Multi-agent environment exploration with AR.Drones

Richard Williams, Boris Konev, and Frans Coenen

Department of Computer Science, University of Liverpool, Liverpool L69 7ZF
{R.M.Williams1, Konev, Coenen}@liv.ac.uk

Abstract. This paper describes work on a framework for multi-agent research using low cost Micro Aerial Vehicles (MAV's). In the past this type of research has required significant investment for both the vehicles themselves and the infrastructure necessary to safely conduct experiments. We present an alternative solution using a robust, low cost, off the shelf platform. We demonstrate the capabilities of our system via two typical multi-robot tasks: obstacle avoidance and exploration. Developing multi-agent applications safely and quickly can be difficult using hardware alone, to address this we also present a multi-quadcopter simulation based around the Gazebo 3D simulator.

1 Introduction

Aerial robotics is a very exciting research field with many applications including exploration, aerial transportation, construction and surveillance. Multi-rotors, particularly quad-copters, have become popular due to their stability and maneuverability, making it easier to navigate in complex environments. Their omnidirectional flying capabilities allow for simplified approaches to coordinated pathfinding, obstacle avoidance, and other group movements, which makes multi-rotors an ideal platform for multi-robot and multi-agent research. One major difficultly in multi-rotor research is the significant investment of both time and capital required to setup a safe, reliable framework with which to conduct research. This discourages researchers from conducting any practical experimentation [10] which leads to a knowledge gap.

This work attempts to bridge that gap and encourage more practical research by providing a framework based on a low cost platform that does not require dedicated infrastructure or expensive platforms to conduct safe, effective experiments. For this work the Parrot AR.Drone[1] quad-rotor, a relatively low cost commercial toy developed for augmented reality games, was chosen as the experimental platform. We develop a two-tiered software architecture for facilitating multi-agent research with AR.Drone. At the lower level of our architecture we provide the basic robotics tasks of agent localisation, position control and obstacle avoidance. To do so, we extend the PTAM [3] key-frame-based monocular SLAM system to provide localisation and mapping functions for multiple agents.

[1] `http://ardrone2.parrot.com/`

An Extended Kalman Filter (EKF) is used for state estimation and a PID controller provides position control and path following. At the higher level we implement collision avoidance based on the Optimal Reciproal Collision Obstacle (ORCA) approach [19] and a multi-agent approach to autonomous flying where each quadcopter agent communicates with it's peers to achieve goals such as exploring an environment. The development and practical evaluation of meaningful higher level multi-agent procedures requires extended experimentation, which is hindered by the short flying time of the AR.Drone. To address this issue, we also develop a simulation environment based on the Gazebo [12] 3D simulator. We conduct both simulated and real world experiments to demonstrate the use of our framework in a multi-agent exploration scenario and validate the veracity of the simulation.

The remainder of the paper is organised as follows. In Section 2 we discuss other approaches to using AR.Drone in single- or multi-robot research. In Section 3 we present our framework in detail. Section 4 introduces the multi-agent exploration scenario and the results and conclusion are discussed in Sections 5 and 6 respectively.

2 Related Work

While the low cost and highly robust construction make AR.Drone an ideal platform for academic research, it is primarily a toy meant to be under human control at all times, whose built-in functionality lacks the precision required for the applications mentioned above. For example, while the AR.Drone features optical flow based horizontal velocity estimation, which can be integrated for position estimation (dead reckoning), it is subject to drift. This is unsurprising as the inclusion of an optical flow sensor on the AR.Drone is mainly for position hold rather than position estimation. Therefore a more robust drift-free position estimation solution is required.

We are aware of three distinct solutions, which have been successfully tested on the AR.Drone: using external high precision positioning devices, using fiducial markers and using visual SLAM. Motion capture systems, such as the Vicon system[2], make use of a system of high-speed infrared cameras and retro-reflective markers to estimate the full 3D pose of arbitrary bodies with sub-millimetre precision. While a motion capture system would provide the highest accuracy it comes with a significant price tag. Additionally many of these systems are limited in the number of targets they can track (usually 4–6).

Fiducial markers, or *tags*, can either be placed on the robots themselves and tracked using a fixed system of cameras [22] or the markers may be placed at known locations in the environment and tracked by on-board cameras (e.g. [17]). This system has the advantage of a high level of accuracy at very low cost. However it does require significant setup and does limit the range of the robots so as to be certain of their position the drones must always be able to see at least one marker.

