Planning the Behaviour of Low-Cost Quadcopters for Surveillance Missions

Sara Bernardini and **Maria Fox** and **Derek Long**
Department of Informatics
King's College London
London, UK, WC2R 2LS
firstname.lastname@kcl.ac.uk

Abstract

Micro Aerial Vehicles (MAVs) are increasingly regarded as a valid alternative to UAVs and ground robots in surveillance missions and a number of other civil and military applications. Thanks to their light weight, small size and aerodynamics characteristics, MAVs are greatly flexible and manoeuvrable, easily portable and deployable, and safe for close interaction. Research on autonomous MAVs is still in its infancy and has focused almost exclusively on integrating control and computer vision techniques to achieve reliable autonomous flight. In this paper, we describe our approach to using automated planning in order to elicit high-level intelligent behaviour from autonomous MAVs engaged in surveillance applications. Planning offers effective tools to handle the unique challenges faced by MAVs that relate to their fast and unstable dynamics as well as their short range, low endurance, and small payload capabilities. We focus on a specific MAV, the "Parrot AR.Drone2.0" quadcopter, and a specific surveillance application, Search-and-Tracking, which involves searching for a mobile target and tracking it after it is found.

1 Introduction

In the last few years, there has been a considerable amount of work in developing autonomous Micro Aerial Vehicles (MAVs). Thanks to their light weight, small size and aerodynamics characteristics, MAVs afford greater flexibility and manoeuvrability than traditional unmanned aircraft. Being capable of hovering, flying at very low speed, turning with a small radius, as well as quickly taking-off and landing, MAVs can perform complex and aggressive manoeuvres (Lupashin and D'Andrea 2012; Purwin and D'Andrea 2009). The emergence of low-cost MAVs, to the point that they can be considered disposable devices, has allowed for rapid prototyping and testing of innovative techniques to support autonomous behaviour. Despite their relatively new appearance in the commercial market, MAVs have already being used in a number of military and civilian missions, including surveillance operations, exploration, weather observation, disaster relief coordination, and civil engineering inspections. Similarly to UAVs, MAVs can be used in any situations in which it would be difficult or dangerous to send a human. However, thanks to their reduced dimensions, they can also be used in scenarios that are inaccessible to large unmanned vehicles, such as indoors, in cluttered outdoor scenarios and in the vicinity of people.

As for other robotic artefacts, building an autonomous MAV is a challenging task because it requires the integration of techniques developed in different fields, including control theory, navigation, real time systems, computer vision and physics. However, in comparison with ground robots, MAVs present unique characteristics that make devising algorithms for autonomy particularly demanding (Bachrach et al. 2010). First, these vehicles are difficult to control as they are inherently unstable systems with fast dynamics. Second, given their reduced dimensions and light weight, MAVs have a restricted payload, i.e reduced computational power as well as noisy and limited sensors. Third, the life of a MAV battery is usually short and allows continuous flight for a limited period, ranging from a minimum of a few minutes to a maximum of around two hours. Finally, MAVs are almost always in motion, as landing and taking-off are expensive operations that cannot be performed too often within a mission. Considering all these factors, MAVs appear particularly well-suited to operate in situations in which there is very little stability, information is changing rapidly and decisions about what action to perform and how to coordinate with other MAVs must be made almost instantaneously. Effective management of uncertainty, restricted resources and tight deadlines are crucial requirements for creating an autonomous and intelligent MAV.

To date, research concerning MAVs has mainly focused on perception and control. In particular, in GPS-denied and contained environments, such as indoor scenarios, the focus has been on perception, with both vision sensors such as cameras (Engel, Sturm, and Cremers 2012b; Bills, Chen, and Saxena 2011; Zingg et al. 2010) and non-vision sensors such as laser range scanners, sonar, and infra-red being widely investigated (Roberts et al. 2007; Achtelik et al. 2009), and on navigation (Engel, Sturm, and Cremers 2012b; Bills, Chen, and Saxena 2011; Courbon et al. 2009; Mori, Hirata, and Kinoshita 2007). Conversely, in outdoor domains, where MAVs are subject to wind and turbulence, the emphasis has been on control, stabilisation and pose estimation (Abbeel et al. 2007; Moore et al. 2009; Fan et al. 2009). The use of automated planning to underpin the behaviour of MAVs has received little attention so far, with some effort devoted to path-planning and trajectory-planning (Bachrach et al. 2010; He, Prentice, and Roy 2008; Hehn and D'Andrea 2012), but none to task planning (to the best of our knowledge).

