Martinez-Carranza, J., & Mayol-Cuevas, W. (2015). Indoor MAV Auto-Retrieval Using Fast 6D Relocalisation. Advanced Robotics, 30(2), 119-130. DOI: 10.1080/01691864.2015.1094409

Indoor MAV Auto-Retrieval Using Fast 6D Relocalisation

José Martínez-Carranza[a], Richard Bostock[b], Simon Willcox[b], Ian Cowling[b] and Walterio Mayol-Cuevas[c]

[a]Department of Computer Science,
Insituto Nacional de Astrofísica, Óptica y Electrónica, México
[b]Blue Bear Systems Research Ltd, UK
[c]Department of Computer Science, University of Bristol, UK

September 9, 2015

Abstract

This paper develops and evaluates methods for performing auto-retrieval of a MAV using fast 6D relocalisation from visual features. Auto-retrieval involves a combination of guided operation to direct the vehicle through obstacles using a human pilot and autonomous operation to navigate the vehicle on its return or during re-exploration. This approach is useful in tasks such as industrial inspection and monitoring, and in particular to operate indoors in GPS-denied environments. Our relocalisation methodology contrasts two sources of information: depth data and feature co-visibility but in a novel manner that validates matches before a RANSAC procedure. The result is the ability of performing 6D relocalisation at an average of $50Hz$ on individual maps containing 120K features. The use of feature co-visibility reduces memory footprint as well as removes the need to employ depth data as used in previous work. This paper concludes with an example of an industrial application involving visual monitoring from a MAV aided by autonomous navigation.

1 Introduction

Auto-retrieval is a realistic and useful scenario where a Micro Aerial Vehicle (MAV) is initially manually guided to a location or locations of interest by a user and subsequently recalled to its starting point. The idea is that the vehicle will autonomously negotiate its return based on the taught path. This is an interesting mixture of supervised and unsupervised operation that is useful in

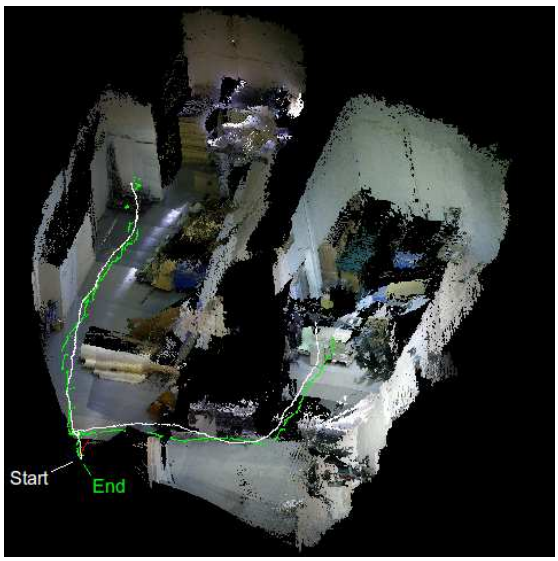


Figure 1: The *taught* trajectory during a first exploration of a MAV (in white). The *repeated* trajectory during re-exploration (in green) is estimated using visual odometry with periodical calls to our fast 6D relocalisation, thus enabling drift correction. A 3D reconstruction of the environment was also produced for this flight using the camera pose estimates and depth information captured during exploration.

tasks such as inspection and monitoring and which frees operators to concentrate on other tasks, once a safe path to reach a location of interest has been established.

Although auto-retrieval is notionally simpler than fully autonomous exploration, it still faces important challenges. Primarily, that of determining where the vehicle is in 6D space accurately, at fast speed and robustly as well as being able to relocalise in an already visited place, which is a well known problem in SLAM. For instance, relocalisation may be used to recover from temporal loss of tracking (e.g. in visual odometry), and used to correct drift in the estimation (loop closure). On the other hand, if relocalisation is achievable at high frame rate, it can be triggered alongside the normal visual tracking, which can help if the vehicle starts to drift w.r.t. the taught path, thus enabling path correction in earnest.

Motivated by the above, in this paper we develop and expand a relocalisation methodology around the notion of 3D feature neighbourhoods as explained in Sec 3. This not only enables fast operation, but it also allows the use of faster yet more ambiguous binary descriptors. Our method starts from a real-time visual odometry approach to collect a 3D map of visual features. In our estimates, the error of this system is of less than 1% over trajectories of about 100 meters. The generated trajectory provides spatial way points to be used for the auto-retrieval

task.

For fast relocalisation, our method exploits the 3D geometric information recorded during the mapping to accelerate the search of 2D-3D matches, a common bottle neck in full pose relocalisation, which is only aggravated by ambiguous features. We use 3D neighbourhood information either provided by a depth measurement (from an RGB-D sensor for instance), or by the spatial 3D distribution of co-visible map points, which does not require a depth reading. Our experiments indicate that the way in which we exploit these neighbourhoods is effective for achieving good pose estimation in full 3D space as well as it is fast enough for on-line control in our MAV application.

Fig. 1 shows an example of the algorithm working, with the *taught* path in white and the *repeated* path in green, as a MAV explores a warehouse of about 14×12 meters, with a traverse path of about 55 meters. The 3D reconstruction of the scene is also performed in real time. The average distance in between the taught and repeated path is of 50 cm.

This paper is organised as follows: section 2 discusses the state of the art related to our work; section 3 describes our fast visual relocalisation methodology; section 4 briefly describes the two aerial platforms used in this work; section 5 presents the results of our experiments that aim to assess each one of the components in our work, and finally in section 6 we discuss our final remarks.

2 Related Work

2.1 Visual Mapping from Micro Aerial Vehicles

There have been various recent works on visual mapping for MAVs. In most cases, and in part due to the limited battery life, these works consider small excursions but still constitute good examples of systems able to operate in GPS-denied environments at low altitude.

