r |\ i = 0, 1, 2\}$. Similar to [10], [11],



(a) (b) (c)

Fig. 4: Examples of homography estimation results shown in one pyramid level. After eliminating false estimates, only (c) will be regarded as a correct homography estimate.

we consider a match between a reference descriptor and the corresponding descriptor with the closest distance to be valid, if the ratio of the closest to the second closest distance is smaller than a threshold T_r.

2) Homography Estimation: As the ORB feature is applied at different individual pyramid levels of the current camera image, we project all matched feature points to the source pyramid level for calculating the homography. The homography $\mathbf{H}_{rc}$ is estimated by using RANSAC, and then further refined by using the Levenberg-Marquardt method to minimize the image projection error. Please note that we limit the iterations in RANSAC to a relatively small number, in order to make this process more efficient. Since our later landing site pose estimation can cope well with a lower true positive detection rate, we opt in favor of a higher processing performance.

3) Eliminating False Estimates: The reference image forms a quadrilateral Q_r when transformed with $\mathbf{H}_{rc}$ to the current image frame. Some examples of the homography estimates we received can be seen in Fig. 4. We dramatically eliminate false estimates by evaluating some basic properties of this quadrilateral: First, it is required to be a convex polygon. Second, all four vertexes of it should have a reasonable relative distances to their centroid and to each other. This will eliminate estimates like the one shown in Fig. 4b: Although the reference image can be found in the current image frame, we reject this frame since we will not achieve a correct pose estimate of the landing site according to this homography estimate. Third, the number of matched features that are inside of this quadrilateral should be larger than a threshold n_q. We determine whether a point locates inside a polygon using a crossing-number-based method.

D. Locating the Landing Site in the Map

After the landing site has been detected in the current camera image by using the above method, we locate its 3D pose in the world coordinate frame. For this task we take advantage of the environment map produced by PTAM, which can consist of a large number of map points. Doing this provides us with much more tolerance towards false negative detections: Even if the landing site is not tracked at camera frame rate, its final pose estimate will be hardly affected, as the landing site should retain a static position in respect to the environment map. Thus, our method is very flexible in respect to the time intervals at which the mapping thread decides to add a new frame for landing site detection. Furthermore, using the map points ensures that only discriminative features are used for locating the landing site.

We first project all map points to a rectified image frame based on their 3D positions and the calibrated camera model [3]. Again, we use a crossing number method to check whether a projected map points is located within the quadrilateral Q_r (see Sect. IV-C.3). Those inside points form the map points subset $\{p_l\}$.

If the size of $\{p_l\}$ is larger than a threshold n_{lmin}, a RANSAC-based method is applied to the points in $\{p_l\}$ to estimate the dominant plane P_d of the landing site. We perform this step in a similar fashion as in [9]: Many sets of three points are randomly selected to form a plane hypothesis, while the remaining points are tested for consensus. The winning hypothesis is further refined by using the consensus set, resulting in the detected plane normal $\mathbf{n}_p$. Together with the mean 3D coordinate value of all consensus set points $\mathbf{x}_m$, this normal defines the plane P_d. Once an estimate for P_d is achieved, we use its corresponding measurements $\mathbf{n}_p$ and $\mathbf{x}_m$ as the initial guess for the RANSAC procedure when evaluating the next image frame. Thus, a much smaller threshold for the number of RANSAC iterations can be applied, which further reduce time costs.

The pose of the landing site can be calculated by projecting the quadrilateral Q_r to the plane P_d. We define $\mathbf{x}_i^p, i = 0,1,2,3,$ as the four vertices of Q_r, which are the image projections of the four corners P_i of the landing site, with the corresponding world coordinate positions $\mathbf{x}_i^w$. After projecting $\mathbf{x}_i^p$ to a normalized image frame with rectified lens distortions, we obtain the normalized coordinates $\mathbf{x}_i^n = (x_i^n, y_i^n, 1)^T$. In the camera coordinate frame, we then have $\mathbf{x}_i^c = s \cdot (x_i^n, y_i^n, 1)^T$, with s being an undetermined scale factor. Thus, in the world coordinate frame we have

$$\mathbf{x}_i^w = s \cdot \mathbf{R}_{wc} \cdot \mathbf{x}_i^n + \mathbf{t}_{wc}, \tag{1}$$

with $\{\mathbf{R}_{wc}, \mathbf{t}_{wc}\}$ being the camera pose in the world coordinate frame, obtained by the tracking thread. Since P_i is located on the plane P_d, we have

$$\mathbf{n}_p \cdot (\mathbf{x}_i^w - \mathbf{x}_m) = 0. \tag{2}$$

From (1) and (2), we can calculate $\mathbf{x}_i^w$. The landing site pose is then obtained as $\mathbf{x}_k = \sum_{i=0}^{3} \mathbf{x}_i^w$, where k is the current image frame index.