[2] `http://vicon.com/`

A third option is a feature-based localisation system, most notable is the monocular SLAM system PTAM (Parallel Tracking and Mapping). This system makes use of FAST [15] visual features and as such does not require fiducial markers to be placed in the environment at know locations but simply requires an environment with sufficient texture such as an office or lab space. PTAM has been successfully employed for quad-copter localisation in in a single agent configuration in a number of projects [2, 5, 6].

The main drawback for multi-robot needs is the lack of multi-camera support in the original PTAM. Castle et al. [3], however, successfully demonstrated that in an augmented reality setting, given the decoupled nature of the tracking and map making processes which is a hallmark of the PTAM approach, it is possible to implement multiple camera tracking and mapping (using a single shared map) with relatively small effort. Castle et al. applied the problem of tracking a single user in an augmented reality setting using two cameras. To the best of out knowledge no one has previously applied the approach to multiple independent cameras or multi-robot localisation and mapping.

3 Framework Architecture

Before giving detail on the low- and high-level architecture of our framework, we briefly mention the two modifications of the original AR.Drone system that were necessary for our framework to work. The drone features two cameras, the front facing camera is capable of streaming images at 640×360 and is fitted with a $92°$ wide angle lens. The downward facing camera is primarily used for the on-board optical flow and captures images at 160×120 which is upscaled to 640×360 for streaming. Our multi-robot scenario requires omni-directional flying, and we found it difficult to keep control of the drone and avoid obstacles based on either of the video streams: the resolution of the downward facing camera was too low for reliable localisation, and the drone was flying into walls and obstacles while moving in the direction not covered by the front-facing camera. Therefore, we re-arranged the front-facing camera to look down. This is a very straightforward modification due to the modular structure of the AR.Drone. As a result, the drone can reliably avoid obstacles as long as they come upwards from the floor.

Another technical difficulty that we met was communication. The drone communicates via 802.11n WiFi which is used to stream control and sensor data as well as a single compressed camera feed. The high bandwidth of the information stream leads to communication latencies, especially when multiple drones are deployed at the same time. Additionally, we found that there is a lot of interference on the 2.4GHz frequency range used by the drone due to various WiFi networks in the proximity of our lab. We tried different options and it turned out that enabling the WPA2 security protocol on the drone and the access point dramatically reduce the latency. Implementing this modification involves installing a cross-compiled version of the open-source application wpa_supplicant and reconfiguring the network settings of the AR.Drone.

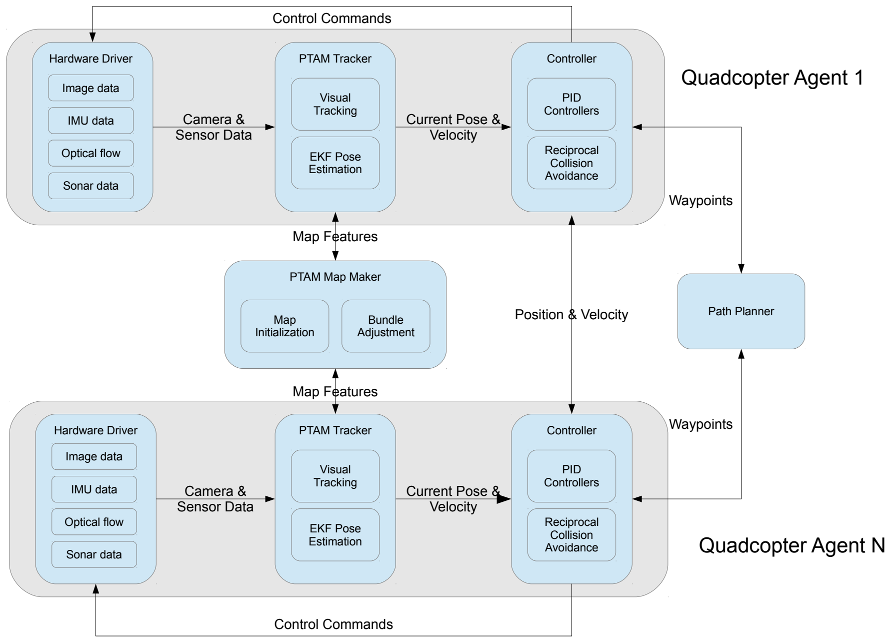

Fig. 1: The System Architecture

Our framework has been implemented in C++ and integrated into the Robot Operating System (ROS) [14], each main component within our system has been implemented as a separate ROS node meaning that in theory each component could be run on a separate processor/computer adding to the scalability of our framework.

