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 low-cost alternative to UAVs and ground robots in surveillance missions and a number of other civil and military applications. 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 low endurance and small payload capabilities. We demonstrate our approach by focusing on the "Parrot AR.Drone2.0" quadcopter and Search-and-Tracking missions, which involve 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 are greatly flexible and manoeuvrable, easily portable and deployable, and safe for close interaction. The emergence of low-cost MAVs, to the point where 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 been 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 cluttered outdoor settings and indoors.

In comparison with other types of robots, MAVs present unique characteristics that make devising algorithms for autonomy particularly challenging (Bachrach et al. 2010). First, these vehicles are difficult to control as they are inherently unstable systems with fast dynamics. Second, given their small 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, from a few minutes to around two hours. Given these constraining factors, MAVs typically operate in situations in which there is very little stability, information is changing rapidly and decisions on what action to perform must be made almost instantaneously. Effective management of uncertainty, restricted resources and tight deadlines is a crucial requirement for creating an autonomous and intelligent MAV. *Automated planning* technology is ideally suited to furnish MAVs with the strategic ability to meet their mission goals within the limits of their physical characteristics.

In this paper, we show how task planning can be used to underpin the high-level operation of an autonomous MAV engaged in surveillance operations. We demonstrate that complementing computer vision and control techniques with automated planning mechanisms supports intelligent and autonomous behaviour in MAVs. We validate our approach by focusing on a specific MAV, the "Parrot AR.Drone2.0" quadcopter, and a specific surveillance application, Search-and-Tracking (SaT).

2 Related Work

To date, research concerning MAVs has mainly focused on perception and control. In particular, in GPS-denied and constrained environments, such as indoor scenarios, the focus has been on perception, with cameras (Engel, Sturm, and Cremers 2012b; Bills, Chen, and Saxena 2011; Zingg et al. 2010) 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). 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 in the context of MAVs has received little at-

*The authors gratefully acknowledge support from EPSRC grant EP/J012157/1.

tention so far, with most relevant effort devoted to path-planning (Bachrach et al. 2010; He, Prentice, and Roy 2008; Hehn and D'Andrea 2012).

In typical SaT operations, the observers are fixed-wing UAVs and the targets are either stationary or move only when subjected to external disturbances, such as wind and current. An example of a typical mission is a UAV that searches for and tracks life-rafts drifting with current. State-of-the-art approaches to this kind of SaT mission make use of Recursive Bayesian Estimation (RBE) techniques for predicting the probability density function of the target's state and solve the search control problem in a greedy fashion over a short planning horizon (usually, a one-step lookahead) (Furukawa et al. 2006; Chung and Furukawa 2006; Furukawa, Durrant-Whyte, and Lavis 2007). Probabilistic-based SaT has proven successful for small-scale and short-term missions involving stationary targets in small geographical areas. However, due to their high computational cost, RBE techniques perform poorly when applied to real-world problems involving targets exhibiting complex motion in wide areas.

The optimal search method used to find the Scorpion submarine (Richardson and Stone 1971) focusses on exploring a very large area to find a stationary target. This approach grids the area and then uses an a priori probability distribution to decide which cells have the highest probability of containing the target. These probabilities are then recalculated using Bayesian updating, taking into account the outcome of exploring the highest probability cells with a range of noisy sensors. Each cell is searched using a fixed tracking pattern (such as a square spiral). Later work (Furukawa et al. 2012) goes beyond the grid-based method, using irregularly-shaped elements to structure the search region, resulting in greater efficiency.

Our approach is most closely related to (Ablavsky and Snorrason 2000), in which simple geometric shapes are used to decompose the search region and the optimal visiting order is then determined. Their method focusses on optimising coverage patterns for different pattern types and shape parameters (eg: eccentricity of an elipsoid shape), but is restricted to path planning, while our approach uses fixed patterns and satisficing search but can be extended to include other operations, such as managing resource constraints. Despite different objectives, the mathematical formulation of the underlying geometric search problem, provided in (Ablavsky and Snorrason 2000), provides an ideal formal underpinning for our approach, and offers several opportunities for extension of our method.

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. Furthermore, in complex applications path-planning is interleaved with the need to manage constraints and other goals, requiring a more strategic approach to operations.