We further refine the landing site pose by integrating m successful estimates of $\mathbf{x}_k$. Estimates with a large difference to $\mathbf{x}_{Lm} = \sum_{k=0}^{m} \mathbf{x}_k$ are assumed to be outliers. The mean value of the remaining inlier is then assumed to be the final landing site pose estimate $\mathbf{x}_L$.

V. Experiments and Results

A. Experimental Setup

1) Quadrotor Platform: Our MAV is based on the open source and open hardware quadrotor platform developed by the PIXHAWK project from ETH Zürich described in [14], which can be found in Fig. 1. The onboard computer is a Kontron microETXexpress computer-on-module (COM) featuring an Intel Core 2 Duo 1.86GHz CPU, 2 GB DDR3 RAM and a 32Gb SSD. The pxIMU inertial measurement unit and autopilot board that we use mainly consists of a MicroController Unit (MCU), and sensors including an accelerometer and a gyroscope. The MCU is a 60 MHz ARM7 microcontroller for sensor readout and fusion, as well as position and attitude control. A PointGrey Firefly MV monochrome camera of only 37 g weight is mounted on the MAV in a downward-facing pose. This camera has an image resolution of 640×480, a maximum frame rate of 60 fps, and is equipped with a lens featuring a 90 degrees viewing angle.

2) External Tracking System: To measure ground truth data of the 6 DOF quadrotor pose and landing site poses, we use an external Optitrack tracking system manufactured by Naturalpoint [1], which comprises 9 infrared cameras in our case. After attaching several highly reflective markers to the quadrotor, the tracking system can provide 6 DOF pose estimates of the quadrotor with a frequency of up to 100 fps. According to our tests, the deviation of the position estimates for a static quadrotor is in the order of only few millimeters.

3) Software: We implemented our software system in several modules using the open-source Robot Operating System (ROS) [17] on Ubuntu Linux 12.04, as it provides the infrastructure for efficient communication among different modules and for logging all onboard data.

B. Navigation and Flight Control Algorithm

1) Nested PID Pose Control: Mellinger et al. [15] describe a nested PID controller that consists of a separate attitude and a position controller. Using a dynamic model of a quadrotor and an accurate 6 DoF pose estimate from an external tracking system, they achieved precise hovering and 3D trajectory control of a quadrotor MAV. To evaluate our vision system, we control the pose of our quadrotor using a very similar controller, which is implemented in the original pxIMU code from the PIXHAWK project. In our case, we set the desired yaw angle to a constant value of $\psi^{des} = 0$. The 3D position estimates from the onboard vision system are used as feedback to the position controller, and a basic Kalman Filter is applied to smooth pose estimation for low level control. The attitude control runs at a frequency of 200 Hz, using the roll and pitch estimates by the IMU, and only the yaw angle is provided by the onboard vision system.

2) Setpoint Path Following for Navigation: In order to search for the landing site, we navigate our MAV with a setpoint-based method, where the MAV follows a predefined searching path. We assume that the MAV has reached a setpoint, if its distance to this point is smaller than a threshold d_s for a period of time t_s. In this case we advance to the next set point on the searching path. Once an initial pose of the landing site $\mathbf{x}_{lini}$ is estimated, we change the setpoint to be above this area, keeping our searching height h_s. After