In contrast with this trend, we believe that task planning carries a significant potential for the development of intelligent MAVs as their short range, low endurance, and small payload capabilities pose challenges that cannot be met by simple low-level control algorithms. In our previous work (Bernardini et al. 2013), we used task planning to underpin the high-level operation of an autonomous UAV engaged in surveillance operations, and particularly in Search-and-Tracking (SaT) missions involving searching for a mobile target and tracking it after it is found. By developing a simulator of a fixed-wing UAV undertaking SaT operations, we demonstrated the effectiveness of our planning-based approach to SaT in comparison with static strategies. In this paper, we show that this approach generalises well to MAVs involved in SaT operations. In so doing, we offer a proof-of-concept that complementing computer vision and control techniques with automated planning mechanisms is a viable way of enabling MAVs to exhibit intelligent and autonomous behaviour. Although we focus here on a specific MAV, the "Parrot AR.Drone2.0" quadcopter (see Figure 1), and a specific surveillance application, SaT, we believe that our approach generalises to other MAVs and robotic vehicles as well as different surveillance operations. Ultimately, our goal is to demonstrate that our planning-based approach can be used in any surveillance scenario to underpin the behaviour of an observer that, knowing the dynamics of the target but not its intentions, needs to quickly make control decisions in the face of such uncertainty and under tight resource constraints.

(a) Indoor (b) Outdoor

Figure 1: Parrot AR.Drone

The work presented in this paper is still in progress, so currently we cannot report on extensive experimental results, which we have not performed yet. However, based on our previous experience on SaT missions with UAVs and the very limited experiments that we have performed, we are confident that our approach will prove successful.

The organisation of the paper is the following. In Sections 2 and 3, we describe the characteristics of the AR.Drone2.0 and of SaT missions, respectively. In Section 4, we give an overview of our planning-based approach to SaT using the quadcopter as the observer. We then explain our approach in detail in Section 5. In Sections 6 and 7, we describe how we have implemented our method and the intended set-up for our experiments. We conclude with final considerations and a description of future work in Section 8.

2 Hardware Platform: Parrot AR.Drone

We use the AR.Drone quadcopter[1] as our prototyping hardware platform and as an example of MAV to which our approach can be applied. The AR.Drone is a low-cost and light-weight quadcopter that was launched in 2010 by the French company "Parrot" as a high-tech toy for augmented reality games. Since then, it has been increasingly popular in academia and research organisations as an affordable test platform for MAV demonstrations (see projects at Cornell University[2] (Bills, Chen, and Saxena 2011), Technische Universität München[3] (Engel, Sturm, and Cremers 2012b) and RMIT University in Melbourne[4] (Graether and Mueller 2012), just to mention a few of them). With respect to other modern MAVs, the AR.Drone has a number of advantages: it is sold at a very low cost, is robust to crashes, can be used very close to people and its onboard software provides reliable communication, stabilisation, and assisted manoeuvres such as take-off and landing.

The AR.Drone is composed of a carbon-fibre tube structure, plastic body, high-efficiency propellers, four brushless motors, sensor and control board, two cameras and indoor and outdoor removable hulls. It has a weight of respectively 380 g and 420 g with the indoor and the outdoor hull, and maximum dimensions of 52×52 centimetres. The drone affords an average speed of 5 meters per second, with maximum speed reported as 11 meters per second. Its lithium polymer battery provides enough energy up to 13 minutes of continuous flight and takes around 90 minutes to recharge.

The quadcopter is equipped with an ARM9 processor running at 468MHz with 128 MB of DDR RAM running at 200MHz and a WiFi network. The drone acts as a wireless server and assigns itself (via its DHCP server) a fixed IP address through which it is possible to communicate with it.

The sensory equipment of the AR.Drone is composed of: i) 3-axis accelerometer; ii) 2-axis gyroscope (for measuring/maintaining orientation); iii) 1-axis yaw precision gyroscope; iv) ultrasound altimeter (for vertical stabilisation); and v) two cameras. The first camera is aimed forward, covers a field of view of $73.5° \times 58.5°$, has a resolution of 1280×720 pixels (720p) and its output is streamed to a laptop at 30 fps. The second camera aims downward, covers a field of view of $47.5° \times 36.5°$ and has a resolution of 320×240 pixels (QVGA) at 60 fps.

The onboard software uses the down-looking camera to estimate the horizontal velocity and the other sensors to control the roll Φ and pitch Θ, the yaw rotational speed $\dot{\Psi}$ and the vertical velocity $\dot{z}$ of the quadcopter according to an external reference value (see Figure 2).

The onboard software provides three communication channels with the drone:

- *Command* channel: used to send commands to the drone (e.g. take-off, land, calibrate sensors, etc.) at 30 Hz.

[1]http://ardrone2.parrot.com/
[2]http://mav.cs.cornell.edu/
[3]https://vision.in.tum.de/research/quadcopter
[4]http://exertiongameslab.org/projects/joggobot

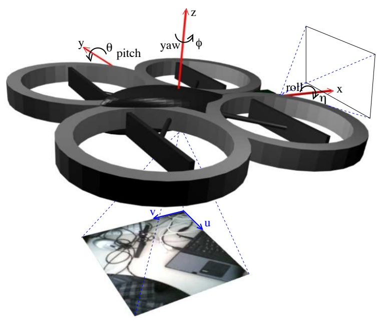

Figure 2: Coordinate system of the AR.Drone

- *Navdata* channel: providing information about the drone status (flying, calibrating sensors, etc.) and pre-processed sensory data (current yaw, pitch, roll, altitude, battery state and 3D speed estimates) at 30 Hz. Since the drone can run a simple analysis of the images from the frontal camera and search for specially designed tags in the images, the navdata channel contains estimates of the tags' positions if such tags have been detected.
- *Stream* channel: providing images from the frontal and/or bottom cameras. Images from both cameras cannot be acquired at the same time, and switching between cameras requires 300 ms.

