2015 Workshop on Research, Education and Development of
Unmanned Aerial Systems (RED-UAS)
November 23-25, 2015. Cancun, Mexico

Towards Autonomous Flight of Micro Aerial Vehicles using ORB-SLAM

José Martínez-Carranza[1], Nils Loewen[2], Francisco Márquez[3], Esteban O. García[4], and Walterio Mayol-Cuevas[5]

***Abstract*— In the last couple of years a novel visual simultaneous localisation and mapping (SLAM) system, based on visual features, has emerged as one of the best, if not the best, systems for estimating the 6D camera pose whilst building a 3D map of the observed scene. This method is called ORB-SLAM and one of its key ideas is to use the same visual descriptor, a binary descriptor called ORB, for all the visual tasks, this is, for feature matching, relocalisation and loop closure. On the top of this, ORB-SLAM combines local and graph-based global bundle adjustment, which enables a scalable map generation whilst keeping real-time performance. Therefore, motivated by its performance in terms of processing speed, robustness against erratic motion and scalability, in this paper we present an implementation of autonomous flight for a low-cost micro aerial vehicle (MAV), where ORB-SLAM is used as a visual positioning system that feeds a PD controller that controls pitch, roll and yaw. Our results indicate that our implementation has potential and could soon be implemented on a bigger aerial platform with more complex trajectories to be flown autonomously.**

I. Introduction

Unmanned Aerial Vehicles (UAVs) or Micro Aerial Vehicles (MAVs) have gained an increased popularity in the last years. Today, several companies offer MAVs ready to be flown without much skill, only with the help of a tablet, a PC or a conventional radio controller. However, several MAVs can be programmed and controlled to behave with certain autonomy.

Autonomous flight is an instance of what can be achieved by combining GPS-based localisation with control theory. But controlling an aerial platform imposes several challenges when it comes to controlling its dynamics, which may be affected by strong perturbations such as wind currents, vibration of the motors.

Another challenge equally relevant, and that we are particularly interested in this work, is that of controlling the vehicle in environments where GPS signal is not accessible or not reliable, a common situation is in indoor scenarios, when the vehicle flies through urban canyons or in cases where the GPS signal may be blocked/distorted due to the presence of clouds, trees or metallic structures.

From the above, in this paper we focus on the problem of controlling a low-cost MAV in a GPS-denied scene. In order to address the latter, we use a state-of-the-art visual SLAM system called ORB-SLAM which, according to the latest results reported in [1], appears to be the best visual SLAM system available to the public. Motivated by this, we used ORB-SLAM as the basis to develop a control system aiming at controlling our MAV under the constraints of neither having access to GPS nor using any other sensors but vision in order to control the vehicle.

We use a commercially available low-cost MAV with a closed architecture that does not allow to run user-made programs on board. Instead, only images can be transmitted via WiFi to a computer. The computer we used, as a sort of Ground Control Station, runs the Robot Operating System ROS, which allows us to get access to the MAV in a bidirectional fashion, this is, we can establish connection with the MAV and then receive video from the camera located at the front of the vehicle and we can send control commands to pilot the vehicle in four degrees: yaw, pitch, roll and height.

We use the ROS package for ORB-SLAM and adapted our system so that we could feed ORB-SLAM with the received imagery. Therefore, at every frame step, ORB-SLAM estimates the camera's position, which is used as the vehicle's 6D estimated position. The latter is then sent to our control algorithm that will use a PD controller in order to calculate control signals that will be sent back to the MAV to control its course. Our preliminary results indicate that even with the delay induced by having to process on the ground, ORB-SLAM effectively delivers useful camera and map estimates for our controller and the controller manages to pilot our MAV for it to be able to follow a target trajectory defined by way points.

Therefore, in order to present our results, this paper is organised as follows: section 2 presents the related work; section 3 describes our system setup; section 4 describes our commercial MAV platform; experiments and results are described in section 5; a final discussion and conclusions are presented in section 6.