[1] http://www.naturalpoint.com/optitrack/products/tracking-tools-bundles

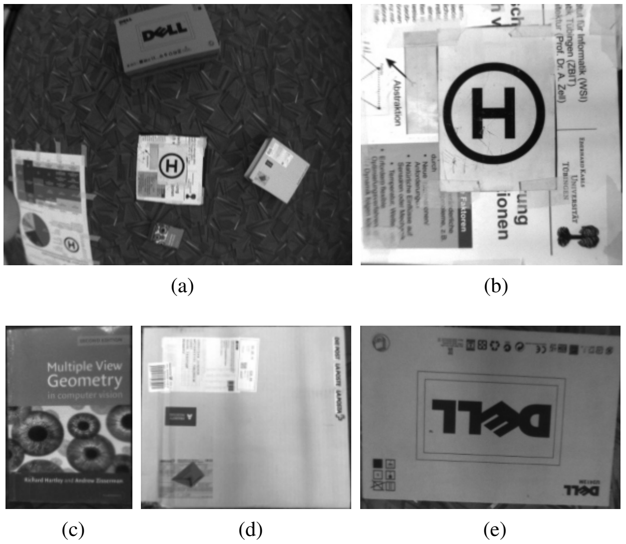


Fig. 5: (a) A scene from the MAV, (b), (c), (d) and (e) are the reference images of the poster landing pad (size 500×500, height 4.5), the book (size 246×175, height 33), the mail package (size 380×335, height 140) and the computer package (size 650×435, height 235), respectively. All size and height are measured in *mm*.

TABLE I: RMSEs (mm) of position estimates for different landing sites during a manual flight.

RMSE	poster	book	mail Pack.	PC Pack.
x-y	24	8	34	43
z	1	8	6	2
3D	24	12	35	43

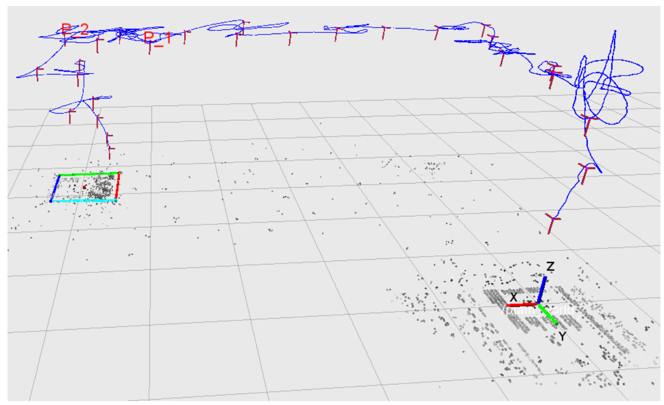


Fig. 7: The built map and MAV trajectory during a searching and landing task.

the final refined pose of the landing site $\mathbf{x}_l = (x_l, y_l, z_l)^T$ is determined, we define the end of the searching path to be $(x_l, y_l, h_s)^T$. Finally, the landing strategy we implemented in [24] is used to land the MAV on the landing site.

C. Landing Site Pose Estimation Results

We evaluate the landing site detection and pose estimation result by processing a video logfile from a manual flight of our MAV above different objects with planar surfaces, which are used to represent different landing sites: a poster pad, a book, a mail package, and a computer package. Each of them has different texture features. Moreover, they are different in size and height. We control the MAV to takeoff from another pad nearby those objects, such that our SLAM algorithm can be initialized by this pad as described in Sect. III-B. Fig. 5a shows a scene during this flight. Reference images of those landing sites are captured by manually holding the MAV above them in different illumination conditions, as show in Fig. 5. Please note that to confirm that our object detection method is invariant to the rotation of a reference image, they are rotated by 180 degrees for this experiment.

We process the same recorded video sequence four times, with each time selecting one of these reference images for landing site detection. The identical MAV trajectory estimated by the visual SLAM algorithm is shown in Fig. 6c and 6f. Fig. 6a, 6b, 6d and 6e show the distribution of the final detection and pose estimation results for the landing sites. The pose estimates of the detected landing sites are projected to the $x-y$ and $x-z$ plane (the world coordinates are indicated as the RGB axis in Fig. 7). The few estimates with relatively large errors do not affect the autonomous navigation since they can be excluded if there is a pose refine process as described in Sect. IV-D. In Fig. 6c, we mark the height of the detected poster pad with black crosses, if it is detected at the corresponding time. Similarly, in Fig. 6f, we mark the MAV yaw angle estimates when the pad is detected. It shows that the poster pad is detected when the MAV is at different positions and yaw angles.

The poster pad provides a total number of 481 ORB features on all three levels, while the book 69, the mail package 153 and the computer package 97 features. Despite their differences we mentioned above, they can be correctly detected and located. The Root Mean Square Errors (RMSEs) of their 3D position estimation are listed in Tab. I.

