Swarm Intelligence manuscript No.
(will be inserted by the editor)

Energy-Efficient Indoor Search by Swarms of Simulated Flying Robots Without Global Information

Timothy Stirling · Steffen Wischmann · Dario Floreano

Received: date / Accepted: date

Abstract Swarms of flying robots are a promising alternative to ground-based robots for search in indoor environments with advantages such as increased speed and the ability to fly above obstacles. However, there are numerous problems that must be surmounted including limitations in available sensory and on-board processing capabilities, and low flight endurance. This paper introduces a novel strategy to coordinate a swarm of flying robots for indoor exploration that significantly increases energy efficiency. The presented algorithm is fully distributed and scalable. It relies solely on local sensing and low-bandwidth communication, and does not require absolute positioning, localisation, or explicit world-models. It assumes that flying robots can temporarily attach to the ceiling, or land on the ground for efficient surveillance over extended periods of time. To further reduce energy consumption, the swarm is incrementally deployed by launching one robot at a time. Extensive simulation experiments demonstrate that increasing the time between consecutive robot launches significantly lowers energy consumption by reducing total swarm flight time, while also decreasing collision probability. As a trade-off, however, the search time increases with increased inter-launch periods. These effects are stronger in more complex environments. The proposed localisation-free strategy provides an energy efficient search behaviour adaptable to different environments or timing constraints.

Keywords Flying robots · Swarm search · Localisation-free search · Energy-efficiency · Mobile robot sensor network deployment

1 Introduction

Swarms of flying robots are promising for search in indoor environments where the ability to move above debris or obstacles such as furniture is advantageous (Nardi et al 2006, Oh et al 2005, Rudol et al 2008, Hoffmann et al 2004, Melhuish and Welsby 2002). Such a swarm can establish a robot sensor and communication network that can impart navigation aid to other robots or humans, e.g. guiding paramedics to injured victims, (O'Hara and Balch 2004, Li et al 2003, Batalin et al 2004, Corke et al 2005, Nouyan et al 2008; 2009).

T. Stirling · S. Wischmann · D. Floreano
Laboratory of Intelligent Systems (LIS), Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
E-mail: tim.stirling@epfl.ch

There are many challenges that must be surmounted in order to achieve swarm search with flying robots indoors. First, robotic search often relies either on absolute positioning (Burgard et al 2005, York and Pack 2008) or localisation (Meyer and Filliat 2003, Stipes et al 2006, Durrant-Whyte and Bailey 2006). Absolute positioning is often achieved with GPS, but this is usually unfeasible indoors due to attenuated signals (Zeimpekis et al 2003, Siegwart and Nourbakhsh 2004). Robot localisation commonly requires environment maps and odometry sensing (Filliat and Meyer 2003). Environment maps may be unknown *a priori* and the online creation of maps requires powerful processing that may not be available on small flying robots (Grzonka et al 2009). Also, such approaches do not scale appropriately with large swarms (Burgard et al 2005). Unlike ground-based robots with wheel encoders, odometry sensing is challenging with flying robots (Iida 2003).

An additional challenge is that flying robots usually suffer from drift due to aerodynamic imbalance and turbulence (Hoffmann et al 2007). This is commonly compensated with GPS outdoors (e.g., York and Pack 2008, Hoffmann et al 2007) or pre-installed 3-D tracking sensors indoors (e.g., Valenti et al 2007). To date, on-board approaches normally involve visual motion-tracking of environmental features, which requires powerful processors or specialised hardware (Fowers et al 2007). Further challenges are limited flight autonomy (usually only 10-15 minutes, Roberts et al 2008), and an increased risk of collisions due to multiple robots flying in proximity (Sharma and Ghose 2007).

In order to overcome the challenge of localisation and positioning without global information, we employ concepts from mobile sensor networks. Sensor networks perform completely distributed processing of local information through a wireless communication network (Howard et al 2002a), thus allowing simple navigation and efficient exploration without requiring absolute positioning or localisation (e.g., O'Hara and Balch 2004, Batalin and Sukhatme 2002, Batalin et al 2004). Most sensor networks are predeployed into the operating environment (e.g., Li et al 2003, Corke et al 2005), which is unfeasible during disaster situations. Alternatively, mobile robot sensor networks can autonomously deploy into unknown environments and spread out to increase the area covered by all the robot sensors (referred to as sensor coverage), (e.g., Batalin and Sukhatme 2002). Most work in mobile sensor network deployment (e.g., Howard et al 2002a, Zou and Chakrabarty 2003, Batalin and Sukhatme 2002) aims at achieving only a static arrangement of robots for sensor coverage, termed blanket coverage (Gage 1992). Here, we demonstrate how to dynamically redeploy robots and increase the coverage area, referred to as sweep coverage (Gage 1992, Liu et al 2005). Various approaches to deploying robot sensor networks exist (discussed in Sect. 2), but are not applicable to swarms of flying robots.

To tackle the challenge of increasing the energy autonomy, we assume that the flying robots can land or attach to some structure of the environment and power down rotors while maintaining communication with neighbouring robots. The work in this paper is based on a hovering robot capable of attaching to ceilings (Roberts et al 2008). Besides minimising energy consumption, this allows the formation of a robot sensor network for efficient long term monitoring. Furthermore, we present an incremental deployment scheme which controls the launching of robots and gradually expands the robot sensor network. This gradual deployment exploits information acquired from the sensor network to guide robots more efficiently and, therefore, reduces unnecessary flight time. Additionally, this reduces the number of concurrent flying robots, thus reducing collision risk. We show how the time between consecutive robot launches can be tuned to balance coverage time and energy efficiency.

Lastly, to achieve autonomous flight and control platform drift we propose that flying robots use spatial information from robots attached to the ceiling using on-board relative-positioning sensors (Roberts et al 2009).

This paper is organised as follows. First, we introduce related work and its limitations. Then we present our robotic and environmental assumptions and an overview of our strategy. We then discuss algorithmic details and experimental set-up. We then present our results from extensive simulation analysis. Finally, we discuss limitations and future work.

2 Related Work

Various authors have proposed search and exploration strategies without absolute positioning or localisation via deployment of a network of sensor nodes or radio beacons. For example, Li et al (2003) use distributed algorithms for guiding robots across a sensor network and Corke et al (2005) use a pre-deployed sensor network to guide flying robots in unknown environments. Both of these approaches require the prior deployment of the sensor network, which would be unlikely in distaster situations or unknown environments. Alternatively, Batalin and Sukhatme (2004) use the robots to deploy a network of radio beacons at runtime to aid efficient exploration and search. Similarly, Pezeshkian et al (2007) deploy a network of static radio beacons forming a long range communication network that could aid robot exploration. However, both these approaches require the inefficient transportation of separate beacon hardware, which is not feasible for flying robots with small payload. Ziparo et al (2007) and Mamei and Zambonelli (2005) use micro-sized Radio-Frequency Identification chips (RFIDs) to act as markers. These have the benefits of low cost, very small size and passive power. Unfortunately RFIDs can only be read from distances too short for flying robots. Our proposed strategy does not rely upon *a priori* deployment of sensors or beacons, or inefficient transportation and retrieval of dedicated hardware, and avoids using RFIDs or other passive short-range devices. Instead, our strategy uses the robots themselves to form beacons within a mobile robotic sensor network to facilitate navigation. This is similar to the approach of Nouyan et al (2008; 2009), although in their paper they use terrestrial robots.

As an alternative to the costly transportation of separate beacons, some authors (Payton et al 2001, Howard et al 2002a, Reif and Wang 1999, Baxter et al 2007, Batalin and Sukhatme 2002) propose to use the robots themselves as sensor nodes that can autonomously deploy into unknown environments. The most common approaches to mobile robotic sensor network deployment are based on virtual forces between robots, termed Social Potential Fields. Attraction and repulsion forces between robots are defined to create a self-organising group behaviour (Reif and Wang 1999). The concept of Social Potential Fields has been applied to disperse robot swarms for sensor coverage tasks while maintaining properties such as line-of-sight connectivity and maximum inter-robot distances (Zavlanos and Pappas 2007, Reif and Wang 1999, Howard et al 2002b, Baxter et al 2007, Zou and Chakrabarty 2003, Poduri and Sukhatme 2004, Batalin and Sukhatme 2002). These basic dispersion strategies have been adapted or extended in several ways to increase the total sensor coverage area or to reduce deployment time. For example, Zou and Chakrabarty (2003) combined potential fields with disc-packing algorithms used for optimal sensor placement to improve coverage from an initially random dispersion, but relied on a centralised and computationally expensive algorithm. Social Potential Fields are promising but have several undesirable limitations. Firstly, they require complex tuning of the force laws. For each field there are usually four parameters (the coefficients and power exponents for attraction and repulsion) and multiple fields are required for task oriented behaviour. Determining the dynamics of such a system is polynomial-space hard, so determining the set of required force law parameters for a desired group behaviour is likely to be computationally unfeasible (Reif and Wang 1999). This means the equilibrium steady-state cannot always be predicted, or the

system may converge to local-minima (Reif and Wang 1999). In comparison, our approach does not require any complex parameter optimisation for functional operation. Additionally, with Social Potential Fields, all robots remain in constant motion and slowly converge to a steady-state. This is not the case with our approach, where most robots are in an efficient static state passively attached to ceilings (see Sect. 3) rather than undergoing continual costly motion. Finally, Social Potential Fields usually only achieve a static formation with coverage area limited to the number of robots. Conversely, our proposed strategy dynamically redeploys the robotic sensor network once an area is searched, thereby increasing the coverage area (e.g., Liu et al 2005).

In another approach, Payton et al (2001; 2004) utilise digitally transmitted "virtual pheromone" messages to create gradients across the robot swarm, facilitating movement coordination away from obstacles and into empty spaces. Virtual pheromones are a Swarm Intelligence (Swa 1999, Dorigo and Birattari 2007) approach to multi-robotic coordination inspired by ant foraging. Payton et al (2001) achieve local dispersion with attraction and repulsion forces, but use pheromones to guide exploration over longer distances. The advantage of virtual pheromones is the potential to adapt to dynamic environments since the pheromone gradients naturally adjust to disturbances. However, the approach has been criticised for slow deployment (Howard et al 2002a). Pheromone robotics is promising but so far no quantitative results have been published. Additionally, all robots undergo continuous motion, not desireable with the limited autonomy of flying robots.

A different swarm search approach is presented by Nouyan et al (2008; 2009), which deploys chains of visually connected robots into unknown environments. The robots form a path connecting a 'nest' to a 'prey' object, facilitating simple navigation by embedding robots in the environment, which is similar to our proposed approach. A limitation of Nouyan et al's approach is that it requires tuning several parameters which affect the probabilistic behaviour of the system. These parameters require extensive environment- and robot group size-dependent optimisation to attain best results. Furthermore, their approach uses a stochastic search behaviour that may duplicate the searched area. The probabilistic generation of chains can also lead to unnecessary locomotion as chains randomly grow or shrink. Because of the challenge of energy autonomy with aerial robots, we propose an efficient systematic search strategy which also does not require parameter optimisation.