3.1 Low-Level Architecture

In this section we describe the main components of our system. A general overview of the key components of our system is shown in (Fig. 1). In what follows we describe in detail the monocular visual SLAM system for mapping and camera pose estimation, the Extended Kalman Filter (EKF) for camera pose and inertial sensor data fusion and the PID based way-point controller.

Localisation and Mapping The framework makes use of the Monocular keyframe-based SLAM package, PTAM, inspired by the work described in [11]. The original system introduced in [11] allowed only a single map to be built using images from a single camera; it was later extended [3] to allow multiple maps to be built using images from multiple cameras. We make use of this additional functionality within our framework to allow multiple quadrotors to localise using, and contribute to, the same map. To this extent, we have extended the system to create a PTAM tracking thread for each quadcopter. Each tracking thread is responsible for generating a visual pose estimate using the underlying PTAM tracking algorithm and fusing this with the additional sensor data using the EKF (see next section). PTAM tracks the camera pose (position and orientation) within the map which makes it necessary, for our purposes, to apply a transformation (based on the physical location of the camera on the drone) to obtain the PTAM estimate of drones pose. Pose estimates take the form of position (x, y, z) and orientation (roll, pitch and yaw angles) i.e. PTAM pose for drone i at time t is given by $\boldsymbol{ptam}_t^i = (x, y, z, \Phi, \Theta, \Psi)$.

State Estimation In addition to the horizontal velocity estimation the AR.Drone also features an IMU (Inertial Measurement Unit) which measures the body acceleration, angular velocity and angular orientation. These measurements are fused on-board by a proprietary filter to give the estimated attitude of the drone. We use this orientation estimate, together with the PTAM pose in an Extended Kalman Filter [18] (EKF) to estimate the current position and orientation w.r.t to the global world frame for each drone.

The sensor data from the drone comes from very low-cost sensors and as such is subject to a significant amount of noise. The accuracy of the pose generated by PTAM can also vary depending on many factors such as the quality of the map, ambient lighting conditions and the motion of the drone (fast movements cause significant problems for rolling shutter cameras, like the ones on the AR.Drone). Filtering is a common approach to extract useful information from noisy/unreliable data, our framework makes use of an EKF to fuse the noisy drone sensor data ($\boldsymbol{imu}_t^i$) with uncertain PTAM poses ($\boldsymbol{ptam}_t^i$) to produce a more precise estimate of the drones pose given by $\boldsymbol{ekf}_t^i$.

The AR.Drone communicates via a WiFi link and there is no synchronisation between video frames and IMU data, thus it is necessary to explicitly handle the synchronisation of the data. Engel et al [6] do this by keeping a history of observations between the time the last visual SLAM estimate was available up to the current moment. When a new estimate is required the filter is rolled forward up-to the required time, integrating all available measurements from the observation buffer. This approach compensates for the delay/missed observations that occur from the unreliability of the WiFi link to the drone. Our framework extends this approach to a multi-robot scenario.

Position Control and Path Following We make use of traditional PID controllers for x,y,z position and Yaw control. Input consists of the desired x,y,z theta (or a sequence of positions for path following), an independent PID controller for each degree of freedom calculates the control command necessary to achieve the goal position. The gains for each PID controller were calibrated experimentally and the values tested on multiple quad-copters.