II. Related Work

The most interesting tasks for MAVs require the vehicle to self localise. Localisation is often achieved in commercial systems by using satellite-based positioning systems (GPS, GALILEO, GLONASS, etc.). However, when flying in GPS-denied environments, the MAVs must carry on-board sensors capable of mapping their environment or depend on external systems, like motion capture [2], [3] or markers in controlled environments [4], [5], [6]. Lidar [7], [8], RGB-D or stereo cameras [9], [10], [11], [12] are often used in uncontrolled environments. However, in most cases those sort of sensors tend to reduce flight time due to heavy weight and are also very expensive to be considered part of common drones.

Cheap cameras are one of the most used devices in drones. The price and weight of an on-board camera makes monocular vision the most efficient technique to localise and navigate the vehicle in its environment. Optic flow [13], [14], [15] is a simple algorithm widely used to measure relative motion of the vehicle and also to discern between nearby and far away obstacles [16], but on the other hand, it tends to accumulate error when used for long term navigation.

Feature-based algorithms have become very popular to estimate position and also to track 3D reference points. PTAM [17] is a visual SLAM framework that has been used on quadrocopters to perform simple tasks, like keeping stable hovering [18], [19], [20] and also to follow geometric paths indoors and outdoors [21]. A recent technique called SVO (semi-direct monocular visual odometry) [22] can also enable way-point-based autonomous navigation, even when it is optimised to work with vertical oriented cameras.

In [23] a new monocular SLAM system based on the extraction of ORB features was presented. Its main advantage compared to other systems is that it is more robust, dealing with viewpoint changes, which makes it more suitable for aerial vehicle tasks, which to the best of our knowledge is an unexplored area.

III. System Setup

The entire processing occurs off-board the quadrocopter, on a laptop with the following specifications: AMD A10 quad-core processor with 4GB in RAM (a modest PC computer). The system is divided into three main stages, which are all built on the Robot Operating System (ROS). In the first stage, real-time video stream from the quadrocopter's horizontal camera is received on the computer using the ardrone autonomy package, which is based on the SDK released by the manufaturer. The video stream is then fed to the input channel of the state of the art ORB-SLAM algorithm, which is the second stage. ORB-SLAM is used to obtain the estimated pose of the quadrocopter and also builds a three dimensional sparse map of the robot's environment. The third stage is the control system, which allows the autonomous flight of the quadrocopter.

A. ORB-SLAM

Two recent monocular SLAM algorithms, which have shown impressive results are Large-Scale Direct SLAM[24] and ORB-SLAM[25], [1], [26]. LSD-SLAM is a direct visual SLAM, which means that it does not extract features from the imagery like ORB-SLAM, but it works directly on the pixel intensity gradient of the imagery. Such an approach is new; most of the SLAM methods up to date have relied on feature extraction.

However, ORB-SLAM, also available in the ROS platform, has emerged as a strong competitor that has leveraged the estimation process based on feature matching. Moreover, it proposes to use the same visual feature for all tasks, i.e., for tracking, mapping, relocalisation and loop closure. In this sense, and as its name indicates, the visual feature to use is the ORB descriptor [27], a binary