An efficient systematic incremental deployment algorithm is presented by Howard et al (2002a). Robot sensor nodes are deployed strictly one at a time, making use of previously acquired information from the sensor network to efficiently guide the subsequent robot to an optimal location. This approach requires all sensor nodes to produce a map of free-space which is shared, combined, and later processed into a map of reachable locations from which an optimum can be selected. Howard's incremental deployment scheme allows most robots to remain static with only a single robot moving at a time into desired positions at optimal speeds. The main disadvantages of this approach are that it is computationally expensive, the deployment is slow and the nodes require accurate global positioning. Conversely, our strategy has low computational complexity that is constant with respect to the environment size and number of robots. It is entirely decentralised, does not require the exchange of large amounts of data such as maps, and only relative-positioning sensors are required rather than GPS. Furthermore, restricting deployment to a single robot at a time and waiting for it to arrive at the desired location before deploying the next robot leads to slow group deployment. We propose a compromise solution by continuously deploying a single robot at set time intervals (inter-launch periods), which we refer to as temporal incremental deployment. This temporal incremental deployment can be adjusted to find the optimal trade-off between coverage time and energy efficiency for specific environments and task requirements.

Finally, most previous research involving flying robots in search tasks usually use GPS in obstacle-free outdoor environments (e.g., York and Pack 2008, Hoffmann et al 2007, Ahmadzadeh et al 2006, Flint et al 2002). Indoor flying robots usually use pre-installed 3-D tracking cameras, such as the Vicon[1] system (e.g., Valenti et al 2007). Our approach utilises an on-board relative-positioning sensor (Roberts et al 2009) to compensate for the lack of global absolute positioning. Much research in controlling multiple flying robots often depends on either centralised planners, global communication, or *a priori* information of the environment or target distributions (e.g., Ahmadzadeh et al 2006, Flint et al 2002, Bryson and Sukkarieh 2007). Our algorithm requires only local communication, decentralised processing and does not require *a priori* environment or target information

3 Methods

In this section we first present robotic and environment assumptions. We then present an overview of the algorithm, details of the search behaviour and finally the experiment set-up.

3.1 Assumptions

Robotic System: Realistic assumptions are made with respect to the limited sensing capabilities, communication range and bandwidth, processing power and memory based on the flying robots and sensors we are developing in our lab (Roberts et al 2007; 2008; 2009). We assume the robots are equipped with short range distance sensors, e.g. infrared sensors, in at least the cardinal directions (e.g., Roberts et al 2007). These sensors have a maximum sensing range S_r; for our sensors $S_r = 3.5$ m. The robots require relative-positioning sensors providing the range and bearing between proximal neighbours as well as local low bandwidth line-of-sight communication (e.g., Roberts et al 2009, Payton et al 2001, Pugh et al 2009, Melhuish and Welsby 2002, McLurkin and Smith 2007). Communication bandwidth requirements are minimal, with communication packets typically only 3 bytes giving a maximum robot bandwidth of 240 bit/s with controllers running at 10 Hz. Communication is assumed to be reliable within a certain range C_r and nonexistent beyond, for our system $C_r = 4.0$ m. All actuation and sensing is subject to noise, which is modelled from characterisation experiments (see Appendix B). Our flying robots have been developed to passively attach to ceilings and power down their rotors, allowing a bird's-eye view for extended periods (Roberts et al 2008). Alternatively, flying robots could merely land to conserve power. We assume that all the robots can maintain a similar heading using compass and/or gyroscopic data, which is feasible with hovering robots that do not require yaw control for directed flight. However, to date this has not been shown in hardware. Alternatively, robots could simply utilise relative headings, calculated by robots sharing bearing information as done by Pugh et al (2009).

Operating Environment: We focus on built environments characterised by straight branching corridors connecting rooms, an assumption true of most indoor environments. We define corridors to consist of two approximately parallel walls separated by less than the maximum distance sensor range S_r. Hence, robots in such a closed space can perceive both opposing walls. Conversely, all other areas such as rooms or wide hallways where robots cannot perceive opposing walls are termed open space. Importantly, corridors contain junctions where

[1] `www.vicon.com`

the corridor branches into separate corridors that lead to environment subareas. Environment cycles (loops) in corridors are handled automatically by our algorithm. As a simplification we assume environments that are aligned in the cardinal compass directions, but the algorithm is readily generalisable to more complex environments, as discussed in Sect. 5. Fig. 5 shows example environments.

3.2 High-level Description of the Swarm Search Strategy

All robots operate in one of two control states: "*beacons*" and "*explorers*". Beacons are static robots passively attached to the ceiling to conserve energy and form the robotic sensor network. Explorers are actively flying, deploying into the environment being guided by the beacons. Beacons sense their local environment and communicate with neighbouring beacons to derive a "*desired direction*" signal to guide nearby explorers. Initially there is only a pre-deployed base beacon. Explorers start clustered on the ground below the base beacon and deploy consecutively, ascending to a designated altitude before following the desired direction signal of the nearest beacon, and then flying from beacon to beacon across the network. Beacons on the edge of the network next to unexplored space indicate adjacent locations where a new beacon is required to expand the network. Explorers that arrive at these locations can ascend to the ceiling, attach, and become beacons themselves. Thereby, explorers disperse out and expand the network. In this manner, explorers can focus on the complex task of flying while the beacons with their stable sensing can calculate the navigation required for efficient search. Beacons can revert back to explorers once a subarea has been searched and can redeploy to unexplored areas, increasing the search area. Explorers utilise their relative-positioning sensor with reference to the static beacons to stabilise flight. The beacon network can compensate for the robots' limited sensing and communication capabilities, overcoming the lack of GPS or localisation ability and extending the communication range of the swarm. Since explorers merely follow the navigation signal of beacons and beacons only process their local environment, computational requirements are low. In the following we give a detailed description of the behaviours of beacons and explorers.

3.3 Beacon Behaviour

Here we describe properties of the beacon network topology, how communication routing information is used for simple navigation, and summarise the search algorithm.

Beacon Network Topology: It is important that the robots form a network of beacons with a topology that fulfils certain requirements. We wish to maximise the beacon sensor coverage and reduce the number of robots required to sense an area. Importantly, beacons must leave no gaps in their sensor coverage and maintain communication with neighbours given the limited reliable sensing and communication ranges ($S_r = 3.5$ m and $C_r = 4.0$ m). Therefore, we aim to separate beacons by an interspace distance $I_r \simeq 3.0$ m, less than the sensing and communication ranges ($I_r < S_r < C_r$), and form a regular square lattice with each beacon maintaining communication with up to 4 neighbours in the cardinal directions (see Fig. 1). This regular topology requires fewer beacons to cover an area compared to a random distribution and maintains a high degree of uniformity in inter-node distances. This uniformity reduces interference between beacons allowing a reduction in communication power and, therefore, ensures a longer system lifetime (Heo and Varshney 2005). Beacons that detect

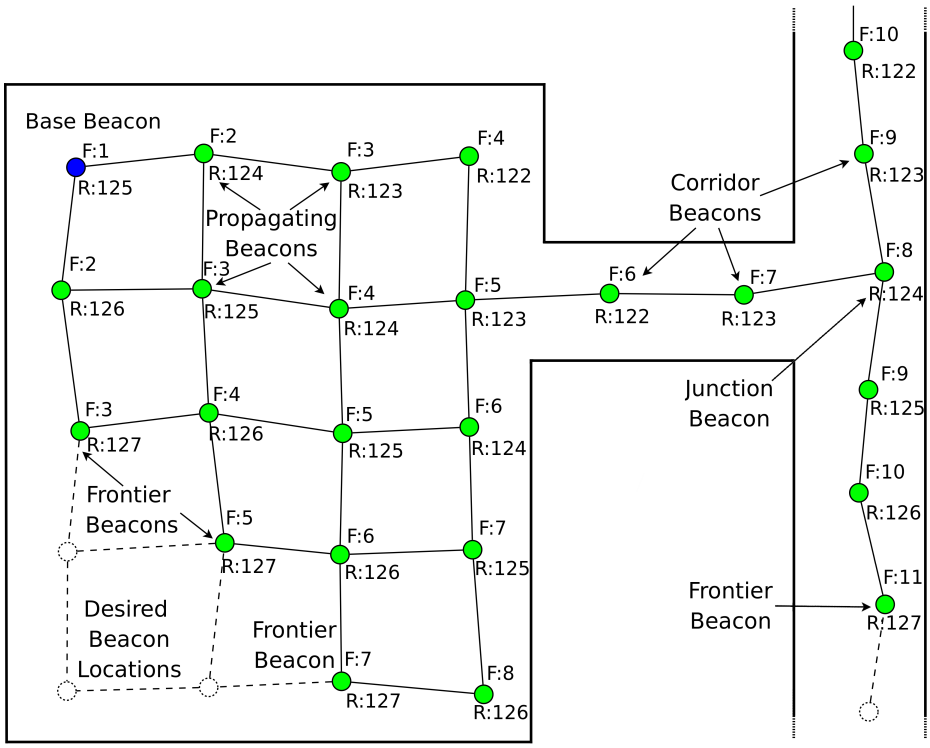

Fig. 1 Beacons are spaced to minimise sensor overlap and maintain communication with neighbours given the limited sensing and communication ranges. Frontier beacons are shown on the network edge next to empty space, while propagating beacons are located in the network interior. Forwards (F) and reverse (R) hop-counts are shown. Following the gradient of increasing reverse hop-counts leads to an empty space where a beacon is required, while following decreasing forwards hop-counts leads to the base

two opposing walls with their distance sensors closer than 3.0 m form "corridor beacons", otherwise they are "open space beacons" (as opposed to the closed space that exists in corridors), see Fig. 1. When beacons are located along corridors their sensory range S_r ensures that only a single chain of beacons is required to span the corridor. Beacons on the edge of the network are called "frontier beacons", while beacons surrounded by neighbours or walls are called "propagating beacons" (see Fig. 1).

Hop-count Gradients: An important mechanism for our robot deployment strategy depends on the hop-counts of messages propagated through the robot beacon network. Hop-counts are integers that indicate the number of nodes through which a communication packet has been routed in a network (Perkins and Royer 1999). The hop-count of a communication packet indicates the distance from the robot node that sent the message, if the robots are approximately uniformly spaced as with our beacon network. This is useful to find the shortest path across the network for navigation and exploration purposes (Li et al 2003, Payton et al 2004) and is equivalent to a distributed Dijkstra's algorithm (Dijkstra 1959), which provides the shortest path between two nodes in an undirected graph. The hop-counts create a gradient across the network which can be followed forwards or backwards. We create two different hop-count gradients with different properties. The first, termed "forwards" gradient, is cre-

ated by messages continually emitted from the base beacon, starting with a hop-count of 1. These messages are received by neighbouring beacons and the hop-count is incremented and propagated outwards away from the base towards the network edge (see Fig. 1). This gradient provides a direction leading away from the base if followed in increasing order, or towards the base if followed in decreasing order. The second hop gradient we refer to as a "reverse" gradient since this emanates from frontier beacons on the network edge with surrounding empty space and decrement, arbitrarily from 127, backwards (see Fig. 1). Conversely to the forwards gradient, the reverse gradient always provides the shortest path to the nearest empty space in unexplored areas of the environment, but not necessarily to the base. Further details of the calculation of hop-counts are given in Appendix A.

Search Algorithm: Beacons are tasked with signalling a desired direction such that nearby explorers are guided to empty space in unexplored areas of the environment (discussed later in Sect. 3.4). The desired direction signal is one of $\langle North, East, South, West \rangle$ and points either towards a neighbouring beacon, or an empty space where a new beacon is required. Beacons use their relative-positioning and distance sensors for local sensing combined with simple information such as hop-counts of neighbouring beacons from the communication network to select the desired direction. In this way, the local sensing of the beacon is combined with information distributed through the network.