3 Mission Description: Search-and-Tracking

Although the AR.Drone is suitable for a number of surveillance applications, we focus here on SaT missions. SaT is the problem of searching for a mobile target and tracking it after it is found. Solving this problem effectively is important as SaT is a common component task in many surveillance operations. SaT missions are plausible both outdoors and indoors as well as both in adversarial and cooperative contexts. We can imagine, for example, a drone chasing a suspect in a parking lot or escorting a worker who performs risky tasks in a factory.

In our case, the observer is the AR.Drone, while the target can be a person or any moving object that proceeds at a speed compatible with the speed of the drone. As we do not deal with object recognition, we assume that our target is identified by a specific tag known in advance by the drone. In addition, we assume that the configuration of the environment that confines the movement of the target is known to the drone. If the SaT mission takes place outdoors, we assume that the road network and the terrain types are known, whereas if the mission happens indoors, we assume that the topology of the indoor space is known to the quadcopter. Finally, we assume that the target has a destination and, to reach such destination, it follows the road network when it moves outdoors and the path structure (corridors, stairs, etc.) when it moves indoors. The target does not perform evasive actions by attempting to use features in the environment for concealment or to deviate from the originally intended path. This is a plausible assumption as the target might be cooperating with the drone or simply unaware of its presence. In practice, as described in Section 6, in our initial experiments, we have used a robotic wheeled vehicle as the target and assume that it follows a road network when it moves. All these assumptions are in line with our previous work relating to a fixed-wing UAV involved in a SaT mission (Bernardini et al. 2013). This helped us to generalise the solution adopted for the UAV case to the MAV case.

The objective of a SaT mission is to follow the target to its destination. In general, a SaT mission proceeds in two phases, which interleave until the target stops or until the observer acknowledges it has lost the target irrevocably and abandons the mission. These phases are:

- *Tracking*: the drone simply flies over the target, observing its progress; and
- *Search*: the drone has lost the target and flies a series of manoeuvres intended to rediscover the target.

Once the target is rediscovered, the drone switches back to the tracking mode.

4 Planning-based Approach to SaT

As described in the previous section, any SaT mission consists of two phases: tracking and search.

We manage the tracking phase through a reactive controller equipped with vision capabilities: the problem is simply one of finding a route that maximises observations of the target. However, when the drone fails to observe the target, it must attempt to rediscover it. How this is achieved depends on the time since the observer last observed the target. For a short period after losing the target, the drone can simply track the predicted location of the target, since the target cannot move fast enough to significantly deviate from this prediction. However, after a period (whose length depends on the speed of the target, the field of view of the imaging equipment, and the observation probability), it will be necessary to make a more systematic effort to rediscover the target by directing the search into specific places. This is when task planning comes into play. Right after loosing track of the target, the following information is available to the drone: the last known location and velocity of the target, the average velocity over the period the target has been tracked, the map of the environment and the current position of the drone itself. Based on this information, it is possible to formulate the task of rediscovering the target as a *planning problem*: a search plan is constructed from a set of candidate manoeuvres (for example, spirals, lawnmowers, polygonal figures, hovering, etc.) that can be arranged in a sequence to attempt to optimise the likelihood of rediscovering the target. If, while flying this search plan, the target is rediscovered, the observer switches back to tracking mode.

As mentioned in the Section 1, we have already demonstrated the effectiveness of a planning-based approach to SaT in our previous work concerning a single UAV tracking and searching for a ground vehicle through a mixed urban, sub-urban and rural landscape (Bernardini et al. 2013). The solution we proposed is to track the target reactively while it is in view and to plan a recovery strategy that relocates the target every time it is lost, using a high-performing *automated planning* tool. The recovery strategy involves flying a sequence of standard search patterns, i.e. spirals and lawnmowers, at particular times and in specific locations of the geographical area considered. The planning problem consists of deciding where to search and which search patterns to use in order to maximise the likelihood of recovering the target. Our results indicated that this approach is successful and certainly outperforms static search strategies. By using our technique, we have been able to tackle SAT problems on a large scale, a 100 kilometre square area, which represents a significant challenge to the problem of search, far beyond the capabilities of current alternatives.

The use of quadcopters in SaT missions makes the planning problem even more challenging than when UAVs are used. In fact, the planning model must reflect the specific constraints pertaining to the quadcopter's capabilities, and the planner must reason within such constraints in order to formulate an effective search strategy. These constraints relate to safety considerations in the use of the drone, the fact that the drone is always in motion, and finally the drone's limited computational power, cheap and noisy sensors, and short battery life. We believe that dynamically planning the behaviour of a quadcopter in the face of all these constraints instead of using a static policy is even more crucial than in the case of UAVs. Only with a carefully crafted strategy, can the quadcopter achieve satisfactory performance within the limitations posed by its technical characteristics.