Perhaps the simplest strategy to integrate vision on MAVs has been the use of optic flow measurements based on the concept of motion parallax. These approaches provide a temporary relative map between the vehicle and the world, which is able to discern nearby and far away obstacles [1]. But the use of optic flow is insufficient for long term trajectory estimation. As an alternative approach, if known markers are available and located in the environment, these can also provide a way to position the MAV relative to the environment [2]. For more generic operation, a map-based approach that uses re-detectable natural features is preferred.

In some cases, mapping frameworks based on bundle-adjustment SLAM such as PTAM [3] have been used to showcase small space autonomous operation over a few square meters, such as hovering and holding position with one of the earliest examples being [4], other works include those of [5, 6]. Similar frameworks have been used for multiple MAV exploration towards a textured environment observed from a a few dozen meters away [6, 7], and using offline 3D reconstruction of the scene. Other approaches have combined stereo vision

with Lidar scanning along the horizontal axis [8]; others have proposed to carry out offline processing of the SLAM for a more accurate map generation [9]. Recently, in [10] it has been presented a semi-dense visual odometry system running at 55 Hz onboard a MAV, which enables way-point-based autonomous navigation.

All of the approaches mentioned before strongly motivate our work in aiming for autonomously navigate in a GPS-denied environment by means of a visual system running at high frame-rates. But in addition, our goal is to enhance robust operation via fast visual relocalisation.

2.2 Visual Relocalisation

Since the randomised-trees-based solution presented in [11], more scalable approaches based on feature descriptors have been proposed. For instance, in [12, 13] where hash tables are used as organisation model. In [3], sub-sampled images of key frames and their poses are stored in a database and thus used to recover from loss of tracking. More advanced approaches following the idea of Bag of Words (BoW) have been proposed, for instance in [14, 15, 16].

Most approaches for relocalisation in MAVs use one of two strategies, either keyframe-based [4, 5, 6] or using dense visual descriptors such as SIFT or SURF [9, 17]. However, whatever approach is followed for relocalisation, if this involves the use of dense descriptors to represent the observed scene, either BoW-like histograms, or visual descriptors, then two possible drawbacks may be faced: i) comparison of float-type vectors may become expensive in proportion to the number of vectors to be compared against; ii) larger memory footprint may be demanded to store such vectors for large maps. Both of these issues may affect a platform with limited computational and storage budget.

2.3 Teach and Repeat

Our work shares similarities with the visual *teach and repeat* tasks on ground vehicles [18, 19, 20]. Where a *teach* stage visual odometry is used to build a data base of visual descriptors associated to 3D points. During the *repeat* stage, visual odometry is used in combination with periodical relocalisation calls in order to maintain a similar global path accuracy [18]. Recently, a basic proof of concept for visual teach and repeat on a quadcopter has been presented in [21]. However, all the heavy computation such as the visual mapping and relocalisation are carried off board, in contrast to our approach where all the processing is done on-board our vehicles.

3 Our 6D Relocalisation System

Our method follows two main components found in similar systems such as in [22, 23, 24, 25]. These ideas are: (1) using binary descriptors for fast comparison of descriptors and low memory foot print when stored in a database; (2) efficient

organisation model of the database by utilising efficient hashing techniques, or by using hierarchical trees. However, the main contribution in our approach is the incorporation of 3D information, obtained via an RGB-D sensor or from the map itself, within the relocalisation pipeline and how this is exploited in order to accelerate the search for 2D-3D matches.

In summary, recovered binary features and their estimated 3D positions at relocalisation time (2D-3D matches) are stored and retrieved from hash tables. Our method then uses neighbourhood 3D information and performs pruning of potential false positive matches, which occur very frenquently due to the use of ambiguous (albeit fast) descriptors, leaving only a compact set of matches which are deemed to be high quality. These are passed to a RANSAC procedure for full pose estimation. The main advantage of using this approach is the reduction in the number of RANSAC iterations, which greatly reduces the relocalisation processing time.

3.1 Relocalisation Using Binary Descriptors

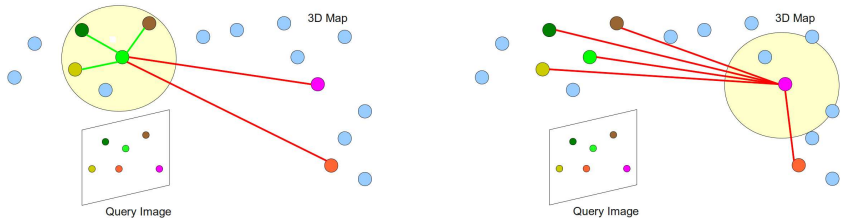
During the mapping stage, each mapped visual feature is associated to a binary descriptor centred at the salient point where the visual feature was initialised. We use ORB descriptors [26] with length of 512 bits. These are organised by using a Locality-Sensitive-Hashing (LSH) technique [27], which enables us to store descriptors in hash tables at almost no cost and with no restriction to online increase of the database[1]. The 3D world position of the salient point is available in the map and therefore can be attached to its binary descriptor. This means that whenever a binary descriptor is retrieved from any of the hash tables, we will have access to its 3D position as well.

Therefore, at relocalisation time, given a query image at some time step, salient points in the image are extracted using the FAST corner detector [28]. For each salient point $\mathbf{s}_i$ the corresponding ORB descriptor $\mathbf{b}_i$ is extracted. Each binary descriptor $\mathbf{b}_i$ is passed to the hash function, which simply takes subsets of bits from the binary number in order to access to the corresponding bins in the hash tables (where potential matches for $\mathbf{b}_i$ exist). A linear nearest neighbour search is performed with all the retrieved binary descriptors from the bins searching for the descriptor that minimizes the Hamming distance with $\mathbf{b}_i$.

Let $\mathbf{b}_{i_m}$ be the best match for $\mathbf{b}_i$ and let $\mathbf{p}_{i_m}$ be the 3D position of the best binary match $\mathbf{b}_{i_m}$. Then a set of 2D-3D pairs is augmented as follows: $\mathbf{C} = \mathbf{C} \cup \{(\mathbf{s}_i, \mathbf{p}_{i_m})\}$ and if $|\mathbf{C}| > c_{size}$ [2] then it is passed to the pose estimation module, which is based on a three-point pose estimator plus RANSAC [29]. The relocalisation is considered successful if the pose estimator finds a minimum set of inliers in $\mathbf{C}$ such that these can be used to estimate a camera pose.