We propose an efficient systematic search that exhaustively searches a subarea of the environment before searching other areas by applying depth-first search (DFS) (Cormen et al 1990). DFS is a graph-traversal algorithm which explores as far as possible along each branch before backtracking and searching other branches. For robotic search, this entails deploying robots serially down a single corridor branch, instead of deploying across multiple corridor branches in parallel, as would be the case with breadth-first search. Corridor branches are searched in clockwise order. DFS is utilised because it has minimal space complexity (Cormen et al 1990), so it maximises the branch depth that can be searched with a chain of robot beacons, thereby maximising search area. DFS can search cyclic graphs (and corridor environments) as long as a memory of visited branches is maintained. The beacons in corridor junctions act as an embedded memory ensuring no infinite loops are possible. Informed search techniques are not possible without heuristics based on *a priori* information on the target distribution. However, the proposed algorithm could be extended to select corridors that have a higher probability of leading to a target if such information is known.

In open spaces, we want to ensure all of the space is completely covered by the beacon's sensors as quickly as possible. This is achieved with a uniform dispersion behaviour where explorers are directed to the nearest empty space. This outward expansion of the network in corridors and open space is referred to as the deployment phase. Once an area has been completely searched, the beacons detach and redeploy as explorers to search other unexplored areas, the redeployment phase. The beacons utilise the forwards and reverse hop-count gradients to select appropriate desired directions and achieve efficient deployment and redeployment across the beacon network, as explained below:

Deployment phase:

- In corridors, the increasing forwards hop-count gradient is used to find the direction away from the base to guide explorers to the edge of the network.
- In open spaces, the increasing reverse hop-count gradient is used to guide explorers to the nearest empty space.

Redeployment phase:

- In corridors, the decreasing forwards hop-count gradient is used to guide explorers backwards to the previous junction, or towards the base beacon.

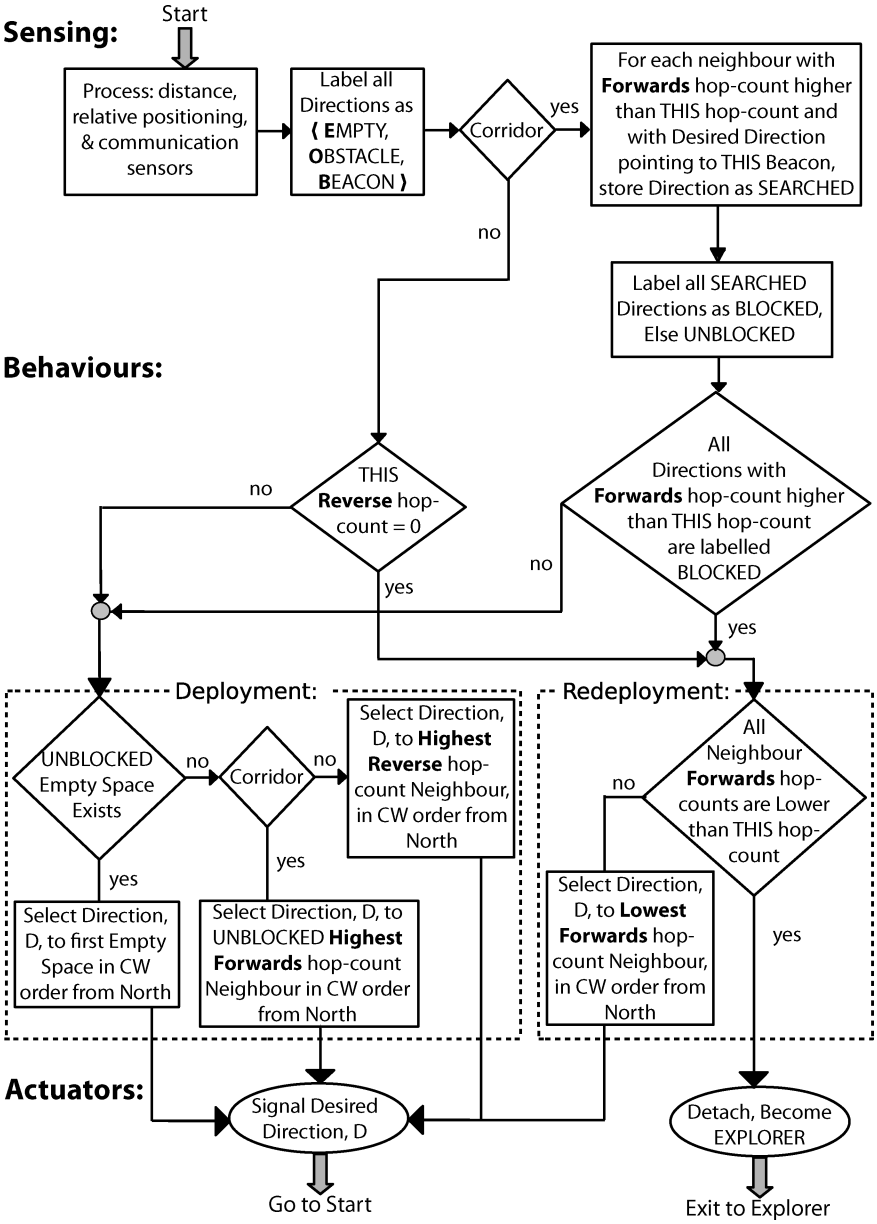

Fig. 2 Control flowchart for beacons. In the sensing stage, using the relative-positioning and distance sensors, all cardinal directions are labelled as either empty, obstacle or beacon. Previously searched directions are labelled as blocked. During deployment, the controller selects the direction to the first unblocked empty space in clockwise order if one exists, otherwise the neighbour with the highest hop-count. In open spaces the reverse hop-counts are used, in corridors forwards hop-counts. During redeployment, the desired direction is signalled pointing to the lowest forwards hop-count neighbour

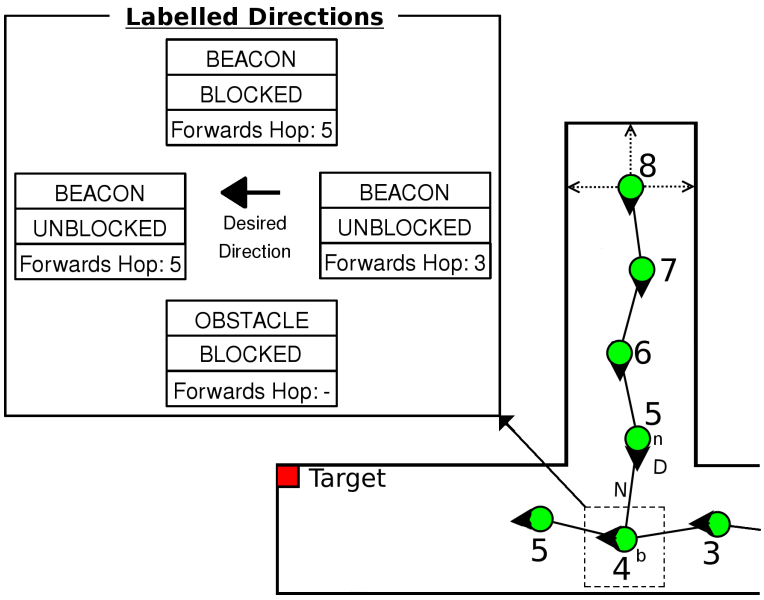

Fig. 3 *Labelled Directions*: An example of the labelled directions of beacon b with hop-count 4. Directions are labelled as either obstacle, empty space or beacon. If a direction has an obstacle, it is labelled as blocked, else unblocked. Since the beacon labelled n, with hop-count 5 north of beacon b with hop-count 4, has an opposed desired direction D pointing south (desired direction D opposes relative vector N, $D \simeq -N$) and has a higher forwards hop-count ($5 > 4$), beacon b stores the north direction as searched. All searched directions in memory are labelled as blocked. *Redeployment:* The beacon with forwards hop-count 8 is in a dead-end and has the highest forwards hop-count, therefore it signals a desired direction pointing backwards to the neighbour with the lowest forwards hop-count. It can then detach and become an explorer, ready to deploy to unexplored areas. The rest of the beacons in this corridor branch will detach recursively

– In open spaces, the decreasing forwards hop-count gradient is also used, guiding explorers to the exit of the open space, or towards the base beacon.

3.3.1 Beacon Controller

Here we describe the beacon controller from sensing to actuation as shown in Fig. 2.

Sensing: Beacons continuously monitor for neighbouring beacons or obstacles such as walls in the four cardinal directions using relative-positioning and distance sensors. A simple mapping function then labels each direction as either: 1) beacon, 2) obstacle, or 3) empty space. In corridors, beacons record whether a direction has been previously searched to prevent search duplication. A beacon b in a corridor considers a direction as searched when a neighbour beacon n, with a forwards hop-count higher than its own forwards hop-count, signals a desired direction D pointing backwards towards beacon b: $D \simeq -N$, where N is the relative-position vector pointing from beacon b to beacon n (see Fig. 3). All searched directions are stored in memory. All searched directions and those with an obstacle are labelled as blocked, otherwise as unblocked. Beacons in open space label all cardinal directions as unblocked. An example labelling is depicted in Fig. 3.

Deployment Phase Selection: The deployment or redeployment phase of beacons is selected in the Behaviours stage (see Fig. 2). Beacons in corridors determine the deployment phase by checking whether all neighbour beacons with a higher forwards hop-count are labelled as blocked. If this is true then there are no more unexplored directions and this beacon enters the redeployment phase, otherwise it continues in the deployment phase. The redeployment phase will start when a beacon detects it is in a dead-end, which is when an enclosed space is sensed with the distance sensors and all neighbouring beacons have a lower forwards hop-count (see Fig. 3). Beacons in open space determine the deployment phase by checking if their reverse hop-count equals 0. A reverse hop-count of 0 indicates that there is no frontier beacon (see Fig. 1) present emitting a reverse hop-count of 127, since without this signal the reverse hop-counts will automatically reduce to 0, and therefore no more beacons are required in this open space. Thereby, a reverse hop-count of 0 indicates the beacon is in the redeployment phase. Otherwise, it is in the deployment phase.

Deployment Phase: In the deployment phase, if an unblocked empty space direction exists then the first such direction in clockwise order from North is selected as the desired direction. Nearby explorers will perceive the desired direction signal and, using the explorer behaviour described in Sect. 3.4, will travel to the neighbouring empty space and become a beacon. This results in a local expansion of the network at the frontier beacons. Propagating beacons signal a desired direction pointing to neighbouring beacons guiding explorers to the network edge. This is achieved by assigning the desired direction to point to the neighbour with the highest hop-count that is unblocked; forwards hop-counts are used in corridors and reverse in open spaces (see Fig. 2).

Redeployment Phase: In the redeployment phase, dead-end beacons (see Fig. 3) signal a desired direction pointing backwards towards the neighbour with the lowest forwards hop-count. This indicates to the preceding beacon that this direction has been completely searched. The direction is stored in memory and labeled as blocked, as described earlier. This information is propagated backwards recursively to preceding beacons which will enter the redeployment phase and also signal a desired direction pointing backwards to their lowest forwards hop-count neighbour. These backwards desired directions also guide any nearby explorers to the previous corridor junction where they can then be guided to new unexplored locations. Open space beacons act similarly and signal a desired direction pointing to the neighbour with the lowest forwards hop-count. Beacons in the redeployment phase can revert back to explorers and detach from the ceiling to redeploy to unexplored environment areas. This occurs only when all neighbours have a lower or equal forwards hop-count, thereby preventing breaking communication links with neighbours (see Fig. 3). All beacons wait the designated inter-launch period before detaching to gain the same benefits as the incremental deployment (described in Section 3.4) and additionally ensure there are no proximal explorers in order to avoid potential collisions. Once the environment is fully searched, robots can incrementally return to the base area and land if required.