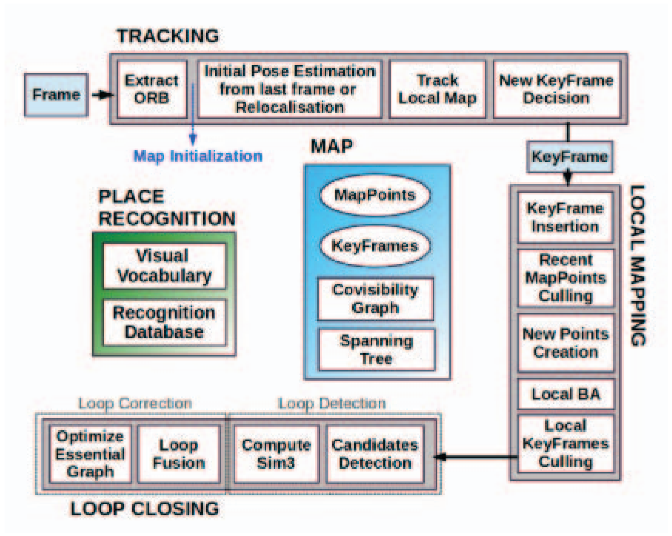


Fig. 1. An overview of the ORB-SLAM algorithm. The diagram was taken from [1].

descriptor robust to changes in rotation based on the estimation of the key point orientation using the intensity centroid of the image patch.

Therefore, ORB-SLAM is able to perform tracking, local mapping and loop closure with the visual feature ORB, hence, each one of these visual processes can be executed efficiently without need of GPU but just few processor instructions. These three proceeses are the three main threads of ORB-SLAM as shown in Fig. 1. Tracking is the thread in charge of estimating the camera pose, which is done by searching correspondences of map points from the previous frames in the current one. Besides this, the tracking thread also takes the decision when a new keyframe is to be inserted in the graph. Keyframe insertion depends on the change of visual information in the current frame with respect to a previous keyframe. The local mapping thread adds new map points to the map and determines if the map points stay part of the map after their creation. Local bundle adjustment optimises the map reconstruction. This thread also discards keyframes that seem redundant. The loop closure thread checks if a place is being revisited with each keyframe insertion. Once a loop is detected, the last keyframe and the loop keyframe are overlayed and global optimisation takes place, which means that the error between the two is distributed throughout the graph.

Fig. 2. The low-cost MAV vehicle used in this work: the Parrot AR Drone.

B. Controller

The control system maintains the quadrocopter on the straight line trajectory. The pitch and roll control is handled by a proportional derivative controller (PD), while yaw control is handled by a proportional controller. The waypoints are pitch references. Once the quadrocopter passes a certain threshold of the waypoint, a flag turns on indicating that the waypoint has been reached, independent of the roll and yaw position. When the waypoint is reached, the quadrocopter enters hover mode for approximately two seconds and then continues its route towards the next waypoint. The quadrocopter lands once it has reached the third waypoint.

IV. Aerial Platform

The Parrot AR Drone 2.0 is a low cost quadrocopter, it is shown in Fig. 2. Once it is switched on, it functions as an access point and any client device with WiFi connection may connect to it. It has two on-board cameras, one of which is horizontal and the other vertical, facing the ground. The horizontal camera is a HD 720p camera and captures images at 30 fps. The vertical camera has a low resolution (320 x 240) and runs at 60 fps. H264 codec is used to encode video stream for a lighter transmission to the computer. The aerial robot was quickly embraced by the research community due to the release of its SDK, which enables the user to receive navigation data and video stream from the quadrocopter and to send commands to control it.

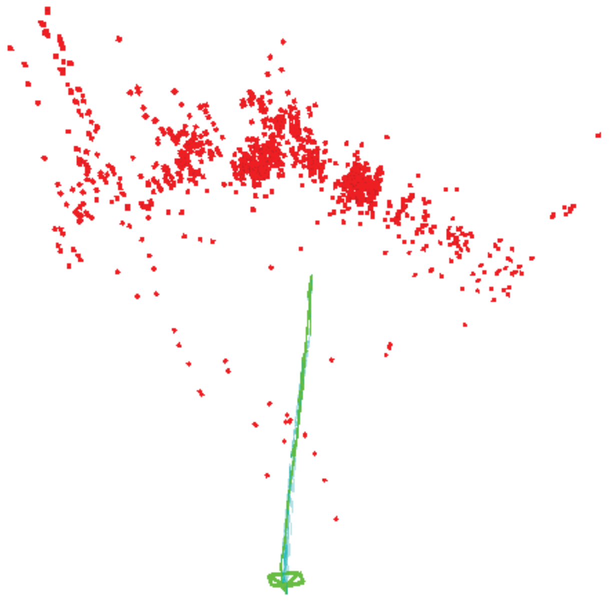


Fig. 3. Top view of the map and vehicle's trajectory obtained with ORB-SLAM. Way points are sampled from the trajectory and the map is used to localise the vehicle during the autonomous flight stage. The current vehicle's position is indicated by the green rectangle located at the bottom of the trajectory. This position also corresponds to the start point.