As our project is still in progress, we have not yet fully handled all the constraints pertaining to quadcopters in our current implementation. Instead, as reported in detail in the next two sections, we have focused on translating the planning approach to SaT used for fixed-wing UAVs to MAVs and robustly implementing such method on a real quadcopter, the AR.Drone. However, modelling the specificities of the drone and exploiting them to formulate more effective plans is part of our future work.

5 Search as Planning

If the drone loses the target beyond the short period for which it tracks its predicted location, it must follow a search strategy to attempt to rediscover it. We propose to exploit hovering, and flight patterns, such as zig-zags, spirals, lawnmowers, polygonal figures and patrolling around obstacles, as the building blocks for a search plan that attempts to maximise the expectation of rediscovering the target. The challenge, then, is to decide where these search patterns should be deployed. One possibility is to use a fixed strategy of simply flying some standard configuration of these patterns over the area where the target was lost. However, a more interesting approach is to see the problem of selection and sequencing of search patterns as a *planning problem*: each search pattern can be assigned a value corresponding to the expectation of finding the target in a search of that area and then the drone can select a sequence of search patterns, linking them together with a series of flight paths, in order to maximise the accumulated expectation of rediscovery within the limits of its resources.

As we have already pointed out in our previous work (Bernardini et al. 2013), despite the inherent uncertainty in the situation, the problem is *deterministic*, since the uncertainty arises in the position of the target and, if the target is found, the plan ceases to be relevant. Therefore, the plan is constructed entirely under the assumption that the target remains undiscovered. Somewhat counter-intuitively, "plan-failure" corresponds to the situation in which the target is found, counter to the assumption on which the plan is based. However, on failure of this assumption, the plan becomes irrelevant and the drone switches back to the tracking mode.

5.1 Planning Domain

The domain model for the search problem has a straightforward structure. There are actions for: i) taking-off; ii) landing; iii) hovering; iv) flying from one waypoint to another; and v) performing basic search patterns, such as spirals, lawnmowers, zig-zags and polygonal figures. The search pattern actions all have similar forms: they each have an entry waypoint and an exit waypoint and the effect, other than to move the drone from the entry to the exit point, is to increase the reward (which is the accumulated expectation of finding the target). The actions are durative and their duration is fixed in the problem instance to be the correct (computed) value for the execution of the corresponding search. The search patterns can only be executed so that they coincide with a period during which the target could plausibly be in the area the pattern covers. This is calculated by considering the minimum and maximum reasonable speeds for the target and the distance from where the target was last observed. The reward is more complicated and is discussed in detail below, but the problem instance associates with each pattern a reward, using a time-dependent function.

As an example of how search actions are modelled in PDDL2.2, the following is the description of the action `doSmallLawnmower`, which specifies the location and availability of a lawnmower search pattern, the time it takes to complete it and the reward available.

```
(:durative-action doSmallLawnmower
:parameters (?from ?to - waypoint ?p - smallLawnmower )
:duration (=?duration (timefor ?p))
:condition (and (at start (beginAt ?from ?p))
                (at start (endAt ?to ?p))
    (at start (at ?from))
    (at end (active ?p)))
:effect (and (at end (at ?to))
             (at start (not (at ?from)))
             (at end (increase (reward) (rewardof ?p)))))
```

The time-dependent reward function is managed by timed-initial fluents in the problem specification that change the reward of the patterns as time progresses. The shape of the function is constructed to represent an approximate lifted Gaussian distribution, with no reward until the target could plausibly have arrived at the search area and no reward after the target is unlikely to be still present in the area. Between these extremes, the reward peaks at the point where the target would be in the centre of the search pattern if driving at average speed. The model can be modified by adding more or fewer intermediate points in the step function approximation of the distribution. To ensure that the planner does not exploit search patterns when there is no reward associated with them, the patterns are only made active during the period when the distribution is positive, using timed initial literals that are asserted and retracted at the appropriate times.

In future work, we plan to enrich the planning domain by adding, for example, a model of the drone's battery. As the battery has a very limited life, the planner needs to take into account the cost of performing different actions in terms of their battery consumption.

5.2 Planning Problem

The problem has no goal, but the plan metric measures the value of the plan in terms of the accumulated expectation of finding the target. A few examples of problems of this sort have been considered before (for example, one variant of the satellite observation problem used in the 3rd International Planning Competition (Long and Fox 2003) had this character) and it is typically the case that bench-mark planners generate empty plans, ignoring the metric. We discuss below our management of this problem.

To create the initial states for our planning problems, we have to manage two tasks: (i) identifying *candidate search patterns*; and (ii) assigning appropriate *rewards* to them. The first task is made difficult by the fact that there are infinitely many patterns that could be used, while the second is made difficult because of the lack of knowledge about the intentions of the target. In what follows, we show how we manage these tasks for outdoor SaT missions, on which we are currently focusing in our experiments. However, similar considerations can be made for indoor spaces.