3 Planning-based Approach to SaT

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 part of many surveillance and search-and-rescue operations. SaT missions are plausible both outdoors and indoors as well as both in adversarial and cooperative contexts.

The objective of a SaT mission is to follow the target to its destination. In general, SaT operations proceed in two phases, which interleave until the target stops or until the observer abandons the mission:

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

In the problems we consider in this paper, the observer is the AR.Drone and the target is a person or any object that moves according to its own intentions and proceeds at a speed compatible with the speed of the drone. We assume that the target needs to reach a specific destination within a confined known area, which might be outdoors or indoors, and chooses an efficient path to do so. In addition, the target does not perform evasive actions by attempting to use features in the environment for concealment. This is a plausible assumption as the target might be cooperating with the drone or simply unaware of its presence. As we do not deal with object recognition, we assume that the target is identified by a specific tag known in advance by the drone, although the drone might fail to observe the target even when it is in view due to its noisy sensors. Finally, we are interested in long-term SaT missions in wide areas, relative to the scale of the drone.

Probabilistic methods currently applied to SaT are not suitable to tackle the characteristics of our scenario. Accurately maintaining a large state space that includes all the possible positions of a target moving unpredictably in a large area is computationally intractable. In addition, our drone cannot afford to employ a myopic approach to search (such as using a one-step lookahead horizon), since the drone operates under tight resource and time constraints due to its limited computational power, noisy sensors, and short battery life. The drone needs to be strategic in deciding what actions to perform, looking ahead at its remaining lifespan and balancing the objective of rediscovering the target with the limitations posed by its technical characteristics within this time frame. It is the need to handle this trade-off that makes automated planning a suitable approach in this context, with planners offering a route to crafting effective strategies for the drone to achieve its mission goal in the face of all relevant constraints.

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. We apply control algorithms to achieve this (see Section 6). When the drone fails to observe the target, it must attempt to rediscover it. For a short period after losing the target, the drone can simply track its predicted location, since the target cannot move fast enough to significantly deviate from this prediction. However, after a longer period, it is necessary to make a more systematic effort to rediscover the target by directing the search into specific places. We formulate the search phase as a *planning problem* consisting of deciding where exactly to search for the target and what manoeuvres to use. The goal is to maximise the likelihood of finding the target while favouring manoeuvres that both minimise the use of the drone's consumable resources and generate robust sensor measurements for stable flight.

In Bernardini et al. (2013), we proposed a deterministic approach to SaT for the case of a single fixed-wing UAV. Here we build on this approach and extend it to handle the specific challenges posed by the use of a quadcopter. We deal with tight resource and time constraints, sensor limitations, complex flight dynamics and thus different search manoeuvres. In addition, we demonstrate the viability of our approach by providing a robust implementation on a real robotic platform.

4 Search as a Planning Problem

Our approach is to search a region by dividing it into subregions corresponding to *search patterns* selected from a library, and then planning the best order in which to visit the selected patterns so as to minimise the expected time to rediscover the target.

In line with SaT and SaR (Search-and-Rescue) international standard guidelines (IMO 2013; NATSAR 2011; CSAR 2000), and in common with (Ablavsky and Snorrason 2000), we employ the following procedure to manage the search phase of a SaT mission:

1. Determine the optimal area where the search effort should be deployed, which is an area where the target is most likely to be found;
2. Divide this area into appropriate sub-areas for assignment to individual *search patterns*, which are sets of manoeuvres for surveying specified regions;
3. Select specific search patterns and their orientations to optimally cover each sub-area;
4. Determine a sequence in which the chosen patterns need to be executed; and
5. Execute the chosen sequence of patterns, switching back to tracking if the target is rediscovered.

Steps 1 and 2 depend on information concerning the specific mission and, in real-world SaT operations, are performed based on a number of biasing factors: the last known position (LKP) of the target, its intentions if known or predictable, its size and characteristics, possible hazards, and results of previous searching, if available. In addition, for outdoor missions, the nature of terrain, the structure of the road network and visibility and weather conditions are considered. For indoor operations, the conditions of the environment and its structure (corridors, stairs, doors, etc.) are taken into account. These factors are used to make predictions on the target's position over time and to construct a probability distribution for the area of operation (see SaR manuals (IMO 2013; NATSAR 2011; CSAR 2000) for additional details and Bernardini et al. (2013) for a specific example). In short, the outcome of Steps 1 and 2 consists of: (i) a confined search area, usually a circular sector centred on the LKP of the target and extending outwards with its symmetry axis aligned with the average bearing of the target over the period it was observed; (ii) a probability distribution of the target's position laid over this sector and constructed considering the above-mentioned factors; and (iii) a number of points within the sector that present the highest probability of rediscovering the target and on which the search patterns will be deployed.