[1] Note that LSH is a simple but still powerful organisation model. Although any other organisation model could be used in our approach, our main point in this work is to demonstrate the advantage of using 3D information for pruning of potential outlier matches.

[2] We carried out several experiments with different video sequences in order to empirically choose a value for c_{size} [24]. In this work, we use $c_{size} = 15$, which offered a good trade-off in terms of relocalisation percentage and processing time.



(a) (b)

Figure 2: Ruling out spurious matches using the concept of 3D neighbourhood: (a) after 2D-3D matches have been retrieved from the hash tables (a 2D position on the query image has been coloured according to its corresponding 3D match), a 3D neighbourhood centred around the 3D point in light green helps to identify those 3D matches around it that fall inside and outside the neighbourhood, suggesting that the light green match is likely to be an inlier as more 2D-3D matches fall inside than outside; (b) when the neighbourhood is tested on the magenta point we find out that none of the matches' 3D positions fall inside, hence, this match is likely to be a false positive.

3.2 Enhanced Relocalisation Using a 3D Neighbourhood

A known problem in the retrieval of 2D-3D matches with procedures like the above, is that of retrieving false positives. This commonly is addressed with a RANSAC-like procedure to identify inliers. However, a large number of RANSAC iterations would be required, which ultimately consumes processing time. Motivated by this, we have been investigating possible ways in which the spurious outliers can be detected and removed from the set before such set is sent to RANSAC. .

In this section we describe an algorithm based on the concept of 3D neighbourhood of map points. The intuition behind this is that true positive 2D-3D matches should be spatially close to each other in a 3D neighbourhood, and although it may not be the general case, many false positive matches may fall far away from such neighbourhood.

Fig. 2 illustrates the neighbourhood concept. After having retrieved a set of putative 2D-3D matches, we can select the 3D position of one of those matches which we can now refer to as a *pivot*. We then proceed to define a neighbourhood around it within a set radius. This neighbourhood helps to identify how many 2D-3D matches fall inside such 3D area. This is an indication of whether the pivot is likely to be an inlier or not. That is, if many matches fall inside this region such matches are more likely to correspond to the same location and hence the pivot is likely to be a true positive. This simple notion results in good improvement in terms of both speed and accuracy of pose estimation.

We exploit the idea mentioned above in order to implement an algorithm that monotonically attempts to construct a chain of quality matches. For this purpose, similar to section 3.1, first we build a set $\mathbf{M}$ containing the retrieve set

of 2D-3D matches. Then we initialise the quality set $\mathbf{C}$ with a match selected (and removed) from $\mathbf{M}$, this first match in $\mathbf{C}$ will be c_1. Likewise, we select another match $m_i \in \mathbf{M}$ for which we set a radius $\mathbf{r}_i$ and test whether the 3D Euclidean distance in between c_1 and m_i, which we call d_{i1}, falls within this radius (next section will explain how this radius may be set). If d_{i1} is less than r_i then m_i is added to $\mathbf{C}$. For a second match m_j in $\mathbf{M}$, and its respective radius r_j, the test is repeated. This time two distances d_{j1} and d_{j2} are calculated as $\mathbf{C}$ contains two elements. Thus, m_j will be rejected if either $d_{j1} > r_j$ or $d_{j2} > r_j$ since we want m_j to be a match whose neighbourhood includes all the accepted matches in $\mathbf{C}$ so far. This process continues by testing all the remaining matches in $\mathbf{M}$ and, in the end, $\mathbf{C}$ will contain a set of matches whose spatial 3D distribution is likely to correspond to inlier 2D-3D matches.

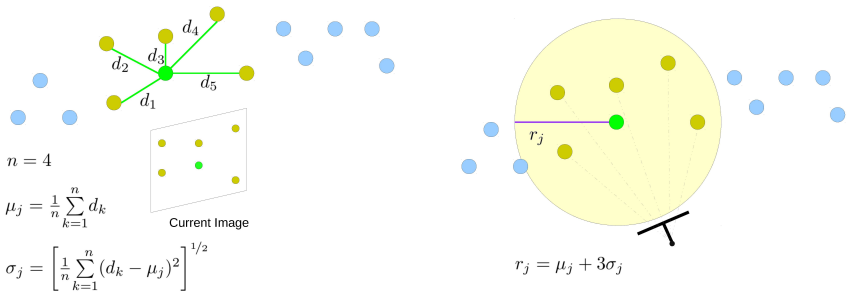
Note that the construction of $\mathbf{C}$ depends on the first element which ultimately may lead to a small or null set. Therefore, a further improvement is that of testing each putative match as the first element in $\mathbf{C}$ and see if a set can be constructed. Thus, the set with the largest number of elements will be selected as the final output set. This procedure is described in more detail in Algorithm 1 of this paper. The following sections describe two specific manners in which the neighbourhood radius r_j can be tailored for each member $m_j \in \mathbf{M}$.

3.3 Using Depth Information to Define Neighbourhoods

We call this approach 3D test or 3DT in the results. It follows previous work [24, 30] where we explored a particular case of the above algorithm that uses depth information whenever available, for instance, from an RGB-D sensor. Using a depth image, it is possible to estimate the 3D distance in between two matches, and compare that to the distance in between the 3D map positions associated to the matches. In essence, $r_j = d^s_{jk} + \epsilon_d$, where d^s_{jk} is the 3D distance, calculated with the depth data from the sensor, in between a match $m_j \in \mathbf{M}$ and a match $c_k \in \mathbf{C}$, and ϵ_d is a depth error associated to the sensor.