3.4 Explorer Behaviour

Explorers start clustered on the ground below the base beacon. They take-off one at a time every t seconds, where t is an adjustable inter-launch period, typically in the range 3-24 s. We call this behaviour temporal incremental deployment. Longer inter-launch periods slow deployment but importantly decrease the density of flying robots reducing collision risks and

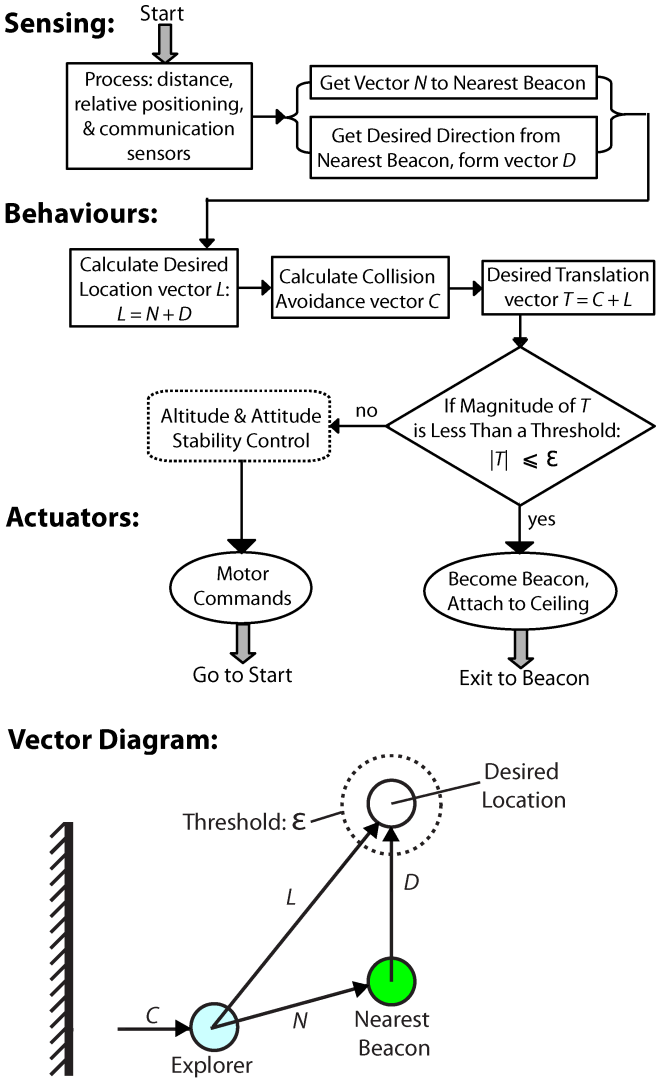

Fig. 4 *Top*: Control flowchart for flying explorers. In each control-step, explorers use their relative-positioning and communication sensor to retrieve the vector N to the nearest beacon and the associated desired direction vector D, and can then compute desired location vector L. A collision avoidance vector, C, is additionally calculated which points away from the nearest obstacle with magnitude inversely proportional to the obstacle's distance. The desired location vector L and collision avoidance vector C are combined to form the translation vector T. If the magnitude of T is less than a threshold the explorer becomes a beacon, otherwise the translation vector is sent to the flight computer. *Bottom*: Vector diagram of explorer behaviour

increasing search efficiency by exploiting more information from the deployed sensor network. This is because, as the beacon network expands, it senses more of the environment. Once a subarea has been completely searched, this is detected and the beacons redeploy to new unexplored areas (see Sect. 3.3.1: *Redeployment*). Before this redeployment commences there may be multiple explorers flying into this subarea where they are not required,

resulting in unnecessary flight. Increasing the inter-launch period decreases the number of concurrent flying explorers, thus reducing unnecessary flight (see results in Sect. 4).

3.4.1 Explorer Controller

To implement the incremental deployment, explorers wait for the elapsed time since the start of deployment to surpass their ID (an integer 1...N) times the inter-launch period. They take-off and ascend to a designated altitude where they then follow the desired direction signal of the nearest beacon, which is initially the base beacon. As explorers fly towards a neighbouring beacon, this neighbour will become the nearest, and so on.

Fig. 4 *top* shows the flowchart of the explorer controller. The relative-positioning sensor is used to determine the vector N pointing towards the nearest beacon. Explorers implicitly know the desired interspace distance I_r, therefore vector D is created pointing from the nearest beacon in the desired direction with length I_r. Vectors N and D combine to form the desired relative location vector: $L = N + D$ (see Fig. 4 *bottom*). Explorers continuously compute the vector L for each control time step. Explorers follow the desired directions and use their distance sensing for local collision avoidance. Collision avoidance is achieved with a potential field behaviour (Khatib 1985) that creates a vector C pointing away from the closest detected obstacle with magnitude inversely proportional to the obstacle distance, similar to what we have previously implemented with our robot prototype (Roberts et al 2007). The magnitude of the collision vector was empirically tuned to minimise collisions without adversely effecting the algorithm's performance. When the explorer is at the mid-point between obstacles, the collision avoidance vector reduces to zero magnitude within the local-minima of the potential field. The desired relative location vector L is linearly added with the weighted collision avoidance vector C to create the desired translation vector T: $T = L + C$. The translation vector T is then sent to the flight computer which performs altitude and attitude stability control and then provides desired pitch and roll forces to a motor controller. However, when the magnitude of the translation vector T is lower than a threshold, $|T| \leq \varepsilon$, the explorer ascends to the ceiling, attaches and becomes a beacon. This occurs when the nearest beacon's desired direction vector D approximately opposes the nearest beacon vector N and there is no collision response: $D \simeq -N$ and $|C| \rightarrow 0$.

3.5 Experimental Methodology

To verify the proposed algorithm and to explore the effects of different inter-launch periods we have conducted extensive simulation analysis. Here we describe the simulator, generation of test environments, and performance metrics.

Dynamics Simulation: Experimental analysis has been conducted with results from a 3-D dynamics simulator with a simple model that applies appropriate forces to the robot body with Gaussian noise. The resultant behaviour includes momentum and drag effects. Details are described in Appendix B.

Test Environments: Because of the hybrid nature of our strategy, we individually tested the performance in corridors, in open spaces with obstacles and in combination (see examples in Fig. 5). Open space environments were randomly generated 5×5 cellular grids with each cell being 3 m $\times$ 3 m. Low, medium and high obstacle densities were generated with 5, 7 or 10 obstacles, respectively. Obstacles were randomly placed on the grid while ensuring that

all free-space was accessable and no corridors were inadvertently generated. Corridor environments were similarly generated from a larger grid with 40 connected 3 m $\times$ 3 m corridor cells, creating 120 m of corridor. The amount of corridor branching was adjusted probabilistically, creating 3 levels of environment complexity termed low, medium and high. The mean branch count (and std. dev.) was empirically measured at 3.56 (1.23), 5.82 (1.07), and 9.5 (1.63), respectively for low, medium and high complexity environments. Environments with a greater branching naturally have shorter corridors on average since the total corridor length was kept constant. The mixed environments with both corridors and open spaces were generated with a combination of these methods and consisted of two rooms of 3 $\times$ 3 cells (9 m $\times$ 9 m) and 22 corridor cells, so approximately half the environment being corridor and half open space. Medium complexity corridor branching was used to generate the corridors.

20 robots were available to deploy in corridor and mixed environments, clustered in a random location. Beacons were separated by the interspace distance I_r of 3.0 m. Therefore, in mixed and corridor environments with a static deployment of beacons, only approximately 50% of the area could be covered with 20 robots ($\frac{20\times3.0}{120}$). However, due to redeployment, the possible coverage was greater. In low, medium and high obstacle density open environments, 15, 18 and 20 robots, respectively, were available, sufficient for complete coverage.

Performance Metrics: We measured coverage time and area, collisions and total swarm energy consumption. Coverage time is the time taken to complete the search of the environment or for completion of the algorithm. Coverage area is defined as the percentage of the environment that is covered by the sensors of all beacons at least once. We assume a fictitious sensor such as a camera is used to detect targets, which has a field-of-view covering a circular area with radius equal to the defined sensor radius S_r. Collisions is the total sum of collisions both between robots and with obstacles, calculated using the simulator physics engine (see Appendix B). A collision between 2 robots is counted as 2 collisions. Total swarm energy consumption is the sum of the energy used by each robot calculated using the energy model described in Appendix C. We measure the total energy each robot spends flying as an explorer, attached to the ceiling as a beacon or resting on the ground before launching. We compared different inter-launch periods (Sect. 3.4) to understand the effect on mean coverage time, collisions and energy efficiency. 100 trials were repeated for each condition with random environments and starting locations.

4 Results

Corridor Environments: Due to our strategy's redeployment mechanism, the mean coverage area is significantly higher than the approximately 50% expected for static deployment; also, it increased with increasing environment complexity (79.4%, 90.1%, and 94.0%, averaged across all inter-launch periods). This increase is due to shorter average corridor lengths with increased branching as the corridor complexity is increased. With shorter corridor lengths, the robot swarm is more likely to reach the end of more corridors before redeployment is required. The percentage of trials achieving complete coverage at low, medium and high corridor complexity was 38.5%, 64.2% and 74.3%, respectively.

We compared the mean coverage time, coverage area, total swarm energy and collisions for 6 different inter-launch periods (6, 8, 10, 12, 18 and 24 seconds, selected from prior data-exploration experiments) and the 3 corridor complexity levels (low, middle and high) using multiple 6$\times$3 2-way *Analysis-of-Variance* (ANOVA) tests. As expected, increasing the inter-launch period increases the coverage time ($F = 510.11$, df = 5, $p < 0.001$), as shown in