In our application, we assume to have these three pieces of information available for use. We abstract from how they are exactly calculated as this depends on each specific mission and is irrelevant to our technique.

Step 3 is concerned with choosing a particular search pattern to cover each sub-area within the optimal area of operation. We adhere to standard procedures (IMO 2013; NATSAR 2011; CSAR 2000) and use the following search patterns (see Figure 1):

- *Parallel Track Search* (PTS), if the search area is large and level, only the approximate location of the target is known, and uniform coverage is desired;
- *Creeping Line Search* (CLS), if the search area is narrow and long and the probable location of the target is thought to be on either side of the search track;
- *Expanding Square Search* (ESS), if the search area is small, and the position of the target is known within close limits;
- *Sector Search* (SS), used similarly to the ESS, it offers several advantages: concentrated coverage near the centre of the search area, easier to fly than the ESS, and view of the search area from many angles (terrain and lighting problems can be minimised);
- *Contour Search* (CS), used to patrol obstacles, always assumed to be polygonal in our application.

The key difference between our approach and (Ablavsky and Snorrason 2000) is in the use of a task planner in Step 4. A subset of the candidate search patterns generated in Step 3 needs to be selected and sequenced for execution (Step 5). The method in (Ablavsky and Snorrason 2000) tackles this problem using a divide and conquer path optimisation method that finds the most efficient coverage of the search region, taking into account the cost of turn manoeuvres and the reward expected from exploring each shape. Our strategy is to formulate the problem of selecting and sequencing search patterns as a task planning problem and use a reward-based method to influence action selection. We assign to each candidate search pattern a value corresponding to the expectation of finding the target in a search of the area corresponding to the pattern. Then the planner chooses and links search patterns in order to maximise the accumulated expectation of rediscovering the target. The planner takes into account the resource constraints and can plan additional actions to manage them. At the same time, the planner takes care to choose trajectories and manoeuvres that ensure safe navigation. Our approach is therefore non-optimal but applies to a larger class of operations than path-planning alone.

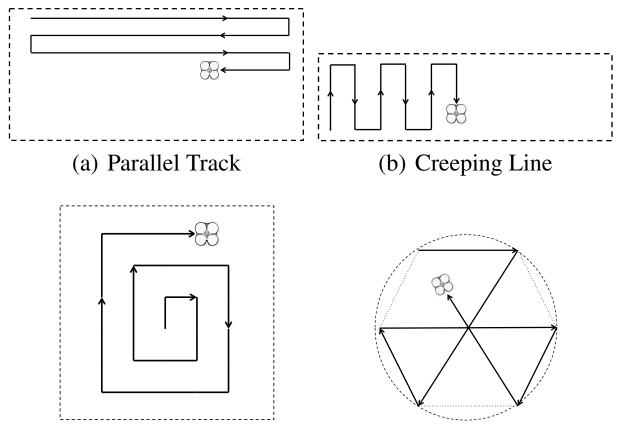

(a) Parallel Track (b) Creeping Line (c) Expanding Square (d) Sector Search

Figure 1: Search patterns used by the drone

4.1 Planning Domain

The domain model for the search problem contains the following actions for the drone: (i) taking-off; (ii) landing; (iii) hovering; (iv) flying between waypoints; (v) performing the five search patterns (PTS, CLS, ESS, SS and CS); and (vi) performing a re-localisation procedure, used when the drone's confusion about its own position rises above a certain threshold.

The take-off and landing actions are straightforward. The only caveat is that the duration of the take-off action and the related battery consumption, pre-computed and fixed in the problem specification, need to take into account the SLAM (Simultaneous Localisation and Mapping) algorithm initialisation.

The search pattern actions all have a similar structure: they 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 total reward (i.e. the accumulated expectation of finding the target) by an amount corresponding to the reward of performing that specific pattern. This amount is established in the problem instance by using a time-dependent function, as discussed in the next section. The search pattern actions are durative and their duration is also fixed in the problem instance and is the 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 covered by the pattern. This is calculated by considering the minimum and maximum reasonable speeds for the target and the distance from the LKP of the target.