This is a test of whether the 3D position of the matches falls within the same neighbourhood, except that the neighbourhood can be defined by means of the depth information. Our experience was two fold: (1) it is possible to quickly prune likely false positive matches by using this 3D Test; (2) the surviving set of matches are high quality matches from which it is likely to relocalise a valid camera pose.

We should highlight that when using depth data, our approach is different from other approaches where RGB-D imagery is used to relocalise the camera pose, such as those in [31, 32] where depth data is used to build a model that relates pixels to camera poses. In contrast, our approach is much simpler in the sense that depth, or any available 3D neighbourhood information, is used to assess the validity of a 2D-3D match, and such assessment is computationally cheap given that only Euclidean distances are computed. Therefore in our approach, 3D information is not used to describe the scene but to validate the descriptors retrieved during the matching stage within the relocalisation pipeline.



(a) (b)

Figure 3: Covisible 3D points are used to defined the radius r_j used in the *algorithm 1* of this work: (a) first, we calculate the mean μ_j and standard deviation σ_j of the Euclidean distances in between co-visible 3D points and the candidate 3D point to be stored in the database (green point); (b) top view illustrating the use of the mean and standard deviation of the distances in order to define a neighbourhood region around the candidate point.

3.4 Using Co-visibilitiy to Define the Neighbourhood

The above instantiation of the algorithm relies on depth data directly measured from the RGB-D sensor and this, unfortunately, can be unavailable, for instance, in outdoors or due to reflective or light-saturated regions. This motivated us to consider alternative ways to define the neighbourhood radius, especially when depth information is not available such as in monocular systems.

Here we explore the idea of using feature co-visibility in order to define the neighbourhood radius. Two visual features are co-visible if these are being observed in the same frame or number of frames, see Fig. 3. This relationship is useful to guide a search where different combinations of features may arise, however, most likely valid combinations are those where features are correlated with each other, and co-visibility is a strong correlation. For instance, in [11] only co-visible matches are used within the RANSAC procedure. In [33], co-visible features are used to train the Conditional Random Field used for place recognition. More recently, [34] constructs graphs of co-visible maps (where features are co-visible) in order to estimate a location for a query image.

Any of the methods mentioned above implies the storage or production of graph models or databases, where the co-visibility relationship is recorded. However, our aim is not to store every single co-visibility relationship, but to use this information to tailor the radius $\mathbf{r}_j$ for every binary descriptor mapped and stored in the hash tables.

Therefore, during mapping, for a given visual feature j we calculate the 3D Euclidean distances in between this and its co-visible features, this is, all the visual features being tracked in the current frame, see Fig. 3a. This computation is performed only when the system has decided that the visual feature is valid and hence, worth it of being stored in the hash tables. This decision is based on

the number of frames that the feature has been successfully observed/tracked. In our system this number is set to be of at least 20 frames. In order to store the feature and compute the required radius in an efficient way, we store and do computations for only one features per frame, this takes $0.24ms$ in average.

From all these distances we compute the mean μ_j and its standard deviation σ_j, thus, the corresponding neighbourhood radius is set to be $r_j = \mu_j + 3\sigma_j$, see Fig. 3b. Note that the constant 3 indicates that the radius r_j will cover 99 % of the distribution of distances calculated from the co-visible features. For further reference in the paper, this co-visibility test is named as **CVT** in the results section.

4 Aerial Platforms

Flight testing was carried out using two different MAVs. Fig. 4a shows the first aerial platform, a quadcopter with transversal dimensions of 66×66 cm, and an integrated RGB-D sensor (ASUS Xtion Pro Live), with resolution of 320×240 pixels, mounted underneath its base. A dedicated vision processor was used on-board this platform to run our visual odometry and relocalisation methods. This processor is a board computer with Intel Core2Duo 2.16GHz processor and 2GB RAM.

Fig. 4b shows the second platform, a hexacopter, with dimensions of 118×118 cm. On this platform, an Odroid U2 computer, with ARM QuadCore 1.7GHz processor and 2GB RAM, was used to run the vision algorithms. The vehicle also carries two cameras: an RGB camera with resolution of 1920×1088 pixels, this camera is used for visual monitoring of the scene and we will refer to this as the *monitoring* camera; the second camera is a stereo camera used for the visual odometry and relocalisation algorithms. Both cameras are located in front of the vehicle, with the stereo rig having an inclination of 15 degrees and being located underneath the RGB camera. The stereo camera was built with two USB cameras IDS UI-1221LE-C-HQ with resolution of 376×240 pixels.

In both platforms, a miniature flight controller (Blue Bear's SNAP autopilot running on a second dedicated board) was used for manual flight and autonomous navigation. All processing was carried out in real time on board the vehicle, with state estimates from the visual navigation module sent to the autopilot for input into flight control algorithms. High-rate IMU and compass heading data was used to stabilise the vehicle attitude, with data from the visual navigation used to control the platform position.

5 Experiments

For each of the experiments in this section our first step is to build a map of 3D points by using a visual odometry system similar to that in [35]. Salient points are detected using FAST (we used up to 1000 points) and template patches are used for feature tracking, but for every initialised feature we extracted its

(a) (b)

Figure 4: The two aerial vehicles used in this work: (a) quadcopter with an RGB-D camera attached underneath its base, this platform was used to test the auto-retrieval concept; (b) hexacopter platform used in our visual monitoring application, with a *monitoring* camera on the top and a stereo camera underneath,

binary descriptor and stored it in the hash tables. Features were initialised with an inverse depth parametrization [36] but the depth obtained from an RGB-D sensor (ASUS X-tion pro live) as initial state, which led to fast convergence. In a similar manner, for our visual monitoring application and as commented before in section 4, features are initialised with the depth estimated with our sparse stereo system. In either case, we use 8 hash tables for our database of binary descriptors. The mean and standard deviation of 3D distances between co-visible mapped features are estimated during this mapping stage [3].