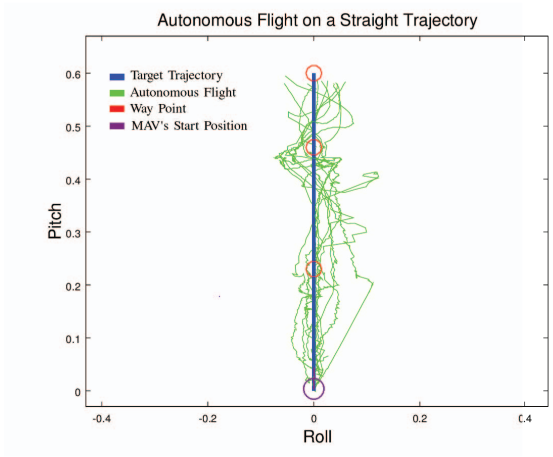


Fig. 4. Experiment 1. Straight Trajectory Autonomous Flights.

V. Experiments

Our experiments demonstrate the autonomous flight on a straight trajectory employing a Teach and Repeat approach. The straight trajectory is defined by a starting point and three waypoints. Two other sets of experiments were conducted where we changed the starting position of the quadrocopter in terms of roll and orientation (yaw). These experiments were designed to showcase the ability of our control to correct the orientation of the vehicle so that it can navigate the straight trajectory in forward direction.

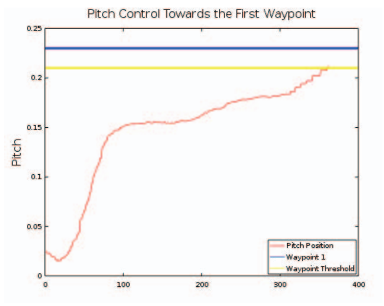
A first and very important step, for all of the three scenarios presented in this work, was that of creating the 3D map and recording the way points sampled from the traversed trajectory during this mapping stage. Fig. 3 shows the 3D map and the vehicle's trajectory estimated with ORB-SLAM. From this trajectory, three way points plus the start position are recorded in our system. The third way point is used as the stop position where the vehicle has to autonomously land once such way point has been reached. The same map and target trajectory was used for all the runs. Also, for all the experiments, we worked with a 640×360 image resolution.

We should indicate that ORB-SLAM assigns an arbitrary scale when working with a monocular camera. This did not affect our experimental framework since all our runs were carried out in the same scene with identical start points and same way point locations. According to the units delivered by ORB-SLAM, the total length of the target trajectory was 0.6. When matched against the real length in the scene, this corresponded to 6 metres.

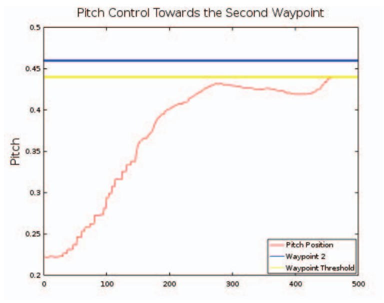
For all the experiments, the goal is for the vehicle to take off and then navigate autonomously towards each way point. Every time the vehicle reaches a way point, it hovers for a couple of seconds in order to indicate that it has reached the way point, after this stop time is over, it navigates towards the next way point. To indicate that the vehicle has reached a way point, a radius around the way point's position is set, if the pitch value falls within that radius then the system indicates that the way point has been reached, hence, the vehicle switches to the hovering mode. For all the experiments we keep the same vehicle height.