To address the first problem we observe that the planner can only consider a finite subset of search patterns and, since we want to perform planning in real time, is limited to being able to consider a reasonably small number of candidates. Therefore, we generate a sample of possible search patterns by randomly selecting a set of shapes (circles, rectangles and polygons) and placing them onto the general search area. There are three steps involved in this (see (Bernardini et al. 2013) for additional technical details on each step)):

1. *Circular sector construction:* we use as our general search area a circular sector that is centred on the last known location of the target and extends outwards with its symmetry axis aligned with the average bearing of the target over the period the target has been observed. The sector extends outwards for a distance whose exact value depends on the total area included in the sector and the relative time required to fly a search pattern of a given area. There is a point at which the area where the target could be present is so much larger than the area that the drone can search, during a period when the target could be present, that the expectation of finding the target diminishes to a negligible value. The angle subtended by the sector reflects the degree of uncertainty in the heading of the target. In general, a target will follow the direction that is forced on it by the paths it uses, but the average bearing will converge, over time, on the direction from the origin to the destination. The longer the target is observed, the closer will this convergence become.
2. *Sampling:* once the relevant sector is identified, we then sample points using a probability distribution laid over the sector. This distribution is based on the density of roads across the sector, which is measured by using a fine-mesh grid and counting the number of significant roads within each grid cell, the terrain type (urban, suburban, mountainous, forested, rough or open rural ground) and the distance from the symmetry axis and from the last known location of the target. The distribution decays linearly with distance from the origin, linearly away from the symmetry axis and is weighted by values for terrain type and road density. Although the density of patterns decays away from the origin, the effect is muted because the relative areas available for selection are proportional to the distance from the origin.
3. *Search pattern generation:* finally, we decide the type of pattern to use for each point: we favour spirals for covering an area of high density road network, particularly in urban or suburban terrain, and lawnmowers when attempting to search over a more elongated stretch covering a major road and including some possible side roads. For spirals, we select a radius based on the width of the sector at that point and the road network density. For lawnmowers, we select an orientation and then width and length. The orientation is based on the road network and is aligned to follow major roads or high densities of roads, while the width and length are determined by examining the road network and probability distribution.

As for the second problem, i.e. *reward selection*, we compute a shortest and longest time of arrival for the target by considering an average speed and variation in speed over the path from the origin to the pattern. In principle, this mechanism should use the road map to identify shortest paths, but this is too costly to compute in real time, so we instead sample the terrain along the straight line from the origin to the leading and far edges of the pattern. This is used to as a guide to the likely speed of the target on this path. In practice, if the straight line path traverses rural areas then the target will either have to use smaller roads or else deviate from the direct path in order to exploit more major roads. In either case, the target will arrive at the target later than if the direct path is through suburban terrain. On the other hand, if the terrain is urban then speed will be constrained by traffic laws and other road users. The earliest and latest times are used to set up a value function, with these as the limits of the reward (outside this range the pattern is awarded no value). The peak reward is calculated as a proportion of the probability density in the distribution across the intersection of the sector and the annulus centred at the same origin and with edges coinciding with the boundaries of the search pattern. This represents a surrogate for the total available probability density across the time period covered by the search pattern, although it is clearly an approximation.

Once the initial state is prepared, we can plan.

5.3 Planning Mechanism

We exploit the period in which the quadcopter tracks the predicted location of the target to perform planning. In our application, we use an off-the-shelf planner called OPTIC (Benton, Coles, and Coles 2012) to build plans for the drone. OPTIC is a version of POPF (Coles et al. 2010) specifically designed to perform anytime, cost-improving search.

We use a time-bounded search limited to 10 seconds because, having the drone a very limited battery life, we are in a time-critical situation. The planner will typically find a first solution very easily, since the empty plan is already a feasible solution, but it will then spend the additional time improving on this by adding further search patterns to the plan, or trying different collections of patterns. The search uses a weighted-$A^\star$ scheme with steadily changing weights in a tiered fashion (see (Benton, Coles, and Coles 2012) for details). The plans produced in this way are monotonically improving, so the final plan produced is the one we select for execution. We use OPTIC because it is very fast at producing its first solution and provides an any-time improvement behaviour. LPG would offer similar characteristics but our experiments indicated OPTIC was better for our problems.

The plan is dispatched via a simple controller, action by action. At the conclusion of execution of the plan, in principle, two alternatives are viable for the drone, depending on how long has passed since the target was last seen: spending more time generating a new plan, or abandoning the search. We always make the drone abandon the search and land at this point in our current implementation.

So far we have implemented a static policy, which is based on replanning. The observer enters a planning phase every time it loses the target. We allow the observer 10 seconds for planning (it can be configured to have longer) at each of these points. We intend to later replace this planning behaviour with an essentially instantaneous action selection using a *learned policy*. Learning a policy, to replace the static policy currently implemented, is a future goal for our work.