5.1 Fast 6D Visual Relocalisation

We investigated the potential of using the 3D neighbourhood test in a scenario containing a mixture of challenging conditions such as poor texture, change of light conditions and scarce depth information. A first video sequence of the scene, containing 6704 RGB-D images with resolution of 320×240 captured at $30Hz$, was used to build the 3D map. The trajectory followed by the camera is shown in black in figure 5, the camera travelled 215 metres, moving forward at a speed of 1 meter per second. A map of approximately 120,000 descriptors was built and stored in the hash tables.

A second video sequence of 3220 images was utilised to test the different configurations of the 6D relocalisation framework used in this work, hence, the relocalisation is tested on every single frame of the sequence. In this sequence, the camera followed the same trajectory as in the first sequence but at a speed of about 2 metres per second. These experiments were run in an Intel i5 Core 2.4GHz processor with 4GB in RAM.

We assessed four configurations of the 6D relocalisation frame work:

[3] A video of the real-time experiments presented in this section can be found at: *http://youtu.be/5uSEDmMH4rg*

- **BD**: Best 2D-3D matches are retrieved from hash tables and then the three-point pose plus RANSAC is used for 6D pose estimation.
- **BD+3DT**: Best 2D-3D matches are retrieved from hash tables, then the neighbourhood test is applied, where the neighbourhood radius r_j is set with the help of the depth d^s_{jk}, obtained from the depth sensor, and the depth error ϵ_d.
- **BD+AT**: We set an arbitrary value to the neighbourhood radius r_j.
- **BD+CVT**: For every j^{th} feature stored in the hash tables, the radius r_j is set with mean μ_j and the standard deviation σ_j obtained from its co-visible mapped features as explained in section 3.4.

From our previous work in [30, 24] we have experimentally set the following values: (1) within the RANSAC procedure, a match is accepted as inlier if the pixel difference in between the pixel coordinate, where the match was found, and the projection pixel coordinate of the corresponding 3D point is of less than 2 pixels; (2) regarding r_j for **BD+3DT**, the depth error was set to be $\epsilon_d = 20cm$.

Table 1 shows the results obtained for each of the above configurations. For this kind of framework, the error in the relocalised pose is usually within centimetres [30], however, from time to time a pose falls far away from the true pose. Such poses are not considered as relocalised, therefore, the percentage of relocalisation includes only those poses within an error of +/-25 cm. The table shows the number of inliers set for each configuration as much as the number of RANSAC iterations. These values were the ones that produced the best results for each configuration. Note that for **BD**, the number of minimum inliers and the RANSAC iterations is higher than those of the other methods, but unfortunately using lower values for **BD** in RANSAC showed very noisy output with several incorrect relocalised camera poses.

Results from table 1 indicate that the 3D neighbourhood test, in any of its variants, enhances the baseline system, not only by increasing the relocalisation percentage, but also by reducing the computational time. Note that for the three configurations of the test the processing time is about $0.2ms$ (indicated as **NB Test** in the table). Additionally, for these configurations less RANSAC iterations are used since we assumed that RANSAC will receive a set of quality matches from the test. However, we should mention that the relocalisation percentage for **BD+3DT** is not as good as in previous experiments where we have obtained around 70% of relocalisation [30, 24]. This is mostly due to the poor quality of depth information in several sections of the sequence (see figure 5).

In contrast, **BD+AT**, where we set the radius $r_j = 5$ metres, produced much better results than when using depth (**BD+3DT**). We should say that for this scene this radius was very adequate, however, we have tested the same threshold in other sequences and the relocalisation percentage drops, meaning that we should adapt this threshold according to the spatial distribution of the scene. On the other hand, without having to manually set any threshold, **BD+CVT**

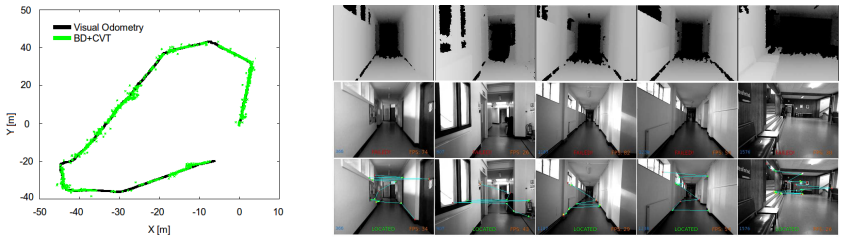


Figure 5: Left image: top view where the visual odometry trajectory is shown in black, the relocalised camera poses obtained with the **BD+CVT** configuration have been overlaid in green. Right image: we noted that **BD+3DT** (second row) fails to relocalise even when apparently there seems to be enough depth data (first row), however, the images lack texture and the depth data of those places where saliency exists is either null or of poor quality. In contrast, **BD+CVT** overcomes these problems in each one of these examples (third row) since the method no longer depends on the use of depth data in order to construct a set of quality matches; the blue lines indicate the set of matches found by the method, detected inliers are set in green, whilst outliers are in red.

outperforms any of the other methods, implying that the neighbourhood radius set by the co-visibility statistics produces a better set of quality matches. The relocalised camera poses for this method are shown in Fig. 5, note that the system manages to achieve 51% of relocalisation. We deem this to be an acceptable outcome since the relocalised poses cover almost all of the traversed space. We also observed that many of the places where the method failed to relocalised were those with very low texture or strong change in light conditions. A different feature would be interesting to investigate here.

5.2 Auto-Retrieval Flight

The auto-retrieval concept enables the demonstration of closed loop flight control independent of path-planning and exploration aspects required for completely autonomous flight. It incorporates manual exploration, in which visual odometry is used for 6D pose estimation, followed by automatic return of the vehicle back to its starting point. In both stages, our fast relocalisation framework is utilised. Our visual odometry during both modes, manual flying and auto-retrieval, ran at 20 Hz.

On the *teach* mode, relocalisation is only triggered if tracking fails, however, for the *repeat* mode, our strategy exploits the fact that the vehicle is expected to repeat the taught trajectory by periodically triggering relocalisation (every 200 frames), which has the effect of reducing drift in the estimated trajectory and keeping the return flight path close to the original one. For this auto-retrieval concept, the platform travels backwards to allow it to re-observe the scene.