A. *Autonomous Flight on a Straight Trajectory*

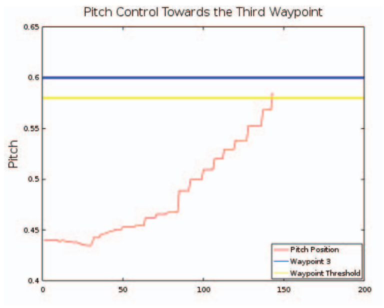
For this scenario we placed the MAV at the start point, as indicated in Fig. 3. The vehicle looks forward in the direction of the line defined by the target trajectory. The yaw value is close to zero. After taking off, the control module kicks in immediately and uses the position estimates to



(a) Pitch Towards the First Waypoint.



(b) Pitch Towards the Second Waypoint.



(c) Pitch Towards the Third Waypoint.

Fig. 5. Pitch value for one of the runs in the *straight trajectory* experiment and for each one of the way points.

calculate the control signals using the PD controllers. The position of the target way point is used as reference within the controllers in order to calculate the control signal. The obtained signal output is scaled accordingly to fall between -1 to 1 as this is the range used to control the MAV. The PD controllers control yaw, pitch and roll aiming at piloting the vehicle so it follows the straight target line passing through each one of the way points.

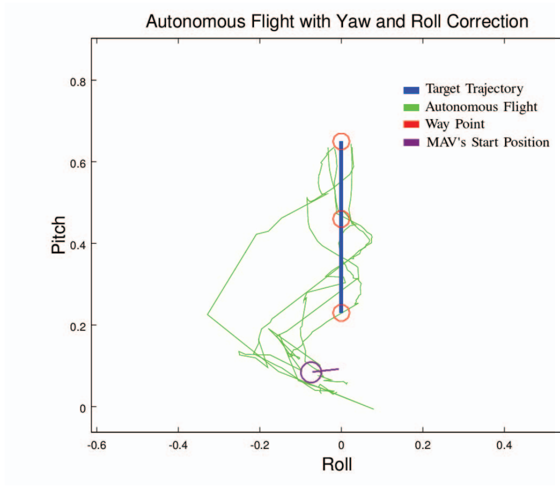
For this experiment we carried out 10 runs. The map for these runs was always the same as much as the way point positions. Fig. 4 shows a top view of the vehicle's position, which was piloted by our PD controller. We registered the vehicles position every time the system indicated that the vehicle had reached a way point and we used these values to calculate a mean error for pitch. These mean values are presented in the first row of table I. In average, the error in pitch is of about 2% w.r.t. the total length of the target trajectory. According to the real scale in metres, this corresponds to an average error of 13.8 centimetres.

Similarly, note that the error in roll is of around 8%. We attribute the increase in error for roll to the delay in the imagery reception. This means that the control module for roll attempts to keep the vehicle over the target line, for which the roll value is zero (the reference), however, the delay will cause the control to work with a vehicle position that is in the past and therefore, the vehicle will pass the reference without being noticed by the control until some milliseconds later, this causes a small oscillation of the vehicle around roll, however, we are confident that this can be mitigated with a more careful tuning of the control gains or by using some sort of predictive filtering within the control input. In this same context, we also believe that the delay affects less to pitch thanks to the hovering mode that kicks in once the system detects that the vehicle has reached the way point.