6 Implementation

In order to carry out a SaT mission, the AR.Drone needs to combine the abstract deliberative skills illustrated in the previous sections with low-level control and vision capabilities. In particular, the drone needs different skills in the two phases of a SaT mission:

- For the tracking phase, the drone needs to be able to accomplish the following tasks:
  - Tag recognition: we assume that our target is identified by a specific tag, and therefore, the drone needs to be able to recognise tags from a distance based on the video stream coming from its cameras. We use computer vision algorithms to solve this problem.
  - Tag following: once the drone has recognised the tag corresponding to the target, it needs to follow it reactively. We achieved this by implementing a Proportional-Integral-Derivative (PID) controller that works based on the navigation data provided by the drone's driver.
- For the search phase, the drone essentially needs to be able to fly autonomously to a given position as the plan formulated by the planner specifies waypoints to fly to and search patterns to execute, which are in turn sets of waypoints to be reached in a particular sequence. Autonomous flight requires a combination of low-level control capabilities, such as maintaining attitude, stabilisation, and compensation for disturbances, and high-level control skills, such as compensation for drift, localisation and mapping, obstacle avoidance and navigation. Conveniently, the AR.Drone provides built-in low-level control, so we have focused on high-level control only. Since we currently ignore obstacle avoidance, our implementation provides capabilities for localisation and mapping, and navigation. The navigation system that we use is composed of three major components: monocular simultaneous localisation and mapping (SLAM) for visual tracking, Extended Kalman Filter (EKF) for data fusion and prediction, and PID control for pose stabilisation and navigation. We describe them in detail below.

We have implemented our application within the Robot Operating System[5] (ROS) framework (Quigley et al. 2009). ROS is an open-source, meta-operating system for robots and provides basic services such as hardware abstraction, low-level device control, implementation of commonly-used functionality and message-passing between processes. ROS facilitates code reuse in robotics as it offers tools for finding, building, and running code across multiple computers. A ROS application can be seen as a peer-to-peer network of processes, called *nodes*, that perform computation and communicate with each other by passing *messages*, which are data structures with typed fields. A node sends a message by *publishing* it to a given *topic*, which is simply a string used to identify the content of the message. A node that is interested in a certain kind of data will *subscribe* to the appropriate topic. A node can also offer a *service* to the other nodes, which send *request* messages for such service and wait for *replies*. In the spirit of ROS, we have leveraged existing packages for implementing our SaT application. In particular, we built on the following ROS packages:

- ARDRONE_AUTONOMY[6]: this is a ROS driver for the Parrot AR.Drone based on the official AR.Drone SDK version 2.0 and developed by the Autonomy Lab at Simon Fraser University. The driver's executable node, ARDRONE_DRIVER, offers a number of features:
  - it converts all raw sensor readings, debug values and status reports sent from the drone into standard ROS messages, which can then be used and interpreted by the other ROS nodes involved in the application;
  - it allows to send control commands to the drone for taking off, landing, hovering and specifying the desired linear and angular velocities; and
  - it provides additional services such as led and flight animations.
- AR_RECOG[7]: this is a ROS vision package developed by the Robotics, Learning and Autonomy at Brown University that allows the drone to recognise specific tags as well as to locate and transform them in the image-space. This package is based on the ARToolKit[8], which is a well-

[5]http://www.ros.org

[6]http://wiki.ros.org/ardrone_autonomy

[7]http://wiki.ros.org/ar_recog

[8]http://www.hitl.washington.edu/artoolkit/

established software library for building augmented reality applications.

- TUM_ARDRONE[9]: this ROS package is based on ARDRONE_AUTONOMY and has been developed by the Computer Vision Group of Technische Universität München. It implements autonomous navigation and figure flying in previously unknown and GPS-denied environments (Engel, Sturm, and Cremers 2012a; Engel, Sturm, and Cremers 2012b). The package is composed of two main nodes:
  - DRONE_STATE-ESTIMATION: This node provides the drone with SLAM capabilities. SLAM is the process of using onboard sensors to estimate the vehicle's position and then using the same sensor data to build a map of the environment around the vehicle. In particular, this node implements a SLAM algorithm based on Parallel Tracking and Mapping (PTAM) (Klein and Murray 2007). Finally, this node implements an EKF for state estimation and compensation of the time delays in the system arising from wireless LAN communication.
  - DRONE_AUTOPILOT: This nodes implements a PID controller for pose stabilisation and navigation. Based on the position and velocity estimates from the EKF, the PID control allows us to steer the quadcopter towards a desired goal location expressed in a global coordinate system. In addition, this node implements a scripting language to control the drone by sending it commands for initialising PTAM, taking off, landing, going to a specific waypoint and moving in a given direction with respect to the current position.
- TUM_SIMULATOR[10]: this is a ROS package that implements a Gazebo[11] simulator for the AR.Drone developed by the Computer Vision Group of Technische Universität München. Since this Gazebo simulator can be used as a transparent replacement for the quadcopter, it allows us to develop and evaluate algorithms more quickly because they do not need to be run on the real quadcopter.

We have implemented two additional ROS nodes for our application: AR_TAG_FOLLOWING and AR_PLANNER. The first node implements a PID controller that, based on the messages received from the AR_RECOG package, allows the drone to follow the detected tag. The second node, AR_PLANNER, wraps the OPTIC planner and allows us to integrate it with the rest of the system. The planner is invoked by the node AR_TAG_FOLLOWING when the tag is not detected by an amount of time sufficient for the node to conclude that the target is lost. After the planner has built the plan, the plan's actions are translated in the scripting language provided by the DRONE_AUTOPILOT node, which then imparts the commands to the ARDRONE_DRIVER node.