Given its cheap and noisy sensors, the quadcopter is often unable to establish its position and state accurately. The SLAM algorithm that we use, PTAM (Parallel Tracking and Mapping) (Klein and Murray 2007), in common with several state-of-the-art monocular SLAM systems, does not perform well in a number of situations (for example, with trees and plants, and in combination with wind) and requires visual features in the environment to be reliable. When the quadcopter cannot count on accurate estimates of its own position and velocity, it becomes unstable and therefore cannot maintain safe flight. To avoid this situation, the planner needs to create plans for the drone that are robust to sensor limitations of this kind. We achieve that by using a numeric function `confusion` that models the fact that, during flight, the drone accumulates uncertainty about its own state. Each manoeuvre performed by the drone increases the level of confusion to an extent specified in the problem instance and, when the confusion goes beyond an established threshold, the drone is forced to perform a `re-localise` action, which involves flying to a location where the SLAM algorithm is known to perform well and re-initialising the state-estimation procedure in such a position. The threshold is set in such a way that, when it is reached, the drone is expected to still be able to reach one of the re-localisation waypoints, although partially confused on its own state. At the same time, the planner plans to keep the confusion level below the threshold by favouring waypoints that maximise the localisation accuracy.

As energy is a critical resource for the drone, we include a model of the drone's battery in the planning domain. Instead of modelling the continuous change of the battery level directly, we use discretised durative actions in combination with numeric step-function updates. The drone's manoeuvres have a condition at start ensuring that the available amount of energy is sufficient to perform the manoeuvre and an instantaneous effect at start modelling their energy consumption. The energy required for the different manoeuvres has been estimated off-line through actual testing with the drone and is fixed in the problem instance (see Section 6).

We use two alternative domains for our drone. In the first domain, we assume that the maximum flight time for the drone is constrained by its battery life. In order to establish a planning horizon for our problem corresponding to the total flight time, we use an *envelope* action `operate` around all the activities of the drone: it must start before the take-off and must finish after landing. In our second domain, we assume that the SaT mission can be subdivided in different chunks. The drone can use the entire battery charge in each chunk and then recharge the battery to perform the next part of the mission. We use a discretised durative action `recharge` to model the recharge of the battery, which only makes new charge available at the conclusion of the action, so that charge gained cannot be exploited until after the recharging is complete. The battery can be charged to full capacity or up to a chosen level.

As an example of how search actions are modelled in PDDL2.2 (Planning Domain Definition Language) (Edelkamp and Hoffmann 2004), Table 1 shows the description of the action `doESS`, which specifies the location and availability of an ESS pattern, its duration, the reward available, the consumption of battery charge and how the execution of the pattern increases the drone's confusion level.

4.2 Planning Problem

The initial state for a planning problem contains all the candidate search patterns among which the planner chooses the ones to execute. The candidate search patterns are the objects of the planning instance. Each candidate search pattern

```
(:durative-action doESS
 :parameters (?from ?to - waypoint ?p - ESS)
 :duration (=?duration (timeFor ?p))
 :condition (and (at start (beginAt ?from ?p))
  (at start (endAt ?to ?p))
  (at start (at ?from))
  (at start (>= (batteryLevel) (batteryRequiredP ?p)))
  (at start (<= (confusion) (maxConfusion)))
  (over all (inOperation))
  (over all (inFlight))
  (at end (active ?p)))
 :effect (and (at start (not (at ?from)))
  (at start (decrease (batteryLevel) (batteryRequiredP ?p)))
  (at start (increase (confusion) (confusionManoeuvre ?p)))
  (at end (at ?to))
  (at end (increase (reward) (rewardOf ?p)))))
```