Simulation For simulations we chose the Gazebo multi-robot simulator as it provides capabilities to model complex 3d environments, reliably model multi-

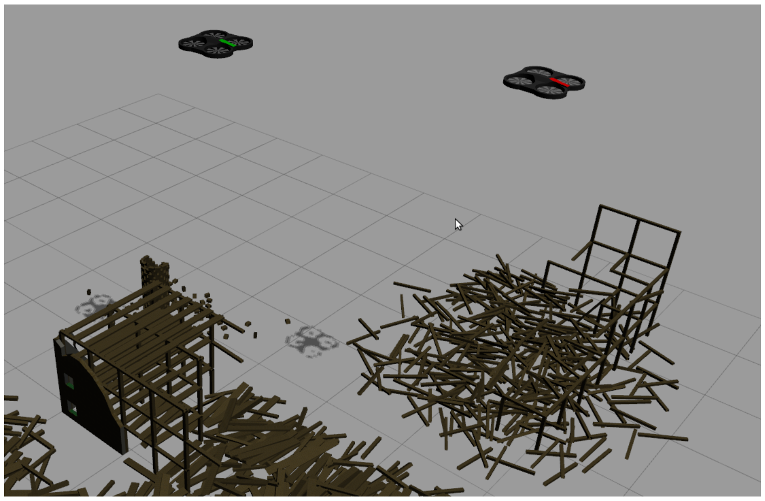

Fig. 2: The simulation environment

ple flying vehicles and generate realistic sensor data including camera images. Gazebo also integrates cleanly into the distributed infrastructure in ROS which means we are able to test our framework on both the simulated and real robots without altering the framework.

Meyer et al. [13] introduced a number of UAV-specific sensor plug-ins for Gazebo such as barometers, GPS receivers and sonar rangers. In addition to this they have created a comprehensive quadrotor simulation that includes accurate flight dynamics. For this work we focused on accurate sensor modeling rather than flight dynamics and as such our simulator uses a simplified flight dynamics model that does not accurately model the aerodynamics and propulsion behavior of the AR.Drone. We make use of Meyer et al's simulator plug-ins to replicate the sensor suite on the AR.Drone.

3.2 Higher-Level Architecture

We have implemented as easy to use interface to control the quadcopters based on ROS messages and actions. The action paradigm in ROS allows the execution of an action, e.g. move to a waypoint or follow a path, and provides facilities to monitor execution of the action, determine the success/failure and preempt a running action. High level agent code can be written in any of the supported ROS programming languages including Python, C++, Java and Lisp using the same interface.

Obstacle Avoidance In our initial experiments, for simplicity, we were fixing the specific altitude at which each drone could operate thus avoiding colliding with one another. This choice, however, limits drastically the number of drones we could safely fly (especially indoors); therefore, a more robust solution was

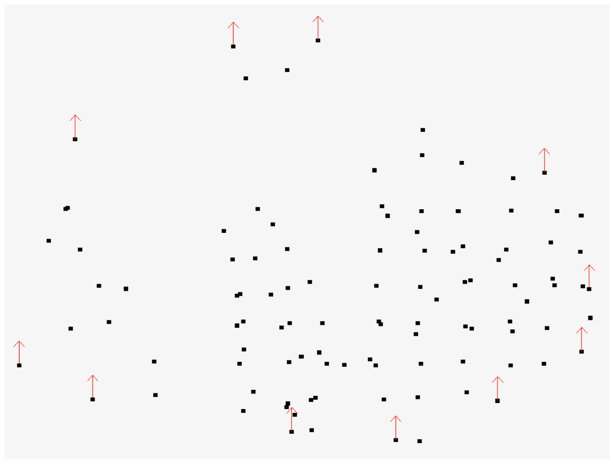

Fig. 3: An example of frontier point extraction showing the downsampled point-cloud and the frontier points (red arrows)

required. We assume that drones are the only dynamic obstacles in the environment. The velocity obstacle [8] (VO) is a representation of the set of all unsafe velocities i.e. velocities that will eventually result in a collision. Van den Berg et al. reasoned that in a multi-agent environment the other agents are not just dynamic obstacles but reasoning agents who themselves take steps to avoid obstacles. This leads to the notion of the reciprocal velocity obstacle [1] (RVO) where each agent takes half the responsibility for avoiding the collision under the assumption that all other agents reciprocate by taking complementary avoiding action. To enable efficient calculation of safe velocities van den Berg et al. introduced the Optimal Reciprocal Collision Avoidance (ORCA) [19] approach which uses an efficient linear program to calculate the half-planes of collision free velocities for each other agent. The intersection of all half-planes is the set of collision free velocities.