As ARDRONE_AUTONOMY is the driver for the drone, we use this package both for the tracking and for the search phase. We use the AR_TAG_FOLLOWING and AR_RECOG nodes for the tracking phase and the DRONE_STATE-ESTIMATION and DRONE_AUTOPILOT for the search phases. The AR_PLANNER node acts as the interface between the the tracking and the search phase. Figure 3 shows the active ROS nodes and topics for the search phase.

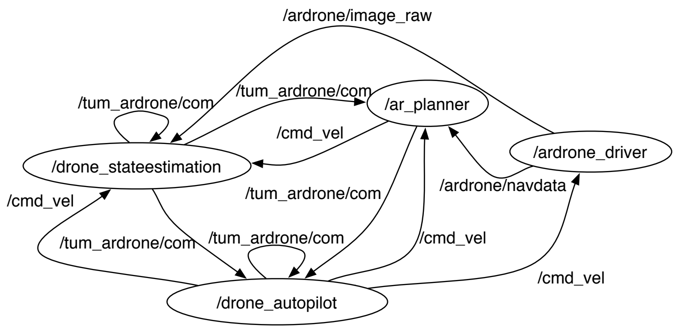

Figure 3: ROS nodes and topics involved in the search phase

Although using ROS and leveraging existing packages for the AR.Drone facilitated the implementation of our application, working with a real quadcopter remains a time-consuming and challenging task for several reasons. First, flying a physical object (in contrast with a simulated one) requires to solve several implementation and low-level issues and to manually tune of a number of parameters. In addition, despite the fact that the Parrot AR.Drone is considerably more robust than other low-cost quadcopters, it is yet an unstable system and its control is not as easy as the control of a ground robot. Finally, the majority of existing ROS packages for controlling the drone do not work for both versions of the drone (AR.Drone1.0 or 2.0) and are not supported by the latest versions of ROS and Gazebo. This required us to modify the existing packages and make them compatible with the new versions of the drone, as well as the ROS and Gazebo frameworks.

7 Experimental Set-up

In order to evaluate our approach to SaT in outdoor scenarios, we intend to create an indoor analogue of the outdoor setting. As our project is still ongoing, we have not yet completed the set-up of our experiments and, consequently, we have not yet performed extensive experiments of our approach. In what follows, we describe the design of the experimental set-up, while we will report on the actual experimental results in future publications.

We use the AR.Drone 2.0 as the observer and a LEGO Mindstorms NXT assembled in the shape of a wheeled vehicle as the target (see Figure 4). This vehicle is able to detect and follow a line marked on the floor and is identified by the drone through a ARTag attached to it (see Figure 5).

We use markers on the floor to create a road network and boxes to create spaces where the drone is unable to see the vehicle. We currently assume that the drone flies at an altitude where there are no obstacles to interfere with its flight. However, we plan to revisit this assumption at a later stage and add walls with windows, poles and other obstacles in

[9] http://wiki.ros.org/tum_ardrone
[10] http://wiki.ros.org/tum_simulator
[11] http://gazebosim.org

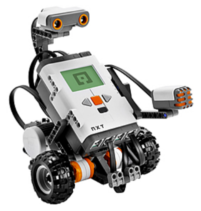

Figure 4: Target vehicle used in our experiments

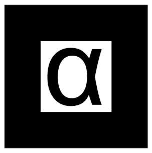

Figure 5: ARTag used to identify the vehicle

the environment. This will require additional control capabilities for our drone. Finally, we assume that the drone knows the configuration of the environment. Although the TUM_ARDRONE and AR_TAG_FOLLOWING packages do not require a priori knowledge of the environment, the planner needs to know the configuration of the road network for generating plausible candidate search patterns. In future work, we intend to experiment with a different set of assumptions. For example, instead of knowing the road network, the drone might know the motion model of the target and reason about such model in order to compute candidate search patterns.

In our previous work (Bernardini et al. 2013), we showed that our planning approach to SaT, i.e. viewing the search problem as a planning problem in which search patterns must be selected and sequenced to maximise the expectation of rediscovering the target, is successful and certainly outperforms static search strategies. Such work presents a number of different factors with respect to the work reported in this paper: it focuses on a fixed-wing aeroplane as the observer and a car as the target, the vehicles are free to move within a large geographical area and the experimental results were obtained by running a computer simulation. At this stage, we cannot yet report on extensive experiments of our SaT approach in the new setting, as our work is still in progress. However, our initial experiments, limited to testing the different modules of our system, seem to suggest that the promising results we obtained in our original scenario could translate to the new setting.

8 Conclusions and Future Work

In this paper, we describe our ongoing work concerning the use of automated planning for the high level operation of a low-cost and light-weight quadcopter engaged in SaT missions. We formulate the search problem as a planning problem in which search patterns must be selected and sequenced to maximise the expectation of rediscovering the target.