During the *repeat* mode, any relocalised pose beyond a threshold is consid-

ered spurious or *false positive* and hence ignored as if the system has not managed to relocalise. On the other hand, in the event of a *false negative* outcome, this is, the relocalisation failed to recover a valid pose, the visual odometry simply will continue operating, however, the relocalisation will be attempted again in the next available frame.

Under manual exploration mode the flight controller stabilises vehicle attitude and responds to attitude (pitch/roll), yaw rate and throttle demands from the pilot. During exploration mode the outputs from the visual odometry are logged at 4Hz, effectively generating a series of closely-spaced way points to be followed during auto-retrieval. When the system switches to autonomous mode the position controller commands the vehicle to return along the original flight path, visiting the sequence of recorded way points in reverse order using the estimates from our combined pose estimation approach for platform position.

We report results of three representative auto-retrieval runs. In all the runs the vehicle's start position is with the camera looking at the left wall in the venue. After taking off, the pilot commands the vehicle to perform a 90^o rotation to the right. After the rotation, the vehicle follows whatever trajectory the pilot commands, with the camera always pointing forward. At some point of the manual flight, the pilot pushes a button to activate the auto-retrieval mode. In this mode, the vehicle attempts a return flight trying to follow the path traversed during the manual flight. As mentioned before, the auto-retrieval uses the pose estimation obtained from our combined approach in order to autonomously direct its position towards the next way point. When the vehicle has reached the last way point, it correctly performs a -90^o rotation, which is an indication of the vehicle arriving at the starting point, after this, for safety reasons, the pilot takes over and lands the vehicle manually.

Fig. 6a-c show the vehicle's trajectory (camera pose estimates) for each one of the runs in a top view only. However, our goal is to observe that the vehicle repeats the same taught path, thus returning to the starting point. The taught trajectory estimated by the visual odometry system is depicted in red. In the same figure, the estimated vehicle's position during auto-retrieval (*repeat*) is depicted in green.

5.3 Visual Monitoring Application

We developed a demonstrator application supported by the auto-retrieval concept. For this we used the vehicle with the stereo sensor, however, stereo matching was called occasionally to calculate sparse triangulation of image correspondences. These correspondences were obtained through ORB-based feature matching. ORB features were extracted only on salient points detected with the FAST corner detector.

Sporadic stereo matching was run in a separate thread at an average of 5 Hz. In this sense, the visual odometry and the relocalisation algorithms processed images acquired with the left camera only, thus performing effectively as a monocular system for most of the time as stereo is only used when features are initialised, hence, we used the co-visibility method for relocalisation. Re-

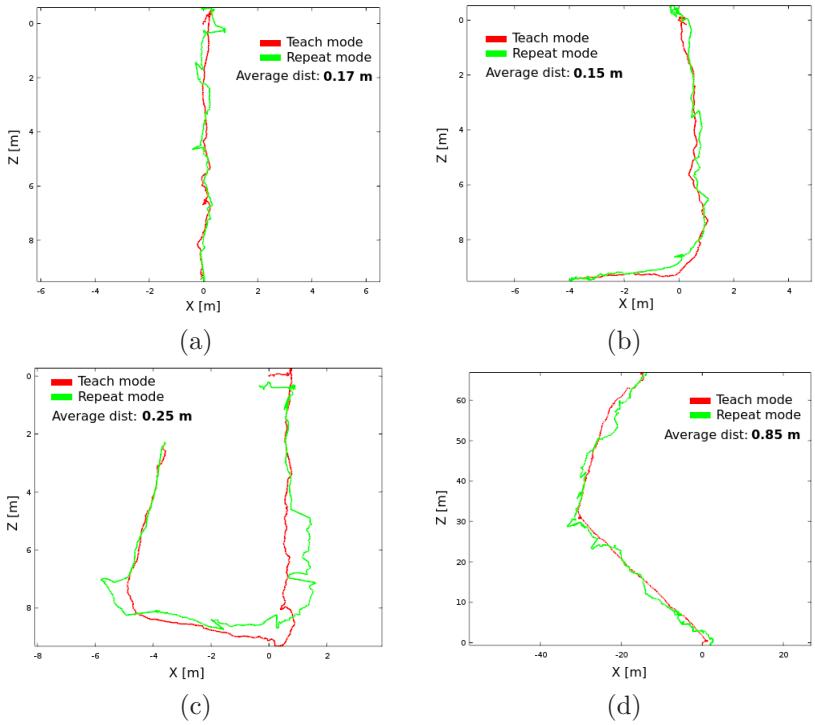


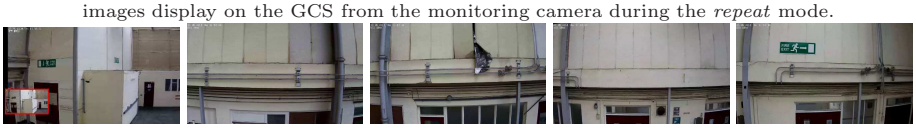
Figure 6: Top view of the vehicle's pose estimates obtained during manual fight (*teach*) and *repeat* mode: in (a-c) results related to the auto-retrieval concept; (d) results related to the visual monitoring application, note that the vehicle travelled a total of 70 metres. The average distance in between the taught and repeated trajectories is also indicated for each run.

localisation was called every 200 frames. We optimized our visual algorithms substantially for the Odroid, and we were able to work at an average of 40 Hz.

Before testing our visual monitoring application, two video sequences of the target scenario were recorded using the stereo camera on board the hexacopter. For both video sequences, the vehicle was manually flown following the expected trajectory for the *teach* and *repeat* mode, about $12K$ images were recorded for each sequence. The first sequence was used to create a map of about $16.5K$ descriptors. The second sequence was used to test our relocalisation for every frame using this map. We ran this experiment on the same i5 processor used in the experiments of section 5.1. A relocalisation of 73% was obtained with an average frame rate of $55Hz$.