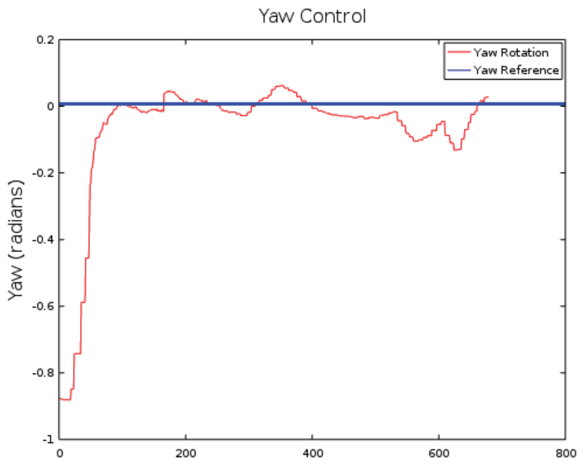
B. *Autonomous Flight with Yaw correction*

We name this set of experiments as *yaw correction* in the spirit that at the beginning of each run, the vehicle is placed besides the target trajectory, either to the left or to the right, and a bit off from the start point. What is interesting here is that the vehicle heading is almost perpendicular to the target trajectory, this is, if it is desired that the vehicle follows the line with the camera looking forward, then the control *has to correct* yaw in order to rotate the vehicle so that it moves forward in the direction of the line defined by the target trajectory.

With the scenario described above, we carried out 10 runs with 5 runs where the vehicle was placed to the left of the target trajectory and 5 runs placed to the right. Fig. 6a shows a top view of the vehicle's position for each run. Note that the vehicle's start position is marked with a purple circle and a line indicating the vehicle's heading. Note that this heading is almost perpendicular to the line defined by the target trajectory, depicted

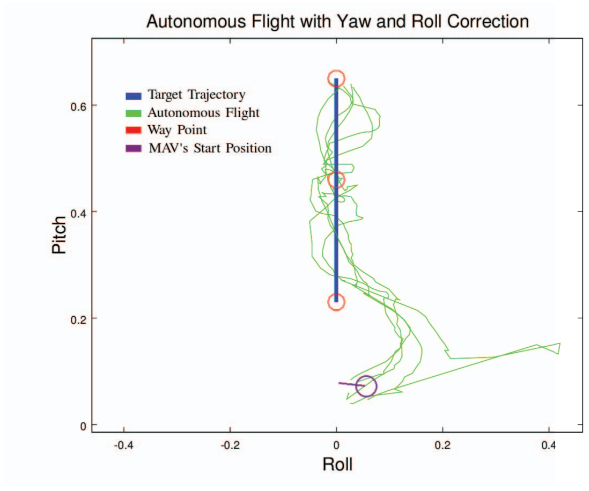


(a) Top view of 5 runs where the MAV (purple) is located to the left of the target trajectory (blue). Note that the front and orientation of the vehicle is marked by the purple line.

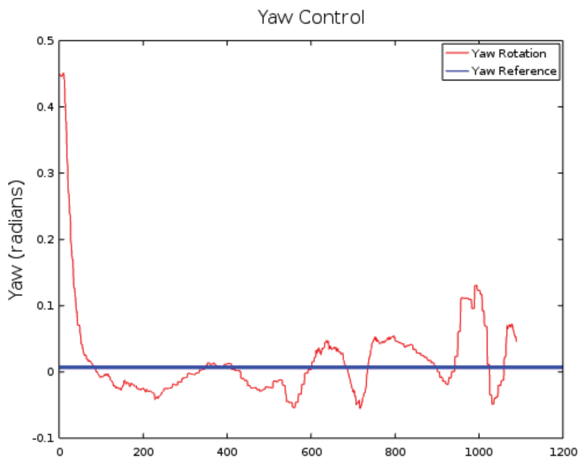


(b) Yaw value for one of the runs converging towards the reference angle indicated by the blue line.

Fig. 6. Yaw correction experiment taking off from the left of the target line (5 runs): the control corrects yaw in combination to roll and pitch in order to pilot the vehicle such that it can navigate towards each of the way points. Note that there are variations in the runs which are due to the arbitrary direction taken by the MAV when taking off. This is due mostly to the ground effect as the experiments were conduced in a closed area.



(a) Top view of 5 runs where the MAV (purple) is located to the right of the target trajectory (blue). Note that the front and orientation of the vehicle is marked by the purple line.