D. Autonomous Navigation and Landing Results

In this experiment, we use the poster pad shown in Fig. 5b as the target landing site. Our MAV autonomously navigates in the environment to search for the landing site and finally lands on it. The trajectory of this searching and landing task, as estimated by our onboard vision system, is shown in Fig. 7. The map points from SLAM are triangulated and refined if new keyframes are added. The pose of each keyframe has been depicted as a three red axes in the final map. The searching strategy should depend on the expected complexity of the landing area. In this experiment, a simple rectangular searching path with height $h'_s = 1.2m$ is implemented. The MAV autonomously navigates along this searching path after takeoff and initialization of the visual SLAM. The landing site is detected when the MAV is at the position $\mathbf{P}_1 =$

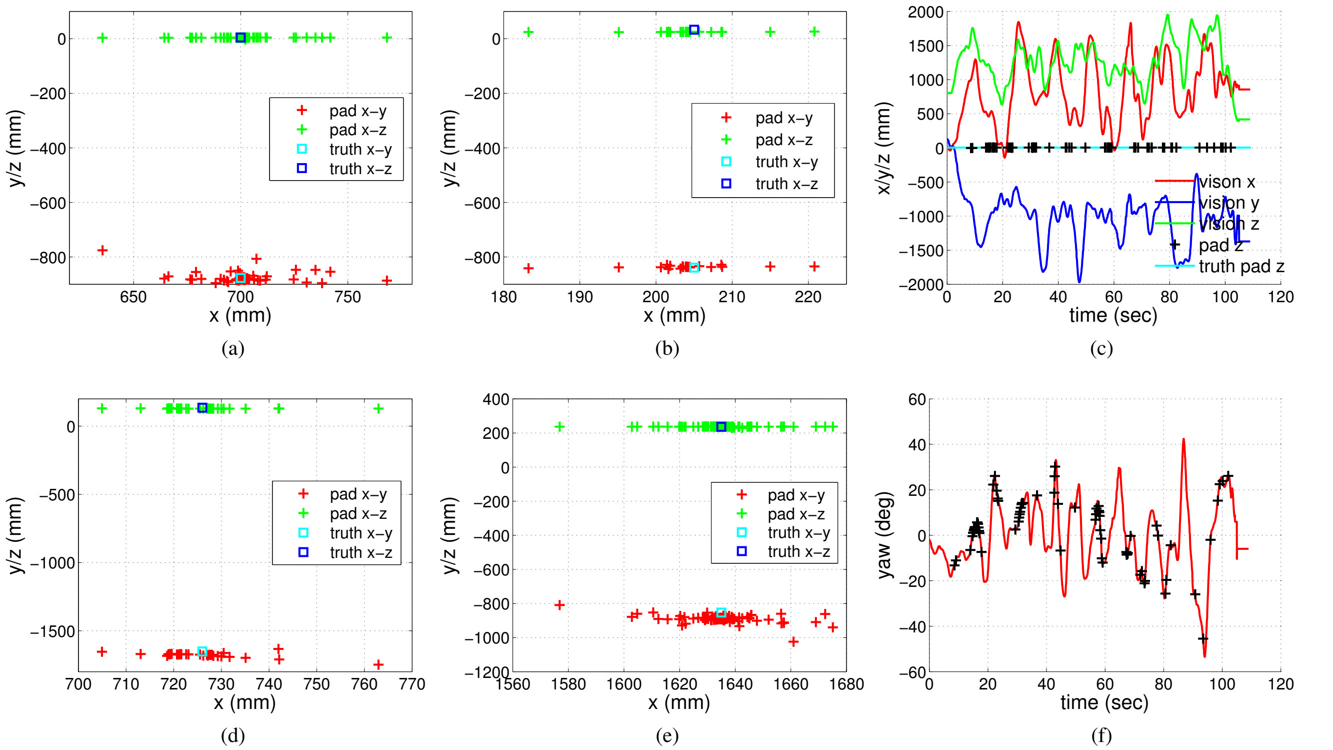


Fig. 6: (a) Position estimates for the poster pad, (b) for the book, (d) for the mail package, and (e) for the PC package. (c) Trajectory of the MAV, and (f) the corresponding yaw angle estimates (a cross is marked if the landing site is detected at the corresponding time).

$(2.001, -1.556, 1.197)^T(m)$, relative to the starting position. When it is at $\mathbf{P}_2 = (2.337, -1.616, 1.201)^T(m)$, landing site detection stops with the computation of the refined landing site pose, which is visualized as a bold color quadrilateral.