Table 1: PDDL2.2 specification of the action `doESS`

is assigned an appropriate *reward*, which is a value corresponding to the expectation of finding the target in a search of the area covered by the pattern. The reward depends on both the probability that the target enters the area being searched and the probability that the drone sees the target while passing over it. Both these probabilities are affected by several factors, such as the type of search pattern, the camera used, the direction in which the target has been travelling and the characteristics of the area traversed by the target (e.g. terrain types and road network for outdoor missions). As we deal with a moving target, the reward is a *time-dependent* function. No reward is assigned until the target has plausibly arrived to the area covered by the pattern and once the target is deemed likely to have left the area. Between these extremes, the reward is modelled as a step function that approximates a lifted Gaussian distribution (Figure 2). It increases when the search pattern becomes active, it increases further when the target is considered most likely to be in the pattern and then it decreases until the end of the useful life of the pattern. The reward peaks at the point where the target would be in the centre of the search pattern if proceeding at average speed. We use a unimodal distribution because we assume that the target moves towards a destination, and so it does not revisit locations that it has already visited, and uses the most efficient path to reach its destination. The variance is generated by uncertainty about the precise path and precise speed of the target. 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. In addition, we also use *timed initial literals*, which are asserted and retracted at the appropriate times, in order to force patterns to be active only during the period when the distribution is positive.

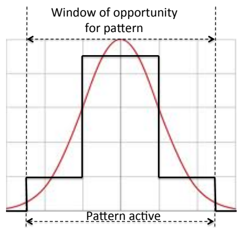


Figure 2: Time-dependent reward function

In the initial state, we also assign a confusion level to each search pattern, which measures the localisation accuracy of the drone in the area covered by the pattern and depends on the physical characteristics of the area.

Usually, the planning problem has just one goal, specifying that the drone should be on the ground at the end of the operations. However, if relevant for a certain mission, one can specify a number of waypoints that the drone needs to visit before landing. We use a plan metric that measures the value of the plan in terms of the accumulated expectation of finding the target and the level of confusion. Clearly, our objective is to maximise the reward and minimise the confusion level, so we use the following metric: `(:metric maximize (- (reward) (* K (confusion)))))`, where the parameter `K` needs to be established case by case based on the desired trade-off between the reward and the confusion level.

4.3 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, since the drone has a very limited battery life, we are in a time-critical situation.

The planner will typically find a first solution very easily, but it will then spend the additional time improving on this by adding further manoeuvres to the plan, or trying different collections of manoeuvres. OPTIC identifies that bigger values of `(reward)`, smaller values of `(confusion)` and bigger level of `(batterylevel)` are preferable. The search uses a weighted-$A^\star$ scheme with steadily changing weights in a tiered fashion. The plans produced 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. Table 2 shows an example of a plan generated by OPTIC for our drone.

Execution time	Action	Duration
0.000:	(take-off origin)	[40.000]
40.001:	(fly origin esp1s)	[71.000]
111.002:	(doESS esp1s esp1e square1)	[45.000]
156.003:	(fly esp1e cp1se)	[106.000]
262.004:	(doCS cp1se cp1se contour1)	[25.000]
287.005:	(doCS cp1se cp1se contour1)	[25.000]
312.006:	(land cp1se)	[2.000]
314.007:	(recharge cp1se)	[1200.000]
1514.008:	(take-off cp1se)	[40.000]
1554.009:	(fly cp1se esp2s)	[104.000]
1658.010:	(doESS esp2s esp2e square2)	[75.000]
1733.011:	(fly esp2e cp2se)	[110.000]
1843.012:	(doCS cp2se cp2se contour2)	[35.000]
1878.013:	(fly cp2se lp1)	[67.000]
1945.014:	(relocalise lp1)	[45.000]
1990.015:	(fly lp1 sp1s)	[73.000]
2063.016:	(doSS sp1s sp1e sector1)	[95.000]
2158.017:	(land sp1e)	[2.000]

Table 2: Plan generated by OPTIC for our drone

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: spend more time generating a new plan, or abandon the search. In our current implementation, we always make the drone abandon the search and land.

5 Hardware Platform: Parrot AR.Drone

We use the AR.Drone quadcopter as our prototyping hardware platform. 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 and, since then, it has been increasingly popular in academia and research organisations as an affordable test platform for MAV demonstrations (Bills, Chen, and Saxena 2011; Engel, Sturm, and Cremers 2012b; Graether and Mueller 2012). The AR.Drone has a number of advantages: it is sold at a very low cost, is robust to crashes, can be safely used in close proximity 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. The drone's technical specifications are reported in Table 3.