(b) Yaw value for one of the runs converging towards the reference angle indicated by the blue line.

Fig. 7. Yaw correction experiment taking off from the left of the target line (5 runs): the control corrects yaw in combination to roll and pitch in order to pilot the vehicle such that it can navigate towards each of the way points. Note that there are variations in the runs which are due to the arbitrary direction taken by the MAV when taking off. This is due mostly to the ground effect as the experiments were conduced in a closed area.

in blue in the figure. Fig. 6b shows, for one run, the evolution of yaw during the whole flight.

Similarly, Fig. 7a shows the outcome for the runs when the vehicle is placed to the right of the target trajectory with the orientation almost perpendicular to the line of the target trajectory. Fig. 7b shows the evolution of yaw towards the reference for one of the runs.

VI. Discussion and Conclusions

We have presented initial results of a work that aims at developing a full system for autonomous flight of low-cost MAVs in GPS-denied environments. In order to estimate the vehicle's position, we explored the use of a sophisticated visual SLAM system, launched last year, called ORB-SLAM. This system is capable of estimating the 6D position of a monocular camera moving freely in space, whilst simultaneously building a 3D map

TABLE I

AVERAGE ERRORS IN PITCH AND ROLL FOR THE EXPERIMENTS ON AUTONOMOUS FLIGHT.

Total Length of Target Trajectory: 0.6 (corresponds to 6 metres)				
Experiment	**Mean Error in Pitch for each Way Point**			**Mean Error for Roll**
	Way Pt 1	Way Pt 2	Way Pt 3	
Straight	0.017±0.0043	0.019±0.0007	0.015±0.0045	0.025±0.0193
Start Left	0.014±0.0056	0.012±0.0077	0.012±0.0046	0.050±0.0501
Start Right	0.014±0.0056	0.011±0.0070	0.013±0.0042	0.069±0.0648
	Average Error: 2.3%			**Average Error: 8%**

of the observed scene.

To the purposes of our work, we used a commercially available low-cost MAV platform. This platform comes with an on-board frontal camera whose images can be transmitted via WiFi to a computer. In order to communicate with this platform, in terms of acquiring imagery from the vehicle and sending navigation commands to the vehicle, we use the Robot Operating System also known as ROS, on which ORB-SLAM was also available as a ROS package.

Before using ORB-SLAM, we tested with some other visual SLAM solutions available as ROS packages such as SVO [28] and LSD SLAM [24]. However, we found difficultes in using SVO with a frontal camera since features far in the horizon did not seem to converge well, this system seems to be more suitable for vertical cameras or for scenarios where the scene structure lies close to the camera view at the outset in order to obtain good depth estimates. Regarding LSD SLAM, the main problem we found was the processing time, which was of 5 fps in our modest computer hardware working with the 640×360 image resolution. We reduced the size to (320×240) achieving a processing speed of 10-15 fps, however, when running outdoors the map did not seem to converge well enough towards the real scene structure.

In contrast to the systems mentioned above, ORB-SLAM did not require any fine tuning or downsampling of the image resolution. We used the 640×360 resolution and it worked at approximately 15-20 fps. We are certain that on a more powerful PC computer (for instance, an Intel i7 quadcore processor) this frame rate would increase significantly. At this point, we should indicate that we decided to use a modest hardware as we are interested in working with low-budget hardware such as the Odroid U3 computer, which is a very small and light computer with a dual core processor, running at 1.7 GHz and with 2 GB in RAM. This computer is ideal to be mounted on a MAV in order to run our algorithms on board. This is, however, part of our future work.