For the application, in the *teach* mode the MAV is flown by a pilot following a specific trajectory. At the outset, the vehicle is placed with the cameras facing the wall to be monitored. After taking off, the pilot controls the vehicle so that it moves sideways following a trajectory parallel to the wall, then a turn is

images display on the GCS from the monitoring camera during the *repeat* mode.



Amplified images corresponding to those shown in the red rectangle.

Figure 7: Snapshots illustrating the outcome of our visual monitoring application during the *repeat* mode. The first row shows the monitoring camera video displayed on the GCS where the inset images in the red rectangle correspond to the retrieved images saved during the *teach* mode.

performed to face the next perpendicular wall and it continues with the sideways fly. A Ground Control Station (GCS) running on a laptop is available for the pilot to observe a top view of the vehicle's position and the live video from the monitoring RGB camera, which is transmitted via telemetry. The pilot serves as an expert to indicate locations of interest, which are recorded by a press of a button. This results in images being captured by the monitoring camera with their corresponding vehicle's position. A set of way points describing the trajectory is also stored.

In the *repeat* mode, the vehicle is placed on the same starting position as in the *teach* mode. The pilot is in control during taking off, however, once on the air, the pilot switches to autonomous navigation after which (similar to the auto-retrieval task) the vehicle flies autonomously aiming at repeating the taught path. During the autonomous flight, the vehicle's position is used to retrieve the stored images from the monitoring camera from the *teach* flight. The system automatically retrieves images whose recorded (vehicle's) position is within a radius of 3 metres w.r.t. the current vehicle's position. The image with the actual closest position is displayed as an inset image on the display showing the live video from the monitoring camera, see Fig. 7. The idea is that the user of the ground station can compare images from the taught and the latest observed images for an area, perhaps to identify changes on the scene. The estimated paths followed by the vehicle are shown in Fig. 6d.

6 Conclusions

In this work we have shown that, by using fast binary descriptors together with an efficient relocalisation methodology, it is possible to develop real-time onboard autonomous operation of micro air vehicles in tasks such as navigation and auto-retrieval.

We have also presented an extension of a validation test for ruling out false positive matches and based on the concept of 3D neighbourhood, however, this extension removes the need for depth data which was a key component in our

previous work. In this context, we have introduced a simple, yet powerful and efficient way of using 3D spatial information of co-visible features in order to define such neighbourhoods. Results indicate that the co-visible information leverages the performance of our relocalisation system whilst maintaining the high frame rate with an average of $50Hz$ on an i5 processor.

The complete system has been effectively tested on two different MAVs, one used to demonstrate the auto-retrieval concept, and another for a simple visual monitoring application. Given the processing limitations of the computers on board, it is expected for our relocalisation method to run slower, however, the speed is reduced by a half only, which was good enough to maintain a performance of $20Hz$ on the quadcopter, and $40Hz$ on the hexacopter, during the *repeat* mode.

In our future work we are aiming at testing our system during longer flights and outdoors.

References

[1] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart., "Mav navigation through indoor corridors using optical flow," in *International Conference on Robotics and Automation*, 2010.

[2] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Pixhawk: A system for autonomous flight using onboard computer vision," in *International Conference on Robotics and Automation*, 2011.

[3] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *International Symposium on Mixed Augmented Reality*, 2007.

[4] M. Bloesch, S. Weiss, D. Scaramuzza, and R. Siegwart., "Vision based mav navigation in unknown and unstructured environments," in *International Conference on Robotics and Automation*, 2010.

[5] L. Kneip, M. Chli, and R. Siegwart, "Robust real-time visual odometry with a single camera and an imu," in *British Machine Vision Conference*, 2011.

[6] M. W. Achtelik, M. C. Achtelik, M. Chli, S. Chatzichristofis, F. Fraundorfer, K.-M. Doth, L. Kneip, D. Gurdan, L. Heng, E. Kosmatopoulos, L. Doitsidis, G. H. Lee, S. Lynen, A. Martinelli, L. Meier, M. Pollefeys, A. Renzaglia, D. Scaramuzza, R. Siegwart, J. C. Stumpf, P. Tanskanen, C. Troiani, and S. Weiss., "sfly:swarm of micro flying robots," in *International Conference on Intelligent Robots and Systems*, 2012.

[7] Sfly-Project. http://www.sfly.ethz.ch.

[8] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Stereo vision and laser odometry for autonomous helicopters in gps-denied indoor environments," in *SPIE Defense, Security, and Sensing*, pp. 733219–733219, International Society for Optics and Photonics, 2009.

[9] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a quadrotor mav," in *International Conference on Intelligent Robots and Systems*, 2012.

[10] D. S. Christian Forster, Matia Pizzoli, "Fast semi-direct monocular visual odometry," in *International Conference on Robotics and Automation*, 2014.

[11] B.Williams, G.Klein, and I.Reid, "Real-time slam relocalisation," in *International Conference on Computer Vision*, 2007.

[12] D. Chekhlov, W. Mayol-cuevas, and A. Calway, "Appearance based indexing for relocalisation in real-time visual slam," in *British Machine Vision Conference*, 2008.

[13] T. Sattler, B. Leibe, and L. Kobbelt, "Fast image-based localization using direct 2d-to-3d matching," in *International Conference on Computer Vision*, pp. 667–674, Nov 2011.

[14] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof, "From structure-from-motion point clouds to fast location recognition," in *International Conference on Computer Vision and Pattern Recognition*, pp. 2599–2606, June 2009.

[15] M. Cummins and P. Newman, "Appearance-only slam at large scale with fab-map 2.0," *International Journal of Robotics Research*, 2011.

[16] G. Baatz, K. Kser, D. Chen, R. Grzeszczuk, and M. Pollefeys, "Leveraging 3d city models for rotation invariant place-of-interest recognition," *International Journal of Computer Vision*, vol. 96, no. 3, pp. 315–334, 2012.