Weight	- 380 g with indoor hull - 420 g with outdoor hull
Dimensions	52×52 centimetres
Running Speed	5 m/s (18 km/h)
Running Time	12 minutes of continuous flight
Battery	- Lithium polymer (3 cells, 11.1V, 1000 mAh) - Charging time: 90 minutes
Embedded Computer System	- 1GHz 32-bit ARM Cortex A8 processor - 1Gbit DDR2 RAM at 200MHz - Wi-Fi 802.11b/g/n - Linux 2.6.32 OS
Inertial Guidance System	- 3-axis accelerometer - 2-axis gyroscope - 1-axis yaw precision gyroscope
Ultrasound Altimeter	- Emission frequency: 40 KHz - Range: 6 metres - Vertical stabilisation
Front-Facing Camera	- 92° wide angle diagonal lens (CMOS sensor) - Camera resolution: 1280×720 pixels (720p) - Video frequency: 30 fps
Bottom-Facing Camera	- 64° diagonal lens (CMOS sensor) - Camera resolution: 320×240 pixels (QVGA) - Video frequency: 60 fps - Ground speed measurement and stabilisation

Table 3: Technical specifications of the AR.Drone 2.0

The onboard software provides three communication channels with the drone:

- *Command* channel: sending commands to the drone (e.g. take-off, land, calibrate sensors, etc.) at 30 Hz.
- *Navdata* channel: providing information about the drone status (flying, calibrating sensors, etc.) and pre-processed sensor data (current yaw, pitch, roll, altitude, battery state and 3D speed estimates) at 30 Hz.
- *Stream* channel: providing images from the frontal and/or bottom cameras.

6 Implementation

To allow the drone to carry out a SaT mission autonomously, we combine the abstract deliberative skills illustrated in Section 4 with low-level control and vision capabilities. We implemented different techniques for the two phases of a SaT mission. We first give an overview of them and then provide additional details on their implementation.

- Tracking phase:
  - Tag recognition: Since we assume that our target is identified by a specific tag, 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 identifying the target, it needs to follow it reactively. We achieve this by implementing a Proportional-Integral-Derivative (PID) controller that works based on the navigation data provided by the drone's driver.
- Search phase: The AR.Drone provides built-in low-level control for robust flight, such as stabilisation and attitude maintenance, so we focus on high-level control only. Since we currently ignore obstacle avoidance, our implementation provides capabilities for localisation and mapping, navigation and compensation for drift. The navigation system that we use is composed of: a monocular SLAM implementation for visual tracking, an Extended Kalman Filter (EKF) for data fusion and prediction, and a PID control for pose stabilisation and navigation.

We implemented our application within the Robot Operating System (ROS) framework (Quigley et al. 2009). We exploited existing packages for implementing our SaT application. In particular, we built on the following ROS packages:

- ARDRONE_AUTONOMY[1]: this is a ROS driver for the Parrot AR.Drone based on the official AR.Drone SDK version 2.0. The driver's executable node, ARDRONE_DRIVER, offers two main features:
  - it converts all raw sensor readings, debug values and status reports sent from the drone into standard ROS messages, and
  - it allows to send control commands to the drone for taking off, landing, hovering and specifying the desired linear and angular velocities.
- AR_RECOG[2]: this is a ROS vision package that allows the drone to recognise specific tags as well as to locate and transform them in the image-space. It is based on the ARToolKit[3], which is a robust software library for building augmented reality applications.

[1]http://wiki.ros.org/ardrone_autonomy
[2]http://wiki.ros.org/ar_recog
[3]http://www.hitl.washington.edu/artoolkit/

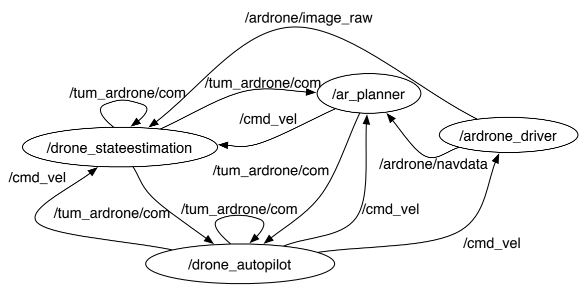


Figure 3: ROS nodes and topics for the search phase

- TUM_ARDRONE[4]: this ROS package implements autonomous navigation and figure flying in previously unknown and GPS-denied environments (Engel, Sturm, and Cremers 2012a; 2012b). The package contains two main nodes:
- DRONE_STATE-ESTIMATION: This node provides the drone with SLAM capabilities by implementing an algorithm based on Parallel Tracking and Mapping (PTAM) (Klein and Murray 2007). In addition, it 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.