We also should comment on the effects of the delay in the received imagery from the MAV to the PC. We expected ORB-SLAM not to work at all, however, its dual optimisation engine (local and global bundle adjustment) coped well enough with this issue and did not have any troubles to estimate the camera position. It was fast and robust enough to the vehicle's agile motion, especially in the experiments where yaw had to be corrected in order to align the vehicle w.r.t. the reference yaw. Certainly, this delay had an impact in the precision of our autonomous navigation whose error was in average of around 2% for pitch and 8% for roll, w.r.t. the total length of the target trajectory. Nevertheless, our results indicate that our control module was capable of using the vehicle's position estimates (even with delay) to pilot the vehicle such that it could navigate passing through each way point until reaching the end position.

Our future work involves continuing with the development of our system based on ORB-SLAM. We plan to improve our control module in order to enable the vehicle to navigate more complex trajectories. We also plan to add an obstacle detection module based on the 3D map estimates generated online. Finally, we are also aiming at working with bigger MAV platforms.

VII. ACKNOWLEDGEMENTS

This work has been funded by the Royal Society-Newton Advanced Fellowship with reference NA140454. We would like to thank David Carrillo López for his support on the experimental setup and video recording of this work.

REFERENCES

[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics (to appear)*, 2015.

[2] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 2520–2525, IEEE, 2011.

[3] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. DAndrea, "A platform for aerial robotics research and demonstration: The Flying Machine Arena," *Mechatronics*, vol. 24, no. 1, pp. 41–54, 2014.

[4] D. Eberli, D. Scaramuzza, S. Weiss, and R. Siegwart, "Vision Based Position Control for MAVs Using One Single Circular Landmark," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1-4, pp. 495–512, 2011.

[5] C. Nitschke, "Marker-based tracking with unmanned aerial vehicles," in *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on*, pp. 1331–1338, 2014.

[6] 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.

[7] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 2878–2883, IEEE, 2009.

[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] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *International Symposium on Robotics Research (ISRR)*, pp. 1–16, 2011.

[10] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers, "Real-time camera tracking and 3d reconstruction using signed distance functions," in *Robotics: Science and Systems (RSS) Conference 2013*, vol. 9, 2013.

[11] 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*, p. 733219, International Society for Optics and Photonics, 2009.

[12] J. Martinez-Carranza, A. Calway, and W. Mayol-Cuevas, "Enhancing 6D visual relocalisation with depth cameras," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 899–906, IEEE, 2013.

[13] K. Schmid, F. Ruess, M. Suppa, and D. Burschka, "State estimation for highly dynamic flying systems using key frame odometry with varying time delays," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 2997–3004, IEEE, 2012.

[14] V. Grabe, H. H. Bülthoff, and P. R. Giordano, "Robust optical-flow based self-motion estimation for a quadrotor UAV," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 2153–2159, IEEE, 2012.

[15] P.-J. Bristeau, F. Callou, D. Vissiere, N. Petit, and Others, "The navigation and control technology inside the ar. drone micro uav," in *18th IFAC world congress*, vol. 18, pp. 1477–1484, 2011.

[16] 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.

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

[18] M. Bloesch, S. Weiss, D. Scaramuzza, and R. Siegwart., "Vision based mav navigation in unknown and unstructured environments," in *ICRA*, 2010.

[19] 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.

[20] 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.

[21] J. Engel, J. Sturm, and D. Cremers, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," *Robotics and Autonomous Systems*, vol. 62, no. 11, pp. 1646–1656, 2014.

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

[23] R. Mur-Artal and J. D. Tardós, "Fast relocalisation and loop closing in keyframe-based SLAM," in *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, pp. 846–853, 2014.

[24] J. Engel, T. Schps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Computer Vision ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), vol. 8690 of *Lecture Notes in Computer Science*, pp. 834–849, Springer International Publishing, 2014.

[25] R. Mur-Artal and J. D. Tards, "Probabilistic semi-dense mapping from highly accurate feature-based monocular slam," in *RSS*, 2015.

[26] Mur-Artal, R., and J. D. Tardos, "Fast relocalisation and loop closing in keyframe-based slam," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 846–853, 2014.

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

[28] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 15–22, May 2014.