Figure 8 shows that the above MAV trajectory fits well with the ground truth data, which proves the accuracy of both the SLAM algorithm and its initialization module. Position $\mathbf{P}_1$ is marked with a blue cross in Fig. 8b , $\mathbf{P}_2$ with a green cross. The initial position estimate of the landing site on the $x-y$ plane is marked with a blue square, and the final refined estimate with a green circle. Both position estimates are close to the ground truth data, which is marked with a black square. The blue and green crosses in Fig. 8a show the initial and final height estimate, comparing with the ground truth height marked with squares. With the landing site size being 500×500 (mm), the initial and final position estimation error is $(-19, -26, -6)^T(mm)$ and $(-11, -27, -5)^T(mm)$, respectively. Table II lists the RMSE of the on-board MAV-pose estimation when compared to the ground truth data.

TABLE II: MAV pose RMSEs of the whole trajectory, with position erros in *mm* and attitude errors in *degrees*

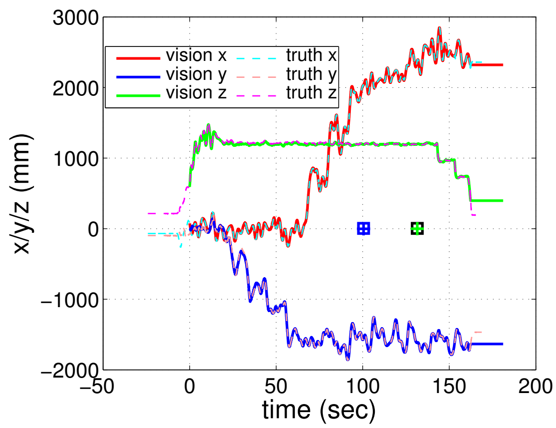
	x	y	z	3D	roll	pitch	yaw
RMSE	8.6	13.6	14.3	21.6	1.04	0.85	1.49

VI. Conclusions and Discussion

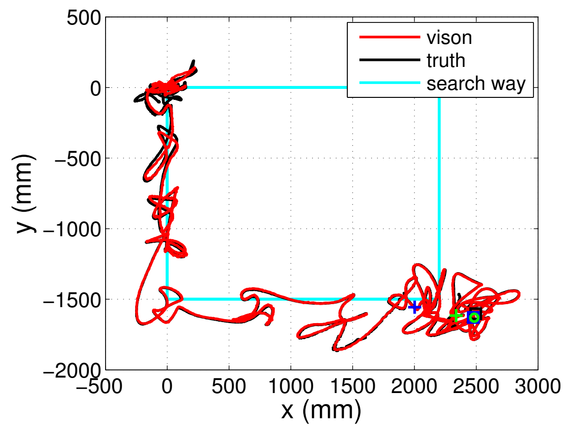
In this paper we have presented a monocular vision system which enables an MAV to navigate autonomously in an unknown environment, and search for the landing site on which it is designated to land. Pose estimates for the control of the MAV are provided by a visual SLAM framework. We solve the landing site detection by integrating a multi-scale ORB feature matching scheme into the mapping thread of the SLAM framework. We take advantage of the map points produced by the SLAM system to accurately estimate the 3D pose of the landing site, using a RANSAC-based method. No absolute scale information of the landing site is needed for its pose estimation.

By evaluating the pose estimation results of different landing sites, we show that our method is flexible and accurate enough for the proposed task of searching for and landing on an arbitrary landing site. Finally, we demonstrate our claim by means of an autonomous navigation and landing flight. Pose estimates of both the landing site and the MAV during this flight have been compared with ground truth data provided by an external tracking system, which shows the high accuracy of our vision system.

For an autonomous landing phase at the end of a long-term mission of an UAV, we propose to fuse IMU data to get its accurate short-term relative pose estimates, which



(a)



(b)

Fig. 8: (a) Position estimates of our vision system on x,y,z axis. (b) MAV trajectory projected to $x-y$ plane. The initial and final position estimates of the landing site and the associated MAV poses are also marked.

can provide a metric scale constraint to initialize PTAM. Thus, autonomous searching for and landing on an arbitrary landing site could be achieved with a similar strategy as proposed in this paper. Although we have achieved promising results within a relatively small area, future work could be fusing IMU data to extend the current monocular visual SLAM system to fulfil large scale tasks. This could not only be used to correct the pose estimates resulting from a monocular SLAM system , but also improve the localization and mapping accuracy of the SLAM system itself.

References

[1] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart, "Onboard imu and monocular vision based control for mavs in unknown in- and outdoor environments", *Proceedings 2011 the IEEE International Conference on Robotics and Automation*, 2011.