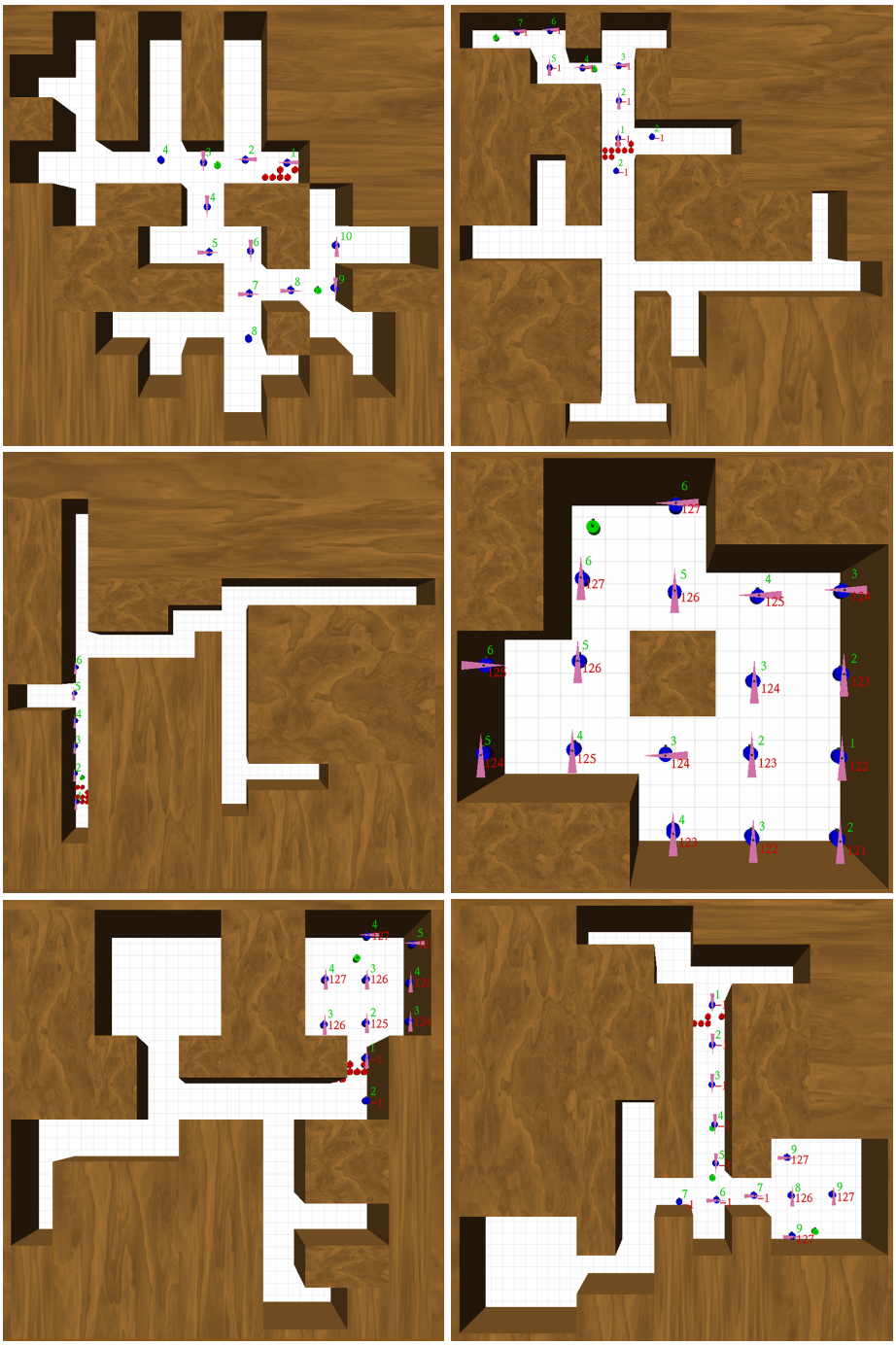

Fig. 5 Examples of randomly generated test environments. *Top Left*: High complexity corridor environment. *Top Right*: Medium complexity corridor environment. *Middle Left*: Low complexity corridor environment. *Middle Right*: Open Space environment with medium obstacle density *Bottom Left & Right*: Mixed environments

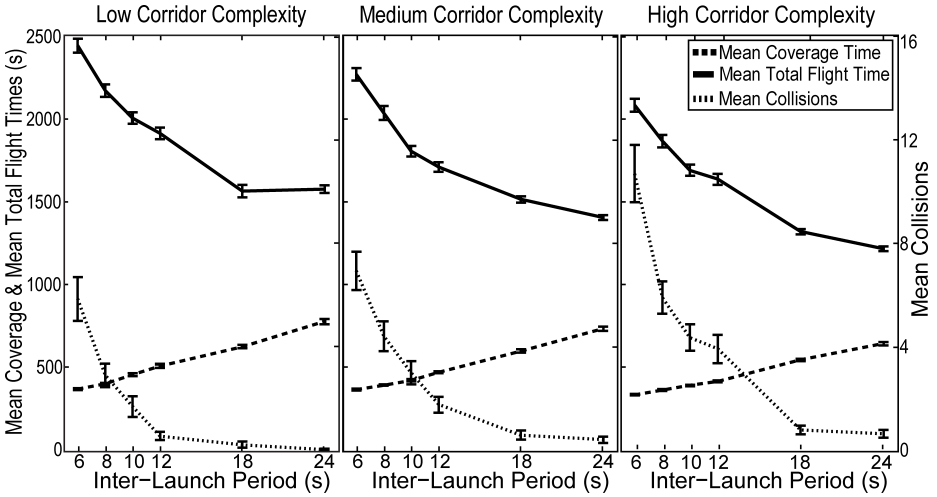

Fig. 6 Corridor Environments: Mean coverage time, mean total flight time and mean collisions (with standard error bars), against 6 inter-launch periods for 3 corridor complexity levels. Increasing the inter-launch period reduces mean total flight time and mean collisions, but with a trade-off with coverage time

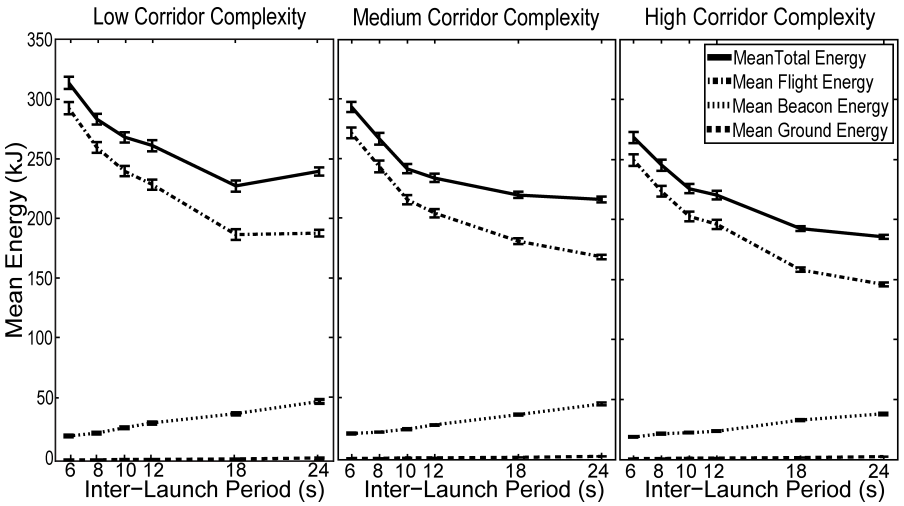

Fig. 7 Corridor Environments: Mean total energy consumption and the constituent mean flight, ground and beacon energies (with standard error bars), against 6 inter-launch periods for 3 corridor complexity levels. Increasing the inter-launch period significantly reduces the mean total energy cost of the search, despite an increase in ground and beacon energies because of the dominating effect of reduced flight energy

Fig. 6, but has no significant effect on the mean coverage area ($F = 1.61$, df = 5, $p > 0.15$). Importantly, increasing the inter-launch period significantly reduces the total flight time of the swarm ($F = 67.29$, df = 5, $p < 0.001$), and the number of collisions ($F = 8.47$, df = 5, $p < 0.001$), both shown in Fig. 6. The increase in inter-launch periods results in larger distances between flying explorers, reducing inter-robot collisions. The slower deployment also results in increased exploitation of the sensed environment through the beacon sensor network reducing the total swarm flight time. This has important effects on the swarm energy

consumption which significantly decreases with increased inter-launch periods (F = 33.28, df = 5, $p < 0.001$). Fig. 7 shows the different components of the energy model (see Appendix C) and the corresponding reduction in swarm energy consumption due to the reduction in flight energy, despite the increase in energy cost of beacons and robots on the ground. The mean total swarm flight time is a strong predictor of the mean swarm energy consumption with a Pearson's correlation coefficient of $r = 0.974$ (df = 16, $p < 0.001$).

We additionally examined environmental complexity effects with varying amounts of corridor branches. Decreased corridor complexity significantly increased coverage time (F = 20.47, df = 2, $p < 0.001$) and total swarm energy consumption (F = 25.29, df = 2, $p < 0.001$), but decreased mean collisions (F = 9.01, df = 2, $p < 0.001$) and mean coverage area (F = 160.48, df = 2, $p < 0.001$). Lower corridor complexity results in an increase of average corridor length. Long corridors are less likely to be covered than short corridors because of the maximum length that can be spanned by a given number of connected robots. Thus, coverage area decreases with reduced corridor complexity. Similarly, the increased average corridor length in less complex environments requires longer to traverse. This increases the mean coverage time and mean total flight time, thus increasing the swarm energy consumption. Simpler environments with fewer junctions reduced collisions with the environment. All interactions between corridor complexity and launch periods on mean coverage time, total flight time, energy and collisions were comparatively minor but statistically significant.

Fig. 6 shows the time required to achieve complete coverage (to ensure constant coverage area), total swarm flight time and mean collisions, against the 6 inter-launch periods (6-24 s) for the 3 different corridor complexities (low to high). Comparing inter-launch periods of 24 and 6 s within the high complexity corridor environment the total swarm flight time was reduced by 41.5% and the mean collisions reduced by 93.9%, while the mean coverage time increased by 90.5%. Fig. 7 shows that increasing the time between consecutive robot launches significantly reduces the swarm energy consumption by 30.8%. This implies the swarm could search a significantly larger environment before running out of energy, or prolong the time the beacon network could perform monitoring tasks. The other corridor complexity environments show similar patterns but with decreased magnitude.

Open Space Environments: We compared 5 different inter-launch periods (3, 4, 5, 6 and 12 s) with 3 amounts of obstacles (5,7,10). Mean coverage area across all conditions was 99.7% with no significant variance across different test conditions. Similar to the corridor environments, increasing the inter-launch periods significantly increased the coverage time (F = 9528.5, df = 4, $p < 0.001$), but decreased total flight time (F = 262.38, df = 4, $p < 0.001$) and collisions (F = 23.5, df = 4, $p < 0.001$), when normalised for the varying swarm size (15, 18 and 20 robots) with different obstacle densities. The reduction in total swarm flight time with increased inter-launch periods also significantly reduced the swarm energy consumption (F = 83.65, df = 4, $p < 0.001$). Obstacle density significantly effected the normalised total flight time (F = 61.23, df = 4, $p < 0.001$) and swarm energy (F = 115.07, df = 4, $p < 0.001$) but not collisions (F = 0.17, df = 4, $p > 0.8$).

Fig. 8 shows the mean time required to achieve complete coverage, mean total swarm flight time and mean collisions against the 5 inter-launch periods for the 3 obstacle densities. Comparing the inter-launch period of 12 s with 3 s within the high obstacle density environment, mean total swarm flight time is reduced by 29.9% and correspondingly the mean swarm energy consumption reduced by 16.4%, the mean collisions reduced from 0.32 to 0.02, while the mean coverage time increased by 149.7%. This is a significant increase in mean coverage time. However, the asymptotic nature of the trend shown in Fig. 8 indicates that the greatest decrease in total flight time, for all obstacle densities, occurs when

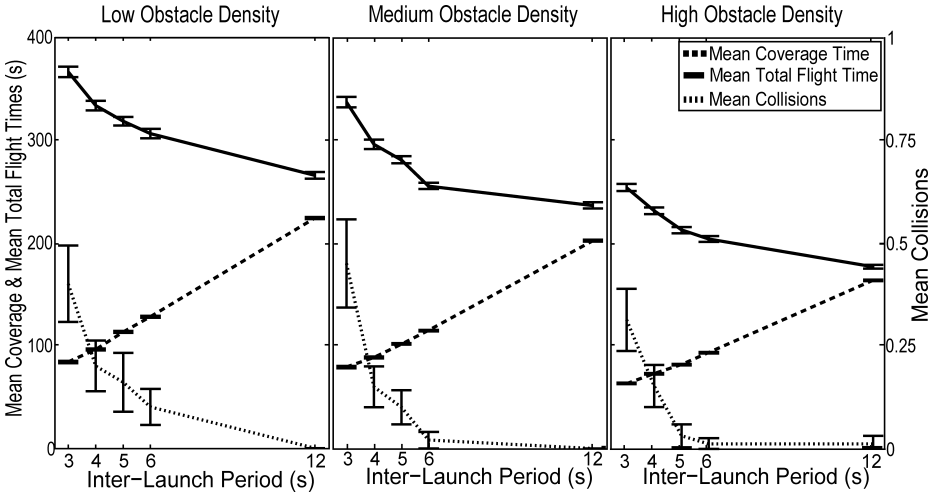

Fig. 8 Open Spaces: Mean coverage time, mean total flight time and mean collisions (with standard error bars) against 5 inter-launch periods at 3 levels of obstacle density. Increasing the inter-launch period reduces mean total flight time and mean collisions, but with a trade-off in increased mean coverage time