We use the ORCA approach with a simple agent communication model, each agent transmits it's current position and velocity to all agents. At each time step an agent will read in all available messages and calculate a safe velocity based on it's current goal and all the other agent positions and velocities. In ORCA each agent is represented by a sphere of fixed radius, however this does not take position uncertainty into account. We apply a similar approach to [9], where the radius of the robots is varied according the uncertainty within the particle filter used for localisation. In our framework we take into account not only the localisation uncertainty but, as our flying robots also lack the comparatively accurate velocity estimates from wheel odometry, we also account for the uncertainty in velocity estimation. The main limitation of this approach is the reliance on communication between the agents, an alternative would be to use relative sensing similar to Conroy et al's [4] work. However a drawback to that approach is the limited field of view of the AR.Drones cameras mean only head on collisions would be avoided, this is exacerbated by the fact the we relocated the front facing camera to look down.

4 Cooperative Exploration with Auction Mechanisms

Our first goal with the framework was to develop a way to deploy a team of AR.Drones to autonomously explore an environment. More formally given a map m, consisting of k feature-points, how can we extract a set of *interest points* $i \subseteq k$ so that by assigning an AR.Drone to explore these interest points extends the existing map m. This presents two challenges:

1. How to extract the set of interest points and
2. How to assign these tasks to the team of AR.Drones in an efficient manner

To address these two challenges we have developed an extension of our framework that uses a frontier-based approach for interest point extraction and a Sequential Single Item (SSI) auction approach for task assignment.

4.1 Interest Point Extraction

One of the most widely used exploration algorithms is the Frontier based algorithm introduced in [20] and extended to multiple robots in [21]. We apply a similar idea in our exploration strategy within our system; however, instead of an occupancy grid map model we have a sparse pointcloud of natural features.

In order to extract frontier points from this sparse pointcloud we first downsample the pointcloud, to reduce the granularity. This has the added benefit of improving the efficiency of the subsequent steps as the feature-map grows.

In order to build a reliable map each drone must be certain of it's position when adding new features, this means that each drone must keep a portion of the existing feature map visible at all times in order to maintain good visual tracking. Therefore the interest points must be sufficiently close to previously mapped areas while still being sufficiently far away so new features can be discovered. To achieve this we first project the set of feature points onto the ground plane and compute the outliers/boundary points, these points represent the boundary between know/mapped areas and unoccupied areas. Due to the down-sampling step these points are sufficiently close to already mapped areas while still being close enough to maintain stable tracking. Figure 3 shows an example of this frontier point extraction process.

4.2 Auction Mechanism

In order to assign frontier points to drones we make use of a simple Sequential Single Item (SSI) auction mechanism, which we adapted as follows:

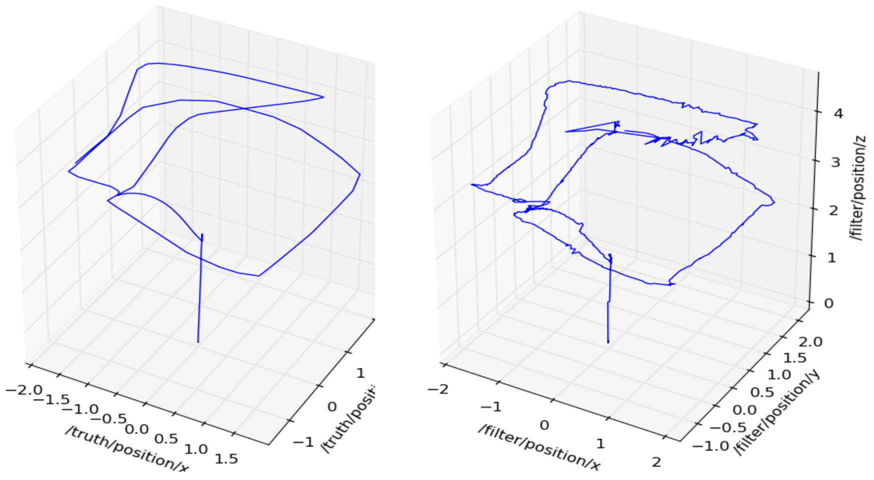

Fig. 4: Plot of the ground truth position (left) and the estimated position (right) Pose Estimation RMSE: 2.65cm

- On initiation the first set of frontier points is extracted from the current map.
- Points are auctioned to the available drones. All points are auctioned off and all drones bid on every point. Bids consists of path costs (linear distance between all points on the path), where the path includes all the previous points a particular drone has won. Therefore a drone bids on the cost of adding the latest point to its existing goals, rather than on the properties of these points.
- Once all points have been won the drones visit each of the frontier points adding a new features to the map as they are discovered.
- After successfully completing their missions another round commences and another frontier point extraction is carried out on the new expanded map.
- This process continues until no new points are added to the map or the drones run low on battery.