We also implemented two additional ROS nodes:

- AR_TAG_FOLLOWING: This node implements a PID controller that, based on the messages received from the AR_RECOG package, allows the drone to follow the detected tag.
- AR_PLANNER: This node wraps the OPTIC planner and allows us to integrate it with the rest of the system.

AR_PLANNER is invoked by the node AR_TAG_FOLLOWING when the target has been lost. Based on real-time data from the *navdata* channel and information concerning the probability distribution of the target's position in the search area, the AR_PLANNER: (i) builds a planning problem; (ii) calls the OPTIC planner with a time limit of 10 seconds; (iii) upon receiving the last plan generated within the time limit, translates the actions of the plan into corresponding commands specified by using the scripting language provided by the ROS node DRONE_AUTOPILOT; and (iv) sends the translated plan to the node DRONE_AUTOPILOT, which then imparts the commands directly to the ARDRONE_DRIVER node.

Since ARDRONE_AUTONOMY is the drone's driver, we use it both for the tracking and the search phases. AR_TAG_FOLLOWING and AR_RECOG were employed for tracking only, whereas the DRONE_STATE-ESTIMATION and DRONE_AUTOPILOT for search only. The AR_PLANNER node acts as the interface between the tracking and the search phases. Figure 3 shows the ROS nodes and topics active during search.

[4]http://wiki.ros.org/tum_ardrone

Although using ROS and exploiting existing packages for the AR.Drone facilitated the implementation of our application, working with a real quadcopter remains a time-consuming and challenging task. 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.

7 Flight Tests

We conducted a series of real-world experiments to assess whether the planner is capable of generating effective plans for the drone and whether the drone is able to fly them accurately to completion. Flight tests were performed indoors in a room of dimensions 20.5m (l) $\times$ 8.37m (w) $\times$ 5.95m (h). The experiments presented here pertain to the search phase of a SaT mission only. They were carried out by using the ROS architecture shown in Figure 3 and the AR.Drone 2.0.

We employed the following procedure to run our tests: we manually generated several planning problems and fed each problem, together with the drone's domain model, into the ROS architecture shown in Figure 3.

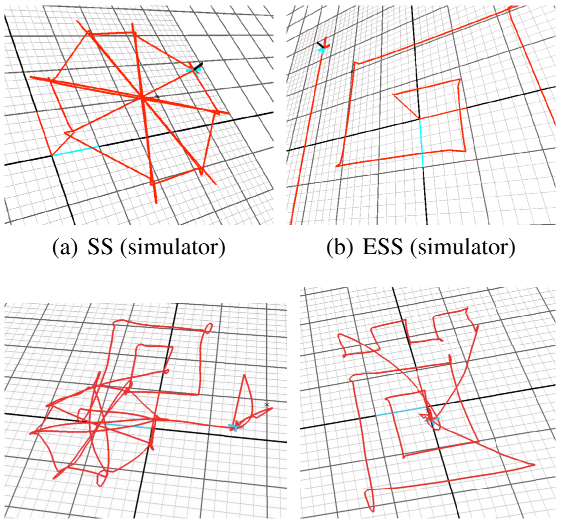
Manually writing the planning problems was a time-consuming task. As described in Section 4.2, the initial state describes the set of candidate search patterns from amongst which the planner chooses those to execute. In particular, it assigns to each pattern the following information: the time and battery needed for flying the pattern, the drone's degree of confusion after executing the pattern, a time window in which the pattern is active and a reward step function within this window. To write realistic initial states, we manually created several search patterns for the drone with suitable dimensions for our room and ran extensive flight tests of such patterns to gather reliable information about their duration and the battery usage. In particular, we created several versions of each pattern with different dimensions, ran each pattern ten times and annotated the average time and energy consumed by the pattern. The results obtained for one version of each patterns are reported in Table 4. The take-off (1m) and PTAM map initialisation take approximately 40 seconds and 10% of battery, while landing (1m) takes five seconds and 2% of battery.