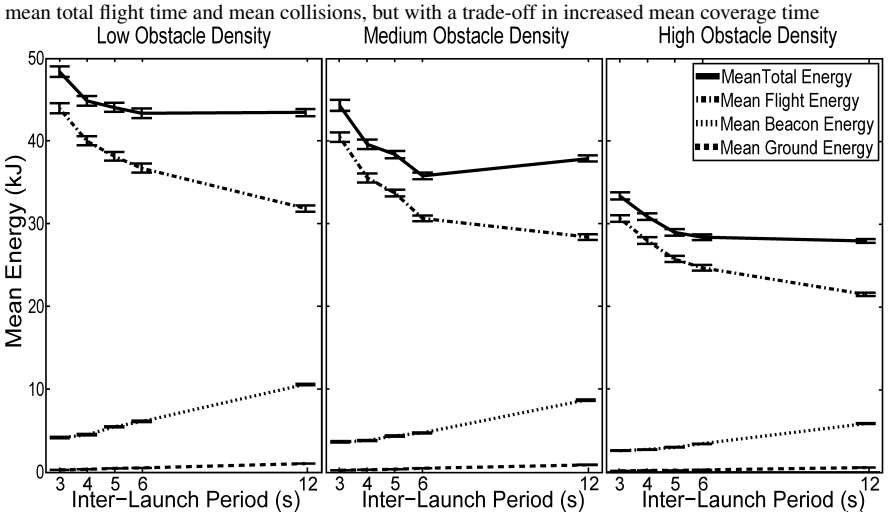

Fig. 9 Open Spaces: Total energy consumption and the constituent flight, ground and beacon energies of the swarm (with standard error bars), against 5 inter-launch periods at 3 levels of obstacle density. Increasing the inter-launch period significantly reduces the total energy cost of the swarm search because of the dominating effect of reduced flight energy. However, the increase in ground and beacon energies at larger inter-launch periods can reduce the efficiency gain and can eventually reverse any energy savings, as can be seen in the medium obstacle density case

increasing the inter-launch period from 3 s to 6 s and not from 6 s to 12 s. When comparing the 3 s to 6 s launch periods, the mean total flight time is still reduced by 15.0% while the total coverage time increases by only 44.9%. Moreover, the increased energy cost of the ground and beacon energies with increased inter-launch periods can result in an increase in swarm energy consumption when the launch period increases from 6 s to 12 s, creating a convex function, as shown in Fig. 9 with the medium obstacle density environment. This

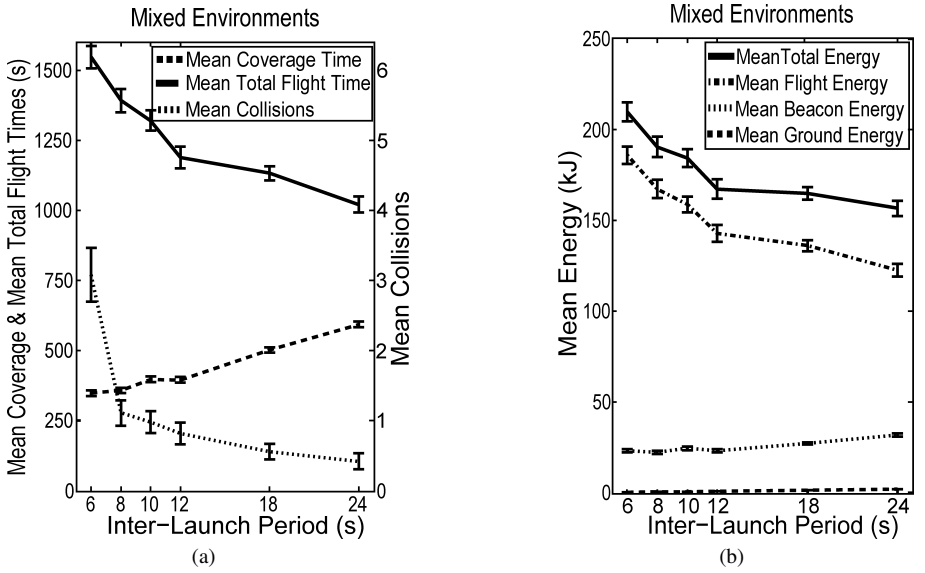

Fig. 10 Mixed Environments: (*a*) Mean coverage time, mean total flight time and mean collisions, and (*b*) mean total energy consumption and the constituent mean flight, ground and beacon energies of the swarm; against 6 inter-launch periods in mixed environments. Standard error bars shown. Increasing the inter-launch period significantly reduces the mean total energy cost of the swarm search

indicates the existence of an optimal inter-launch period which minimises the swarm energy consumption. However, such an optimum is dependent on the environment.

Mixed Environments: We compared 6 different inter-launch periods (6, 8, 10, 12, 18 and 24 seconds). Mean coverage area did not vary significantly ($F = 1.59$, df = 5, $p = 0.16$). Increasing the inter-launch periods significantly increased the mean coverage time ($F = 99.85$, df = 5, $p < 0.001$) as expected, and significantly decreased mean total swarm flight time ($F = 28.75$, df = 5, $p < 0.001$) and mean collisions ($F = 22.34$, df = 5, $p < 0.001$). The reduction in mean total swarm flight time with increased inter-launch periods also significantly reduced the mean swarm energy consumption ($F = 16.52$, df = 5, $p < 0.001$).

Fig. 10(a) shows the mean total swarm flight time, mean collisions and the mean coverage time for the 6 launch periods. Comparing the inter-launch period of 24 s with 6 s, the mean total swarm flight time was reduced by 34.0% and the mean collisions reduced by 86.4%, while the coverage time increased by 70.5%. Fig. 10(b) shows the mean total swarm energy consumption and the corresponding flight, beacon and ground energies. As with the other environment types, the increase in inter-launch period from 6 s to 24 s significantly reduced the energy consumption by 25.3% due to the reduction in energy spent in flight.

Theoretical Results: We have also derived some basic theoretical results to analyse scalability effects of the swarm algorithm. The *worst-case* coverage area A_c is linearly proportional to the beacon separation distance I_r and the swarm size n: $A_c \propto nI_r$. For example, in long corridors without any junctions the robot beacons can at worst cover a corridor length of nI_r. However, in the *best-case* the length of corridor that is searchable increases quadratically

with the number of robots in corridors with numerous junctions, $A_c \propto (nI_r)^2$. This can be intuitively derived from the area which can be covered by a linear chain of robots separated by I_r that rotates around the first robot in the chain, resulting in a circular coverage area given by $\pi(nI_r)^2$. This result is constrained by the possible architecture of indoor environments but indicates the beneficial scaling properties of the proposed algorithm, considering that the processing and memory costs are constant and independent of environment or swarm sizes. With respect to coverage time and energy efficiency, there is a complex interaction between environment structure, inter-launch period and robot velocity precluding simple predictions, but this is a focus of future work.

With respect to the energy efficiency of ceiling attachment, if the robot beacons had to hover in position rather than attach to the ceiling, the swarm search would have required 3.6× more energy, based on the results for mixed environments with a 6 s launch period. Different environment types and complexities exhibit similar efficiency gains. This highlights the importance of ceiling attachment for search or surveillance with aerial robotics.

5 Discussion

The presented algorithm is limited to regular environments aligned in the cardinal compass directions as a simplification. However, this simple approach can be generalised to more complex environments. For example, if the robots can measure distances in a higher angular resolution than just the 4 cardinal directions, such as with a rotating distance scanner, then a more detailed analysis of the beacon's local environment is possible, allowing the perception of oblique corridors and irregularly shaped environments. This can still be achieved with the direction labelling scheme described in Sect. 3.3. The desired direction signal must increase angular resolution and perhaps be generalised into an appropriate coarsely encoded 2-D vector. Importantly, such a vector is still communicable with low-bandwidth communication, and computation and memory utilisation remain minimal.

As well as generalising to more complex environments, the proposed approach is suitable for most hover-capable flying robots. The main specialised feature of our robots is their ability to attach to ceilings (Roberts et al 2008). However, various perching mechanisms are being researched (e.g., Desbiens et al 2009) using technologies such as gecko-inspired elastomer adhesives (Unver et al 2006). However, flying robots could simply land on the ground to become a beacon, but loosing the advantage of maintaining a bird's-eye view.

The proposed algorithm requires standard sensory and processing systems that are typical for small mobile robots. However, the proposed approach also depends on an appropriate relative-positioning sensor. Many similar sensors exist (e.g., Payton et al 2001, Pugh et al 2009, Melhuish and Welsby 2002, McLurkin and Smith 2007) and alternative technologies are possible using ultrasound (Bisson et al 2003) or computer vision (Nakamura et al 2003), etc. In nature, birds often use visual processing in order to perceive the position of neighbours during tasks like flocking, achieving high precision spatial coordination even under highly dynamic flight (Ballerini et al 2008). Our proposed infrared relative-positioning sensor (Roberts et al 2009) provides a simple robust solution with low processing requirements.

So far we have not examined the effects of robot failure. Explorer failure should merely reduce the searchable area with an effect proportional to the swarm size (see Sect. 4: *Theoretical Results*). However, beacon failure may result in disruption to the communication network. There should be little effect in open spaces where multiple communication pathways are available and sensor and communication ranges overlap sufficiently. However, in corridors with only a single chain of beacons, beacon failure could lead to failure of the

whole communication network. The effect of this would be to disrupt communication links to the base robot, breaking navigation paths to or from the base. To mediate this, the failure should be detected via an unexpected break in communication with a neighbour. The failed beacon can then be replaced by either a nearby explorer, or a beacon from the edge of the robot network, which could be guided by an additional hop-count gradient signal emanating from the neighbour of the failed beacon. Efficiently replacing failed nodes in a sensor network has been extensively examined by Wang et al (2005). With regards to errors in the desired direction signals of beacons, currently no explicit checks are made in order to prevent deadlocks or local cycles (where following the path of desired directions returns an explorer to the starting beacon). However, these could be detected by either explorers having a memory of visited beacons and associated desired directions, or communication packets could be routed following the desired directions and the communication path logged in the packet. If the explorer or communication packet continuously reaches the same beacons via the same path then a cycle exists, which can then be corrected. Importantly, environment cycles in corridors are handled automatically, see Appendix A.

We have shown the effects of varying the inter-launch period on coverage time, collisions and swarm energy cost. From the results (Figs. 6 to 10(b)), the general trend clearly shows that increased inter-launch periods reduce collisions and swarm energy cost but with a trade-off with increased coverage time. Interestingly, the effects are more pronounced in more complex environments since the swarm will redeploy to new environment subareas more frequently. There exists an optimal inter-launch period that minimises swarm energy cost since the asymptotic reduction of energetic flight quickly levels off, while the increase in coverage time increases the energy robots use as beacons. This optimal inter-launch period is dependent on the environment type and complexity. Furthermore, the optimum will be application dependent with a specific trade-off in coverage time for reduced collisions and energy cost. For a specific environment the optimal inter-launch period can be easily calculated given the relative weightings for coverage time, collisions and energy. Cubic interpolations of the mean coverage time, collisions and energy for the specified environment can be generated to create high resolution smooth respresentations. These interpolations can be normalised to $[0, 1]$ and the linear weighted sum computed with the specified relative weightings. The location of the minimum of this weighted sum provides the optimal inter-launch period.