5 Evaluation

Sufficient work has already gone into verifying the performance of PTAM-based AR.Drone localisation w.r.t the ground truth [6,7,16]. Our system does not perform significantly differently from any of these system and as such our testing extended to simply verifying our system performs in line with the results others have reported. To that end we conducted several manual flight tests in simulation and compared the estimated path to simulated ground truth. Figure 4 shows the results of one such test. As expected the performance is simulation

#AR.Drones =1				#AR.Drones =2				#AR.Drones =3			
Time	FP	MP	DP	Time	FP	MP	DP	Time	FP	MP	DP
900	86	21814	7567	437.76	74	21055	5820	310.84	79	18010	5017
900	192	21069	6582	467.47	82	20687	5852	353.23	79	19048	5949
900	161	22193	6813	434.01	69	17938	4746	316.33	74	17666	4559
900	146	19370	5992	403.89	74	17080	5063	487.04*	81	18298	5242
900*	125	12987	3767	508.34	86	19417	6167	332.15	74	16319	5200

Table 1: A table showing the results from 3 explorations experiments using one, two and three drones, respectively. The table shows the time taken and the number of: Frontier Points (FP), Map Points (MP) Deleted Map Points (DP)

is marginally better than other reported results this can be accounted for by the lack of distortion and lighting effects in the camera images and the lack of physical disturbances such as air wind, air flow disturbances from other drones.

Collision Avoidance We conducted collision avoidance experiments both in simulation and in the real world. The simulated experiments featured 2-4 AR.Drones and 75 experiments were conducted. The drones were instructed to follow intersecting paths, and the collision avoidance mechanism was tasked with resolving the situation. Of the 75 simulated experiments conducted 73 collisions were avoided, the 2 collisions that did occur were due to complete PTAM tracking loss. The limited flying time and available resources made running real experiments more difficult so we were only able to complete 12 experiments with the 2 AR.Drones we have available. Out of the 12 experiments 3 resulted in collisions, with 9 collisions successfully avoided. Interestingly the 3 cases of collision were a result of WiFi communications latency i.e the robots took avoiding measures but did so too late to avoid the collision.

Exploration We conducted three sets of exploration experiments with teams of 1, 2 and 3 drones. The experiments were conducted in the simulated environment shown in Figure 2, the area consists of 20 x 17 metre area with the take-off/landing location roughly in the centre. The environment model consisted of piles of planks and damaged structures to simulate the exploration of a disaster site. Additionally the use of large numbers of similar models was designed to be particularly challenging for a Visual-SLAM based localisation framework. The results from the exploration experiments are shown in Table 1, a timeout of 15 minutes was set for all experiments. It is interesting to note that while 1 AR.Drone is not able to completely map the environment within the time limit the number of map points found is larger particularly when comparing the results from the single AR.Drone experiments to those of the 3 AR.Drone experiments. This is accounted for by the fact that a single AR.Drone exploring the frontiers of a map often has to cross from one side of the map to the other and is able to find additional points within the existing map. Another interesting case is the two simulated runs marked with a '*'. These cases highlight the utility of using multiple agents for exploration tasks, there the single robot became lost and the time taken to recover meant the AR.Drone mapped significantly less (1200 points) of the environment. In the 3 robot case and this resulted in some additional time but they were still able to produce a complete map of similar size to the other 3 AR.Drone experiments. More details of these experiments including videos can be found at the following web-page: `http://cgi.csc.liv.ac.uk/~rmw/drone.html`

6 Conclusion

In this paper we have presented a framework for multi-agent research using a low-cost quadcopter platform. We demonstrate the use of the framework on two typical multi-robot tasks: collision avoidance and environment exploration. Future work will involve improving the robustness of the collision avoidances to tracking loss of a single/multiple drones by defining un-safe zones to avoid when a drone has become lost. As mentioned we would also like to investigate more complex interest point extraction and auctioning mechanisms to improve the consistency and reliability of our exploration system. The main limitation of the framework is the reliance of wireless communication for state estimation and control, this limits the range of the AR.Drones and introduces problems of latency and data loss. The release of many low-cost ARM-based development boards such as the Raspberry Pi[3] offer low cost and low power-consumption solutions to adding on-board processing to the AR.Drone. This would allow us to move time critical tasks such as tracking, control and collision avoidance on-board the AR.Drone.