	PTS-1	CLS-1	ESS-1	SS-1
Area (m*m)	4*3	1*4	4*4	3*3
Perimeter (m)	19	9	20	29
Time (sec)	50	40	45	95
Battery (%)	10	15	5	10

Table 4: Characteristics of patterns PTS-1, CLS-1, ESS-1 and SS-1.

On average, OPTIC produces around 8 plans per problem instance in its 10 second window.

When we made the drone execute the plans produced by the planner, we observed that the drone is able to fly all the figures very accurately. Figure 4 shows, in the top row, the execution of a SS pattern and an ESS pattern by the drone's simulator and, in the bottom row, the execution of two fragments of a plan involving combination of patterns by the real drone. By comparing these pictures, it appears that the drone can execute the planned manoeuvres very precisely.


(a) SS (simulator) (b) ESS (simulator)

(c) SS + ESS (drone) (d) ESS + CLS (drone)

Figure 4: A comparison between the figures in the above and bottom rows shows that our drone is able to execute the plans generated by the planner very accurately

In future we would like to run experiments to compare our approach with other methods. While Ablavsky *et al.* have developed a mathematical model of a comparable approach, there appears to be no implemented system available. In terms of path-planning, both approaches are very similar, but Ablavsky *et al.* could not manage resource constraints or non-search goals. It would be worth comparing our search pattern-based approach with the probabilistic methods of (Richardson and Stone 1971) and (Furukawa et al. 2012), on experiments to find a stationary or drifting target, to explore scaling. Our future work will explore this further.

8 Conclusions and Future Work

In this paper, we describe our work on the use of automated planning for the high level operation of a low-cost quadcopter engaged in SaT missions. We formulate the search problem as a planning problem in which standard search patterns must be selected and sequenced to maximise the probability of rediscovering the target. We use an off-the-shelf planner, OPTIC, to solve the planning problem and generate plans for the drone.

The use of planning technology in the search phase of SaT missions satisfies the drone's requirement of having to constantly compromise between two orthogonal metrics: one favouring locations where the probability of rediscovering the target is high and the other favouring locations that generate robust sensor measurements for safe flight. If the drone needs to complete its mission within the limit of its battery life (no recharge action available), yet another metric comes into play, favouring manoeuvres that minimise battery use. Only with a carefully crafted strategy can the drone juggle these constraints and achieve satisfactory performance.

Although we focus here on the Parrot AR.Drone2.0 SaT missions, our approach can easily be generalised to other MAVs and robotic vehicles as well as to 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 while under tight deadlines and resource constraints.

References

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.

Ablavsky, V., and Snorrason, M. 2000. Optimal search for a moving target: a Geometric Approach. In *Proceedings of the AIAA GNC Conference*.

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.

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.

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)*.

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)*.

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)*.

Chung, C. F., and Furukawa, T. 2006. Coordinated Search-and-Capture Using Particle Filters. In *Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision*.

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*.

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.

CSAR. 2000. *Canadian National Search and Rescue Manual*. Department of National Defence.

Edelkamp, S., and Hoffmann, J. 2004. PDDL2.2: The language for the classical part of the 4th international planning competition. In *Proceedings of the 4th International Planning Competition (IPC-04)*.

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*.

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)*.

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.

Furukawa, T.; Bourgault, F.; Lavis, B.; and Durrant-Whyte, H. F. 2006. Recursive Bayesian Search-and-tracking using Coordinated UAVs for Lost Targets. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA 2006)*, 2521–2526.

Furukawa, T.; Mak, L. C.; Durrant-Whyte, H. F.; and Madhavan, R. 2012. Autonomous Bayesian Search and Tracking, and its Experimental Validation. *Advanced Robotics* 26(5-6):461–485.

Furukawa, T.; Durrant-Whyte, H. F.; and Lavis, B. 2007. The element-based method – Theory and its application to Bayesian search and tracking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, 2807–2812.

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.

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.

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.

IMO. 2013. *International Aeronautical and Maritime Search and Rescue Manual (IAMSAR)*. United States Federal Agencies.

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)*.

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.

NATSAR. 2011. *Australian National Search and Rescue Manual*. Australian National Search and Rescue Council.

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)*.

Richardson, H. R., and Stone, L. D. 1971. Operations analysis during the underwater search for Scorpion. *Naval Research Logistics* 18:141–157.

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)*.

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.