Finally, ongoing work aims to further improve energy efficiency using different deployment schemes, flight-path optimisation and heterogeneous sensing. One possibility for increasing efficiency is an adaptive group-size mechanism that automatically adjusts the number of deployed explorers based on implicit social cues such as perceived robot density. Such a mechanism may better adapt to different environments and continuously adapt at runtime, compared with the temporal incremental deployment proposed in this paper.

With the promising results obtained in simulation, we aim at transferring the algorithm to real robots. So far we have demonstrated autonomous indoor operation of a single quadrotor flying robot including much of the important low-level control required to realise the proposed algorithm. Roberts et al (2007) presented a prototype robot performing autonomous stability control, take-off, landing, and altitude control with a mean absolute error of under 27 mm. A collision avoidance behaviour similar to what is used in the proposed algorithm has also been successfully demonstrated as well as a method for controlling platform drift by centering in a room using the distance sensors (Roberts et al 2007). Furthermore, we have developed reliable autonomous ceiling attachment and detachment control (Roberts et al 2008), as used by the explorers becoming beacons. The energy efficiency of attachment to the ceiling and the accuracy of the energy model of the motor system have been quanti-

tatively tested, with a predicted flight endurance error of less than 1% (Roberts et al 2008). The communication protocol and control and processing of the distance scanner have been successfully demonstrated. In addition to the above work we have developed a custom-made infrared relative-positioning sensor suitable for flying robots (Roberts et al 2009), based on the technology presented by Pugh et al (2009). This sensor is important for the spatial coordination used in the proposed algorithm and to control the platform drift associated with hovering robots (Hoffmann et al 2007). This is part of current research, so far we have prototyped the anti-drift controller within simulation and are awaiting new flying robot hardware before commencing testing. Once this has been achieved, all important low-level flight control will be complete and we then aim to develop the proposed algorithm on the novel flying robots we are developing. Further details, photos and videos of the progress in hardware development can be found on our website[2].

6 Conclusions

We presented a novel method to realise indoor search by swarms of flying robots. Utilising flying robots offers the advantage of easily overcoming obstacles and, therefore, it allows fast terrain coverage. However, GPS, as commonly used with outdoor flying robots, cannot be used indoors. In this paper we presented an algorithm that solely relies on local sensing and local communication. We proposed the self-deployment of a network of robot beacons which facilitates simple search and navigation without the need for high-bandwidth global communication, *a priori* world models, synchronisation, absolute positioning, or odometry sensing. Long range communication is provided using the robots' short range communication by deploying a communication and sensor network of locally connected robot relays. To achieve autonomous indoor aerial robotic control, we avoided utilising the commonly used methods of absolute positioning with external 3-D tracking sensors (which requires prior deployment) or complex visual processing (not available on-board and which would require appropriate illumination). Instead, we proposed to use the local on-board relative-positioning sensors of nearby "beacon" robots to directly control flight.

To establish a sensor network with flying robots, we proposed that robots can switch from active flying to a passive surveillance mode. In our case robots could attach to the ceiling and power down their rotors. This increases operational endurance, which is usually constrained by the limited flight time of flying robots. To further reduce energy consumption, we introduced an incremental deployment scheme. Robots are launched consecutively, separated by an adjustable inter-launch period. We have shown that the algorithm can successfully search indoor environments and that increasing the inter-launch period significantly reduces the swarm energy consumption by up to 30.8% in our simulations. Furthermore, incremental deployment reduces robotic density and hence the risk of collisions.

Our results indicate that there may be a trade-off between energy efficiency and search time. However, this can be optimised with the relative importance of each variable being application specific. In some time-critical scenarios the importance of energy efficiency may be small relative to coverage time, but it is clear that if the robots do not have sufficient energy to complete the task then energy efficiency becomes paramount.

[2] `http://lis.epfl.ch/research/projects/Eyebot`

7 Acknowledgements

This work is part of the “Swarmanoid Project”, a Future Emerging Technologies (FET IST-022888) project funded by the European Commission, and is also supported by the Swiss National Science Foundation Grant No. K-23K0-117914. The authors would like to thank James Roberts who provided support with technical details concerning hardware, and the reviewers and editors who provided extensive valuable feedback.

Appendix

A Hop-Counts

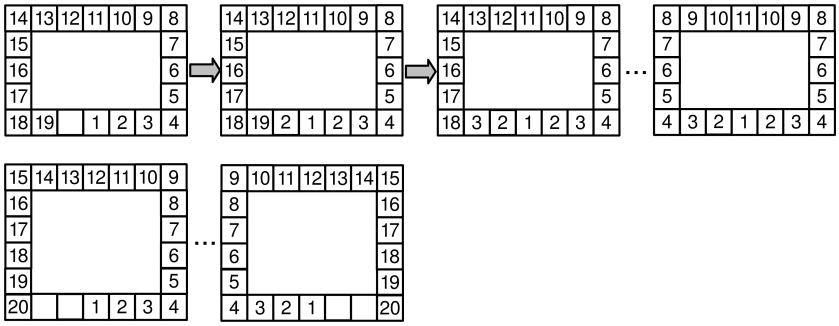

Fig. 11 Forwards hop-counts in cyclic environments. *Top*: If the cycle is shorter than the length coverable by the swarm then the network self-connects and correct hop-counts are propagated back. *Bottom*: If the cycle is longer than can be covered by the swarm then the network will redploy and search from the other direction

Hop-counts are calculated using small communication packets sent through the relative-positioning sensor. This is used to ensure neighbouring beacons have numerically adjacent hop-counts (i.e.$\pm$ 1). From a graph-theoretic point of view, if each beacon is considered a vertex in a graph representing the beacon sensor network, then the forwards hop-counts are simply the minimum number of edges between each beacon and the base beacon. Conversely with reverse hop-counts and frontier beacons. Hop-counts are continuously updated to ensure the shortest path is discovered. This is important in open space areas where the shortest path to the nearest frontier beacon constantly changes as the network expands. In cyclic corridor environments, if the beacon network self-connects, the terminal beacons may initially have disjoint hop-counts (see Fig. 11 *top*). However, due to the dynamic hop-count updates, the correct values are quickly propagated through the network (see Fig. 11 *top*). If the cycle is shorter than the length coverable by the swarm, the network self-connects, the correct hop-counts are propagated back and the shortest path to the base is ensured (Fig. 11 *Top*). Cycles longer than can be covered by the swarm will result in the beacon network redeploying and searching from the opposite direction. If the corridor length is less than twice the distance coverable by the swarm the complete cycle is searched (Fig. 11 *Bottom*); however, the shortest path from a target to the base may not be guaranteed. Cycles too long to be fully searched when searched from both directions can be simply considered as two separate corridors that are both too long to cover. These results hold even if the cycle connection occurs in a corridor junction.

B Flight Dynamics

A custom made 3-D dynamics simulator was developed in order to conduct experimental analysis. Appropriate forces are applied to a rigid robotic body. An input force vector F of the robot is calculated and applied to

the underlying dynamics simulator, for which we use the Open Dynamics Engine[3] (ODE). ODE is also used to detect and measure the number of collisions, as well as to provide the rigid-body collision response. The collisions are direct point-point collisions. Attitude and yaw-rate are assumed to be stabilised by a low-level stability controller such as presented by Gurdan et al (2007). Gaussian noise was applied to the body state to simulate imprecise dynamics and platform drift, with std. dev. set empirically (2.5 cm) as the platform has not yet been characterised. However, the altitude control precision was modeled from previous work (Roberts et al 2007). The input vector is given by:

$$F = F_g + F_c + F_d + \varepsilon_N \ , \tag{1}$$

where ε_N is a Gaussian noise vector for the roll, pitch and thrust standard deviations: $\varepsilon_N \sim \mathcal{N}(0, \hat{\sigma})$. Platform weight F_g, desired control force F_c and air-drag F_d vectors are calculated as follows. The platform weight vector is simply given by $F_g = m \cdot g$. The control force vector is formed from the desired pitch f_p and roll f_p forces, calculated from the explorer algorithm (see Sect. 3.4), combined with the altitude control thrust f_a, from a proportional-differential controller with an *a priori* command to compensate the platform weight:

$$F_c = \begin{pmatrix} f_p \\ f_r \\ f_a \end{pmatrix} \ . \tag{2}$$

The air-drag force vector is calculated using the standard quadratic equation suitable for flying systems with a relatively high Reynolds number:

$$F_d = -\frac{1}{2}\rho v^2 A C_d \hat{v} \ , \tag{3}$$

where ρ is the specific air mass-density, v is the scalar translational air-speed of the platform, A denotes the estimated frontal reference area, C_d is the estimated drag coefficient and $\hat{v}$ is the normalised velocity vector.

Sensor noise was modeled as Gaussian noise with std. dev. measured from characterisation experiments: 2.5 cm for the ultrasound altitude sensor and 5 cm for infrared distance sensors. The relative-positioning sensor uses values from (Pugh et al 2009): at 3.0 m the range std. dev. is 17 cm and bearing std. dev. is 6.1°.

C Energy Model

In order to measure the swarm energy consumption, a simple energy model suitable for flying robots was developed. Various energy models have been proposed for terrestrial robots (e.g., Mei et al 2005). However, due to the substantial differences in dynamics, a new model was developed based on our previous work (Roberts et al 2008), which showed a mean error of only 0.97% in predicting flight endurance. We categorise the robot power consumption into two parts: the electrical motor power E_m required for flight, and the electronic power E_e required to power sensors, processors and communication. From (Roberts et al 2008), we assume a nominal hovering motor power consumption of $E_m = 110$ W. To formulate the electronics power model, many components have been characterised (relative positioning sensor, CPU, flight computer) and manufacturer data-sheets used elsewhere (infrared distance and ultrasound altitude sensors). We have generalised the electronics power model to have 3 energy rates: low, medium and high. High power mode is used by flying robots which require all sensors to be operating continuously at high refresh rates, including the flight computer, ultrasound altitude sensor and infrared distance sensors. Medium power mode is used by beacons since only some sensors are required and they can operate under efficient lower refresh rates (Mei et al 2005). Low power mode is used exclusively by robots that are waiting on the ground before launching when only infrequent communication is required. The corresponding power consumption for low, medium and high power modes are $E_{e_L} = 0.5$ W, $E_{e_M} = 5$ W and $E_{e_H} = 10$ W, respectively. Importantly, this gives a total power consumption for flying robots of $E_m + E_{e_H} = 120$ W, compared to the relative low power of static beacons and robots on the ground of only $E_{e_M} = 5$ and $E_{e_L} = 0.5$ W.