References

1. Van den Berg, J., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on. pp. 1928–1935. IEEE (2008)
2. Bills, C., Chen, J., Saxena, A.: Autonomous mav flight in indoor environments using single image perspective cues. In: Robotics and automation (ICRA), 2011 IEEE international conference on. pp. 5776–5783. IEEE (2011)
3. Castle, R., Klein, G., Murray, D.W.: Video-rate localization in multiple maps for wearable augmented reality. In: Wearable Computers, 2008. ISWC 2008. 12th IEEE International Symposium on. pp. 15–22. IEEE (2008)
4. Conroy, P., Bareiss, D., Beall, M., van den Berg, J.: 3-d reciprocal collision avoidance on physical quadrotor helicopters with on-board sensing for relative positioning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. Submitted (2013)

[3] `http://www.raspberrypi.org/`

5. Dijkshoorn, N.: Simultaneous localization and mapping with the ar. drone. Ph.D. thesis, Masters thesis, Universiteit van Amsterdam (2012)
6. Engel, J., Sturm, J., Cremers, D.: Camera-based navigation of a low-cost quadrocopter. IMU 320, 240 (2012)
7. Engel, J., Sturm, J., Cremers, D.: Scale-aware navigation of a low-cost quadrocopter with a monocular camera. Robotics and Autonomous Systems (2014)
8. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. The International Journal of Robotics Research 17(7), 760–772 (1998)
9. Hennes, D., Claes, D., Meeussen, W., Tuyls, K.: Multi-robot collision avoidance with localization uncertainty. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1. pp. 147–154. International Foundation for Autonomous Agents and Multiagent Systems (2012)
10. Kendoul, F.: Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. J. Field Robot. 29(2), 315–378 (Mar 2012)
11. Klein, G., Murray, D.: Parallel tracking and mapping for small ar workspaces. In: Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on. pp. 225–234. IEEE (2007)
12. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on. vol. 3, pp. 2149–2154. IEEE (2004)
13. Meyer, J., Sendobry, A., Kohlbrecher, S., Klingauf, U., von Stryk, O.: Comprehensive simulation of quadrotor uavs using ros and gazebo. In: Simulation, Modeling, and Programming for Autonomous Robots, pp. 400–411. Springer (2012)
14. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: ICRA workshop on open source software. vol. 3 (2009)
15. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: Computer Vision–ECCV 2006, pp. 430–443. Springer (2006)
16. Sa, I., He, H., Huynh, V., Corke, P.: Monocular vision based autonomous navigation for a cost-effective mav in gps-denied environments. In: Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME International Conference on. pp. 1355–1360. IEEE (2013)
17. Sanchez-Lopez, J.L., Pestana, J., de la Puente, P., Campoy, P.: Visual quadrotor swarm for imav 2013 indoor competition. In: ROBOT2013: First Iberian Robotics Conference. pp. 55–63. Springer International Publishing (2014)
18. Thrun, S., Burgard, W., Fox, D., et al.: Probabilistic robotics, vol. 1, pp. 48–54. MIT press Cambridge (2005)
19. Van Den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Reciprocal n-body collision avoidance. In: Robotics research, pp. 3–19. Springer (2011)
20. Yamauchi, B.: A frontier-based approach for autonomous exploration. In: Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on. pp. 146–151. IEEE (1997)
21. Yamauchi, B.: Frontier-based exploration using multiple robots. In: Proceedings of the second international conference on Autonomous agents. pp. 47–53. ACM (1998)
22. Zickler, S., Laue, T., Birbach, O., Wongphati, M., Veloso, M.: Ssl-vision: The shared vision system for the robocup small size league. In: RoboCup 2009: Robot Soccer World Cup XIII, pp. 425–436. Springer (2010)