[17] H. Lim, S. N. Sinha, M. F. Cohen, and M. Uyttendaele, "Real-time image-based 6-dof localization in large-scale environments," in *International Conference on Computer Vision and Pattern Recogntion*, 2012.

[18] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *J. Field Robot.*, vol. 27, pp. 534–560, Sept. 2010.

[19] W. Churchill and P. Newman, "Experience-based navigation for long-term localisation," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1645–1661, 2013.

[20] C. McManus, P. Furgale, B. Stenning, and T. D. Barfoot, "Lighting-invariant visual teach and repeat using appearance-based lidar," *Journal of Field Robotics*, vol. 30, no. 2, pp. 254–287, 2013.

[21] A. Pfrunder, A. P. Schoellig, and T. D. Barfoot, "A proof-of-concept demonstration of visual teach and repeat on a quadrocopter using an altitude sensor and a monocular camera.," in *Conference on Robot Vision*, pp. 238–245, 2014.

[22] J. D. T. Raúl Mur Artal, "Fast relocalisation and loop closing in keyframe-based slam.," in *International Conference on Robotics and Automation*, 2014.

[23] J. Straub, S. Hilsenbeck, G. Schroth, R. Huitl, A. Moller, and E. Steinbach, "Fast relocalization for visual odometry using binary features," in *International Conference on Image Processing*, pp. 2548–2552, Sept 2013.

[24] J. Martinez-Carranza and W. Mayol-Cuevas, "Real-time continuous 6d relocalisation for depth cameras," in *Workshop on Multi VIew Geometry in RObotics (MVIGRO), in conjunction with Robotics Science and Systems*, 2013.

[25] D. Galvez-Lopez and J. Tardos, "Real-time loop detection with bags of binary words," in *International Conference on Intelligent Robots and Systems*, 2011.

[26] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *International Conference on Computer Vision*, 2011.

[27] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *International Conference on Very Large Data Bases*, 1999.

[28] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*, 2006.

[29] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tards., "A comparison of loop closing techniques in monocular slam.," in *Robotics and Autonomous Systems*, 2009.

[30] J. Martinez-Carranza, A. Calway, and W. Mayol-Cuevas, "Enhancing 6d visual relocalisation with depth cameras," in *International Conference on Intelligent Robots and Systems*, 2013.

[31] A. P. Gee and W. Mayol-Cuevas, "6d relocalisation for rgbd cameras using synthetic view regression," in *British Machine Vision Conference*, 2012.

[32] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in rgb-d images," in *International Conference on Computer Vision and Pattern Recognition*, June 2013.

[33] C. Cadena, D. Galvez-Lopez, F. Ramos, J. Tardos, and J. Neira, "Robust place recognition with stereo cameras," in *International Conference on Intelligent Robots and Systems*, 2010.

[34] E. Stumm, C. Mei, and S. Lacroix, "Probabilistic place recognition with covisibility maps," in *International Conference on Intelligent Robots and Systems*, pp. 4158–4163, Nov 2013.

[35] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel, "1-point ransac for ekf-based structure from motion," in *International Conference on Intelligent Robots and Systems*, 2009.

[36] J. Montiel, J. Civera, and A. J. Davison, "Unified inverse depth parametrization for monocular slam," in *In Proceedings of Robotics: Science and Systems*, 2006.

Algorithm 1: 2D-3D matching using 3D Neighbourhood Test

INPUT: $\mathbf{I}$ // Query Image
$\mathbf{HT}$ // Hash Tables
OUTPUT: $\mathbf{C}$ // Largest set of 2D-3D quality matches

\\ **Step 1**: Retrieve 2D-3D Matches
$\mathbf{S} = \mathbf{extract\_salient\_points}(\mathbf{I})$
$\mathbf{M} = [\,]$ // Initial set of 2D-3D matches
for each $s_i \in \mathbf{S}$ **do** // For each salient point
  $b_i = \mathbf{retrieve\_best\_binary\_descriptor}(\mathbf{HT})$
  $p_i = \mathbf{get\_descriptor\_3D\_position}(b_i)$
  $\mathbf{M} = \mathbf{M} \cup \{(s_i, p_i)\}$// Add 2D-3D match
end
\\ **Step 2**: Construct set of quality matches
$\mathbf{C} = [\,]$
for each $m_i \in \mathbf{M}$ **do**
  $\mathbf{M} = \mathbf{M} - \{m_i\}$
  $\mathbf{C}_{test} = \{m_i\}$
  for each $m_j \in \mathbf{M}$ **do**
    $r_j = \mathbf{neighbourhood\_radius}(m_j)$
    $is\_consistent\_match = \mathbf{true}$
    for each $c_k \in \mathbf{C}_{test}$**do**
      $d_{jk} = \mathbf{3D\_euclid\_dist}(m_j, c_k)$
      if $d_{jk} > r_j$ **then** // Neighbourhood check
        $is\_consistent\_match = \mathbf{false}$
        break
      end if
    end for
    if $is\_consistent\_match$ **then**
      $\mathbf{C}_{test} = \mathbf{C}_{test} \cup \{m_j\}$
    end if
  end for
  if $|\mathbf{C}_{test}| > |\mathbf{C}|$ **then**
    $\mathbf{C} = \mathbf{C}_{test}$
  end if
end for
\\ Pose estimation using three-point-pose+RANSAC will use $\mathbf{C}$
return C

Table 1: Mean results of continuous relocalisation using ORB binary descriptors of 512 bits and 8 hash tables.

Configuration	Min. # Inliers	# Iter. for RANSAC	Processing Time [ms]					Reloc. [%]
			FAST	Binary Match	NB Test	RANSAC	TOTAL	
BD	12	860	0.97	15.25	0	34.52	50.7 ± 16.6	11.5
BD+3DT	6	100	0.88	14.96	0.23	2.25	18.3 ± 6.1	26.5
BD+AT	6	100	0.88	14.14	0.20	2.51	17.7 ± 6.1	47.9
BD+CVT	6	100	0.95	14.68	0.21	2.8	18.6 ± 7.3	**51.7**