The approach described here builds on the work of Bernardini *et al.* (2013) on planning the behaviour of a UAV that searches for a moving target across large areas and over a long time. We have demonstrated that our planning-based approach to SaT for UAVs generalises well to the case of MAVs, although the temporal and spatial scales of UAV and MAV missions are quite different. We believe that, given the specific technical characteristics of MAVs, i.e. short range, low endurance, and small payload capacities, augmenting their control architectures with planning capabilities is crucial to obtaining effective performance.

In future work, we intend to generalise our approach further to any situations in which an observer with limited resources needs to track a moving object, whose contingent behaviour is unknown, based on its motion model. Automated planning is ideally suited to generate intelligent behaviour in the face of uncertainty, tight deadlines and resource constraints.

References

[2007] Abbeel, P.; Coates, A.; Quigley, M.; and Ng, A. Y. 2007. An application of reinforcement learning to aerobatic helicopter flight. In *In Advances in Neural Information Processing Systems 19*. MIT Press.

[2009] Achtelik, M.; Bachrach, A.; He, R.; Prentice, S.; and Roy, N. 2009. Stereo Vision and Laser Odometry for Autonomous Helicopters in GPS-denied Indoor Environments. In *Proceedings of the SPIE Unmanned Systems Technology XI*, volume 7332.

[2010] Bachrach, A.; de Winter, A.; He, R.; Hemann, G.; Prentice, S.; and Roy, N. 2010. RANGE - Robust Autonomous Navigation in GPS-denied Environments. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, 1096–1097.

[2012] Benton, J.; Coles, A.; and Coles, A. 2012. Temporal Planning with Preferences and Time-Dependent Continuous Costs. In *Proceedings of the Twenty Second International Conference on Automated Planning and Scheduling (ICAPS-12)*.

[2013] Bernardini, S.; Fox, M.; Long, D.; and Bookless, J. 2013. Autonomous Search and Tracking via Temporal Planning. In *Proceedings of the 23st International Conference on Automated Planning and Scheduling (ICAPS-13)*.

[2011] Bills, C.; Chen, J.; and Saxena, A. 2011. Autonomous MAV Flight in Indoor Environments using Single Image Perspective Cues. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*.

[2010] Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling*.

[2009] Courbon, J.; Mezouar, Y.; Guenard, N.; and Martinet, P. 2009. Visual navigation of a quadrotor aerial vehicle. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5315–5320.

[2012a] Engel, J.; Sturm, J.; and Cremers, D. 2012a. Accurate figure flying with a quadrocopter using onboard visual and inertial sensing. In *Proc. of the Workshop on Visual Control of Mobile Robots (ViCoMoR) at the IEEE/RJS International Conference on Intelligent Robot Systems*.

[2012b] Engel, J.; Sturm, J.; and Cremers, D. 2012b. Camera-based navigation of a low-cost quadrocopter. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*.

[2009] Fan, C.; Song, B.; Cai, X.; and Liu, Y. 2009. Dynamic visual servoing of a small scale autonomous helicopter in uncalibrated environments. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5301–5306.

[2012] Graether, E., and Mueller, F. 2012. Joggobot: a flying robot as jogging companion. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, 1063–1066. ACM.

[2008] He, R.; Prentice, S.; and Roy, N. 2008. Planning in Information Space for a Quadrotor Helicopter in a GPS-denied Environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2008)*, 1814–1820.

[2012] Hehn, M., and D'Andrea, R. 2012. Real-time Trajectory Generation for Interception Maneuvers with Quadrocopters. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 4979–4984. IEEE.

[2007] Klein, G., and Murray, D. 2007. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*.

[2003] Long, D., and Fox, M. 2003. The 3rd International Planning Competition: Results and Analysis. *Journal of Artificial Intelligence Research (JAIR)* 20:1–59.

[2012] Lupashin, S., and D'Andrea, R. 2012. Adaptive fast open-loop maneuvers for quadrocopters. *Autonomous Robots* 33(1-2):89–102.

[2009] Moore, R.; Thurrowgood, S.; Bland, D.; Soccol, D.; and Srinivasan, M. 2009. A stereo vision system for UAV guidance. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 3386–3391.

[2007] Mori, R.; Hirata, K.; and Kinoshita, T. 2007. Vision-based guidance control of a small-scale unmanned helicopter. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2648–2653.

[2009] Purwin, O., and D'Andrea, R. 2009. Performing aggressive maneuvers using iterative learning control. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 1731–1736. IEEE.

[2009] Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; Wheeler, R.; and Ng, A. 2009. ROS: an open-source Robot Operating System. In *Proceedings of the International Conference on Robotics and Automation (IJCAI)*.

[2007] Roberts, J.; Stirling, T.; Zufferey, J.; and Floreano, D. 2007. Quadrotor using minimal sensing for autonomous indoor flight. In *European Micro Air Vehicle Conference and Flight Competition (EMAV2007)*.

[2010] Zingg, S.; Scaramuzza, D.; Weiss, S.; and Siegwart, R. 2010. MAV navigation through indoor corridors using optical flow. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, 3361–3368.