[2] M. Bloesch, S. Weiss, D. Scaramuzza, R. Siegwart, "Vision Based MAV Navigation in Unknown and Unstructured Environments", *Proceedings 2010 IEEE International Conference on Robotics and Automation*, Anchorage, AK, pp. 21–28, May 2010.

[3] J.Y. Bouguet, "Camera Calibration Toolbox for Matlab", *http://www.vision.caltech.edu/bouguetj/calib_doc*, 2004.

[4] G. Bradski, "The OpenCV Library", *Dr. Dobb's Journal of Software Tools*, 2000.

[5] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features", *In European Conference on Computer Vision*, 2010.

[6] A. Cesetti, E. Frontoni, A. Mancini, P. Zingaretti, S. Longhi, "A Vision-Based Guidance System for UAV Navigation and Safe Landing using Natural Landmarks", *Journal of Intelligent & Robotic Systems*, 57(1–4), pp. 233–257, 2010.

[7] F. Fraundorfer, L. Heng, D. Honegger, G. Lee, P. Tanskanen, M. Pollefeys, "Vision-Based Autonomous Mapping and Exploration Using a Quadrotor MAV", *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2012.

[8] P. J. Garcia-Pardoa, G. S. Sukhatmeb, J. F. Montgomery, "Towards vision-based safe landing for an autonomous helicopter", *Robotics and Autonomous Systems*, 38(1), pp. 19–29, 2002.

[9] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces", *in International Symposium on Mixed and Augmented Reality*, pp. 225–234, Nov. 2007.

[10] H. Lu, Z. Zheng, "Two novel real-time local visual features for omnidirectional vision", *Pattern Recognition*, 43(12), pp. 3938–3949, December 2010.

[11] D.G. Lowe, "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision*, 60(2), pp. 91–110, 2004.

[12] R. Mahony, V. Kumar, P. Corke, "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor", *Robotics & Automation Magazine, IEEE*, 19(3), pp. 20–32, Sept. 2012.

[13] I. F. Mondragón, P. Campoy, C. Martínez, and M. A. Olivares-Méndez, "3D pose estimation based on planar object tracking for UAVs control", *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 35–41, May, 2010.

[14] L. Meier, P. Tanskanen, L. Heng, G.H. Lee, F. Fraundorfer, M. Pollefeys, "PIXHAWK: a micro aerial vehicle design for autonomous flight using onboard computer vision", *Autonomous Robots*, 33(1–2), pp. 21–39, 2012.

[15] D. Mellinger, N. Michael, V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors", *The International Journal of Robotics Research*, Jan. 2012.

[16] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, "ORB: an efficient alternative to SIFT or SURF", *2011 IEEE International Conference on Computer Vision (ICCV)*, Nov. 2011.

[17] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system", *in Open-source software workshop of the Int. Conf. on Robotics and Automation*, Kobe, Japan, 2009.

[18] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection", *In Proc. 9th European Conference on Computer Vision (ECCV'06)*, Graz,May 2006.

[19] S. Saripalli, J.F. Montgomery, G.S. Sukhatme, "Visually guided landing of an unmanned aerial vehicle ", *IEEE Transactions on Robotics and Automation*, Vol. 19(3), 2003, pp. 371–380.

[20] S.A. Scherer, D. Dube, P. Komma, A. Masselli, A. Zell, "Robust real-time number sign detection on a mobile outdoor robot", *In Proceedings of the 6th European Conference on Mobile Robots (ECMR 2011)*, Orebro, Sweden, September 2011.

[21] S.A. Scherer, D. Dube, A. Zell, "Using Depth in Visual Simultaneous Localisation and Mapping", *in IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5216–5221, 2012.

[22] K. Schauwecker, N. R. Ke, S. A. Scherer, and A. Zell, "Markerless Visual Control of a Quad-Rotor Micro Aerial Vehicle by Means of On-Board Stereo Processing", *in Autonomous Mobile System Conference (AMS)*, Springer, pp. 11–20, 2012.

[23] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-SLAM-Based Navigation for Autonomous Micro Helicopters in GPS-Denied Environments", *Field Robotics*, 28(6), pp. 854–874, 2011.

[24] S. Yang, S.A. Scherer, A. Zell, "An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle", *Journal of Intelligent & Robotic Systems*, 69(1–4), pp. 499–515, 2013.