References

Ahmadzadeh A, Buchman G, Cheng P, Jadbabaie A, Keller J, Kumar V, Pappas G (2006) Cooperative control of UAVs for search and coverage. In: Proceedings of the Conference on Unmanned Systems, AUVSI, Arlington, pp 1–14

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

Ballerini M, Cabibbo N, Candelier R, Cavagna A, Cisbani E, Giardina I, Orlandi A, Parisi G, Procaccini A, Viale M, Zdravkovic V (2008) Empirical investigation of starling flocks: A benchmark study in collective animal behaviour. Animal Behaviour 76(1):201–215

Batalin M, Sukhatme G (2002) Spreading out: A local approach to multi-robot coverage. In: Proceedings of the International Symposium on Distributed Autonomous Robotic Systems, Springer, Berlin, pp 373–382

Batalin M, Sukhatme G (2004) Coverage, exploration and deployment by a mobile robot and communication network. Telecommunication Systems 26(2-4):181–196

Batalin M, Sukhatme G, Hattig M (2004) Mobile robot navigation using a sensor network. In: Proceedings of the International Conference Robotics and Automation, IEEE Press, Piscataway, vol 1, pp 636–641

Baxter JL, Burke EK, Garibaldi JM, Norman M (2007) Multi-robot search and rescue: A potential field based approach. In: Mukhopadhyay S, Gupta G (eds) Autonomous Robots and Agents, Springer, Berlin, pp 9–16

Bisson J, Michaud F, Letourneau D (2003) Relative positioning of mobile robots using ultrasounds. In: Proceedings of the International Conference on Intelligent Robots and Systems, IROS '03, IEEE Press, Piscataway, vol 2, pp 1783–1788

Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York

Bryson M, Sukkarieh S (2007) Co-operative localisation and mapping for multiple UAVs in unknown environments. In: Proceedings of the Aerospace Conference, IEEE Press, Piscataway, pp 1–12

Burgard W, Moors M, Stachniss C, Schneider F (2005) Coordinated multi-robot exploration. IEEE Transactions on Robotics 21(3):376–386

Corke P, Peterson R, Rus D (2005) Localization and navigation assisted by networked cooperating sensors and robots. The International Journal of Robotics Research 24(9):771–786

Cormen TH, Leiserson CE, Rivest RL (1990) Introduction to Algorithms. MIT Press, Cambridge

Desbiens AL, Asbeck AT, Cutkosky MR (2009) Scansorial landing and perching. In: Proceedings of the 14th International Symposium on Robotics Research, Springer, Berlin, pp 1–14

Dijkstra E (1959) A note on two problems in connexion with graphs. Numerische Mathematik 1(1):269–271

Dorigo M, Birattari M (2007) Swarm intelligence. Scholarpedia 2(9):1462

Durrant-Whyte H, Bailey T (2006) Simultaneous localization and mapping (SLAM): part I. IEEE Robotics & Automation Magazine 13(2):99–110

Filliat D, Meyer J (2003) Map-based navigation in mobile robots: I. a review of localization strategies. Cognitive Systems Research 4(4):243–282

Flint M, Polycarpou M, Fernandez-Gaucherand E (2002) Cooperative control for multiple autonomous UAV's searching for targets. In: Proceedings of the 41st Conference on Decision and Control, IEEE Press, Piscataway, vol 3, pp 2823–2828

Fowers SG, Lee DJ, Tippetts BJ, Lillywhite KD, Dennis AW, Archibald JK (2007) Vision aided stabilization and the development of a quad-rotor micro UAV. In: Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation, IEEE Press, Piscataway, pp 143–148

Gage D (1992) Command control for many-robot systems. Unmanned System Magazine 10(4):28–34

Grzonka S, Grisetti G, Burgard W (2009) Towards a navigation system for autonomous indoor flying. In: Proceedings of the International Conference on Robotics and Automation, IEEE, Piscataway, pp 2878–2883

Gurdan D, Stumpf J, Achtelik M, Doth KM, Hirzinger G, Rus D (2007) Energy-efficient autonomous four-rotor flying robot controlled at 1 kHz. In: Proceedings of the International Conference on Robotics and Automation, ICRA '07, IEEE Press, Piscataway, pp 361–366

Heo N, Varshney PK (2005) Energy-efficient deployment of intelligent mobile sensor networks. IEEE Transactions on Systems, Man and Cybernetics, Part A 35(1):78–92

Hoffmann G, Rajnarayan D, Waslander S, Dostal D, Jang J, Tomlin C (2004) The stanford testbed of autonomous rotorcraft for multi agent control (STARMAC). In: Proceedings of the 23rd Digital Avionics Systems Conference, DASC '04., IEEE, Press, Piscataway, vol 2, pp 1–10

Hoffmann G, Huang H, Waslander S, Tomlin C (2007) Quadrotor helicopter flight dynamics and control: Theory and experiment. In: Proceedings of the Guidance, Navigation, and Control Conference, AIAA, Reston, pp 1–20

Howard A, Mataric MJ, Sukhatme GS (2002a) An incremental self-deployment algorithm for mobile sensor networks. Autonomous Robots 13(2):113–126

Howard A, Mataric MJ, Sukhatme GS (2002b) Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In: Proceedings of the 6th Distributed Autonomous Robotic Systems, Springer, Berlin, vol 5, pp 299–308

Iida F (2003) Biologically inspired visual odometer for navigation of a flying robot. Robotics and Autonomous Systems 44:201–208

Khatib O (1985) Real-time obstacle avoidance for manipulators and mobile robots. In: Proceedings of the International Conference on Robotics and Automation, IEEE Press, Piscataway, vol 2, pp 500–505
Li Q, Rosa MD, Rus D (2003) Distributed algorithms for guiding navigation across a sensor network. In: Proceedings of the 9th annual international conference on Mobile computing and networking, MobiCom '03, ACM, New York, pp 313–325
Liu B, Brass P, Dousse O, Nain P, Towsley D (2005) Mobility improves coverage of sensor networks. In: Proceedings of the 6th international symposium on Mobile ad hoc networking and computing, ACM, New York, pp 300–308
Mamei M, Zambonelli F (2005) Physical deployment of digital pheromones through RFID technology. In: Proceedings of the Swarm Intelligence Symposium, SIS '05, IEEE Press, Piscataway, pp 281–288
McLurkin J, Smith J (2007) Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. In: Alami R, Chatila R, Asama H (eds) Distributed Autonomous Robotic Systems 6, Berlin, pp 399–408
Mei Y, Lu YH, Hu YC, Lee CSG (2005) A case study of mobile robot's energy consumption and conservation techniques. In: Proceedings of the 12th International Conference on Advanced Robotics, ICAR '05, IEEE Press, Piscataway, pp 492–497
Melhuish C, Welsby J (2002) Gradient ascent with a group of minimalist real robots: Implementing secondary swarming. In: Proceedings of the International Conference on Systems, Man and Cybernetics, IEEE, Press, Piscataway, vol 2, pp 509–514
Meyer JA, Filliat D (2003) Map-based navigation in mobile robots: II. a review of map-learning and path-planning strategies. Cognitive Systems Research 4(4):283–317
Nakamura T, Ohara M, Ogasawara T, Ishiguro H (2003) Fast self-localization method for mobile robots using multiple omnidirectional vision sensors. Machine Vision Applications 14(2):129–138
Nardi RD, Holland O, Woods J, Clark A (2006) SwarMAV: A swarm of miniature aerial vehicles. In: Proceedings of the 21st International UAV Systems Conference
Nouyan S, Campo A, Dorigo M (2008) Path formation in a robot swarm. Swarm Intelligence 2(1):1–23
Nouyan S, Groß R, Bonani M, Mondada F, Dorigo M (2009) Teamwork in self-organized robot colonies. Transactions on Evolutionary Computation 13(4):695–711
Oh PY, Joyce M, Gallagher J (2005) Designing an aerial robot for hover-and-stare surveillance. In: Proceedings of the 12th International Conference on Advanced Robotics, IEEE Press, Piscataway, pp 303–308
O'Hara KJ, Balch TR (2004) Distributed path planning for robots in dynamic environments using a pervasive embedded network. In: Proceedings of the 3rd International Conference on Autonomous Agents and Multi-Agent Systems, IEEE Press, Piscataway, vol 3, pp 1538–1539
Payton D, Daily M, Estowski R, Howard M, Lee C (2001) Pheromone robotics. Autonomous Robots 11(3):319–324
Payton D, Estkowski R, Howard M (2004) Pheromone robotics and the logic of virtual pheromones. In: Şahin E, Spears W (eds) Swarm Robotics: SAB 2004 international workshop, Springer, Berlin, LNCS, vol 3342, pp 45–57
Perkins CE, Royer EM (1999) Ad-hoc on-demand distance vector routing. In: Proceedings of the 2nd Workshop on Mobile Computing Systems and Applications, WMCSA '99, IEEE Press, Piscataway, pp 90–100
Pezeshkian N, Nguyen H, Burmeister A (2007) Unmanned ground vehicle radio relay deployment system for non-line-of-sight operations. In: 13th IASTED International Conference on Robotics and Applications, ACTA Press, Calgary
Poduri S, Sukhatme GS (2004) Constrained coverage for mobile sensor networks. In: Proceedings of the International Conference on Robotics and Automation, IEEE Press, Piscataway, vol 1, pp 165–171
Pugh J, Raemy X, Favre C, Falconi R, Martinoli A (2009) A fast on-board relative positioning module for multi-robot systems. IEEE Transactions on Mechatronics, Focused Section on Mechatronics in Multi-Robot Systems 14(2):151–162
Reif J, Wang H (1999) Social potential fields: A distributed behavioral control for autonomous robots. Robotics and Autonomous Systems 27(3):171–194
Roberts J, Stirling T, Zufferey JC, Floreano D (2007) Quadrotor using minimal sensing for autonomous indoor flight. In: European Micro Air Vehicle Conference and Flight Competition, EMAV '07
Roberts J, Zufferey JC, Floreano D (2008) Energy management for indoor hovering robots. In: Proceedings of the International Conference on Intelligent Robots and Systems, IEEE Press, Piscataway, pp 1242–1247
Roberts J, Stirling T, Zufferey JC, Floreano D (2009) 2.5D infrared range and bearing system for collective robotics. In: Proceedings of the International Conference on Intelligent Robots and Systems, IROS '09, IEEE Press, Piscataway, pp 3659–3664
Rudol P, Wzorek M, Conte G, Doherty P (2008) Micro unmanned aerial vehicle visual servoing for cooperative indoor exploration. In: Proceedings of the Aerospace Conference, IEEE Press, Piscataway, pp 1–10

Sharma R, Ghose D (2007) Swarm intelligence based collision avoidance between realistically modelled UAV clusters. In: Proceedings of the American Control Conference, IEEE Press, Piscataway, pp 3892–3897

Siegwart R, Nourbakhsh IR (2004) Introduction to Autonomous Mobile Robots. MIT Press, Cambridge

Stipes J, Hawthorne R, Scheidt D, Pacifico D (2006) Cooperative localization and mapping. In: Proceedings of the 2006 International Conference on Networking, Sensing and Control, IEEE, Piscataway, pp 596–601

Unver O, Uneri A, Aydemir A, Sitti M (2006) Geckobot: A gecko inspired climbing robot using elastomer adhesives. In: Proceedings of the International Conference on Robotics and Automation, ICRA '06, IEEE Press, Piscataway, pp 2329–2335

Valenti M, Bethke B, How JP, Farias DP, Vian J (2007) Embedding health management into mission tasking for UAV teams. In: American Control Conference, IEEE Press, Piscataway, pp 5777–5783

Wang G, Cao G, Porta TL, Zhang W (2005) Sensor relocation in mobile sensor networks. In: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM '05, IEEE Press, Piscataway, vol 4, pp 2302–2312

York GW, Pack DJ (2008) Cooperative persistent surveillance search algorithms using multiple unmanned aerial vehicles. In: Grundel D, Murphey R, Pardalos P, Prokopyev O (eds) Cooperative Networks: Control and Optimization, Edward Elgar Publishing, Cheltenham, pp 279–290

Zavlanos MM, Pappas GJ (2007) Potential fields for maintaining connectivity of mobile networks. IEEE Transactions on Robotics 23(4):812–816

Zeimpekis V, Giaglis GM, Lekakos G (2003) A taxonomy of indoor and outdoor positioning techniques for mobile location services. SIGecom Exchange 3(4):19–27

Ziparo VA, Kleiner A, Nebel B, Nardi D (2007) RFID-based exploration for large robot teams. In: Proceedings of the International Conference on Robotics and Automation, ICRA '07, IEEE Press, Piscataway, pp 4606–4613

Zou Y, Chakrabarty K (2003) Sensor deployment and target localization based on virtual forces. In: Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM '03, IEEE Press, Piscataway, vol 2, pp 1293–1303