Energy-Time Efficiency in Aerial Swarm Deployment

Timothy Stirling and Dario Floreano

Abstract A major challenge in swarm robotics is efficiently deploying robots into unknown environments, minimising energy and time costs. This is especially important with small aerial robots which have extremely limited flight autonomy. This paper compares three deployment strategies characterised by nominal computation, memory, communication and sensing requirements, and hence are suitable for flying robots. Energy consumption is decreased by reducing unnecessary flight following two premises: 1) exploiting environmental information gathered by the robots; 2) avoiding diminishing returns and reducing interference between robots. Using a 3-D dynamics simulator we examine energy and time metrics, and also scalability effects. Results indicate that a novel strategy that controls the density of flying robots is most promising in reducing swarm energy costs while maintaining rapid search times. Furthermore, we highlight the energy-time tradeoff and the importance of measuring both metrics, and also the significance of electronics power in calculating total energy consumption, even if it is small relative to locomotion power.

1 Introduction

Autonomous systems must manage their own energy resources to complete missions successfully [15]. This is notably evident with small aerial robots, which have severely limited flight autonomy (typically 10–15 minutes [19, 23]). Although energy efficient algorithms are paramount in creating truly autonomous aerial robots, prior research is sparse [23]. Previously we developed an algorithm for indoor aerial swarm search [22] that exploited the ability of our robots to attach to ceilings, saving energy [19]. This paper expands this work and compares methods to deploy

Timothy Stirling and Dario Floreano
Laboratory of Intelligent Systems (LIS), Ecole Polytechnique Fédéral de Lausanne (EPFL), Switzerland e-mail: `tim.stirling@epfl.ch`

a swarm of aerial robots into unknown environments, aiming to reduce the total swarm energy cost with rapid operation for a search task.

A complex problem in swarm robotics is controlling deployment into unknown environments. If robots deploy to unnecessary locations, energy is wasted. Furthermore, they may interfere with other robots, e.g. by increasing collision risk [20]. Conversely, if an area receives insufficient robots the task may be unachievable or performance reduced. Rapid deployment is desired to expedite tasks such as disaster mitigation. However, time and energy are not independent, and often there is a tradeoff [13, 6, 10]. Previous research considered only ground robots. However, aerial robots have significantly different energy dynamics, require substantially more energy to locomote [19], and the small payload entails reduced sensing and processing capabilities. Time and energy were previously either examined independently, or only with multi-objective functions that mask trends in the individual metrics. Prior work also usually neglected the energy consumption of sensors and processors [12].

This paper compares three strategies suitable for aerial swarms, characterised by minimal computation, communication and sensing requirements. They are suitable for microcontrollers rather than powerful CPUs and are simple to implement to avoid further complicating autonomous flight control. Total swarm energy and search time are examined in 3-D simulation using a complete energy model validated on real robots [19]. Finally, scalability performance is examined by increasing the robot group size.

2 Related Work

In work by Rybski *et al.* [21], increasing the number of deployed robots led to the phenomenon of *diminishing returns*, as proposed by economists [1]. Additional robots increased performance by decreasing amounts until a peak was reached, after which additional robots no longer improved performance. Moreover, Rosenfeld *et al.* [20] examined scalability in foraging tasks and noted that after a peak in performance, additional robots usually decreased performance (*negative returns*) due to spatial constraints and interference. Spatial constraints are stronger in confined areas, such as narrow corridors, causing congestion and increased collision risk. Therefore, it is important to control deployment to minimise time and energy costs.

The tradeoff between group size and efficiency was examined by Hayes [6] in a search task. A multi-objective performance function was used incorporating search time, energy and robot initialisation costs. This analysis allowed the prediction of the optimal number of robots to complete the search. However, Hayes' analysis assumed an obstacle-free square arena and ignored spatial constraints and interference. Similarly, Mei *et al.* [11] researched methods to determine the optimal group size under constraints of energy, time and environment area. It was shown that energy limitations significantly affected the required group size. However, the environment size was known *a priori* and a centralised planner used.

For operation in unknown environments, Howard *et al.* [7] developed an algorithm that deployed robots one at a time, thereby avoiding interference. Each subsequent deployed robot exploited environment information acquired from previously deployed robots and was guided to optimal environment locations. However, Howard's approach is slow [7]. Alternatively, Zlot *et al.* [25] present a market economy-based architecture [5] for efficient multi-robot exploration. This maximises search area while minimising total travel distance. However, high bandwidth communication is required. Both Howard's and Zlot's approaches are computationally expensive with centralised processing, usually undesirable in swarm robotics. Unfortunately, neither authors provide quantitative results for energy or time costs.

For simple robots with decentralised control, Chang *et al.* [2] deployed robots based on perceived local environment size, reducing unnecessary locomotion. When larger areas are discovered, additional robots are requested to aid exploration, reducing search time. However, this was assessed with ground robots in a simple discrete 2-D simulator with basic environments that reduced spatial constraints and interference. Additionally, the deposition of artificial pheromones was used to control deployment, but no such sensor currently exists for real flying robots.

Alternatively, researchers have developed mechanisms to improve efficiency by controlling robot activation. For example, Liu *et al.* [9] examined energy efficient task allocation in foraging robots. The ratio of active foragers to resting robots was adjusted based on simple adaptation rules. These rules included internal cues of successful foraging, environmental cues from collisions, and social cues of successful foraging by other robots. However, foraging differs from search because (un)successful foraging over time indicates the robot's utility, which can be used to control activity. Furthermore, foraging often involves retrieval to a communal nest, facilitating global coordination through local communication. Finally, a basic energy model was used and time costs were not examined.

In summary, previous research focused only on ground robots, but aerial robots have considerably different energy characteristics [19]. We compare three strategies that are scalable, decentralised, require no *a priori* environment information, and are suitable for aerial robots with nominal computation, sensing and communication requirements.

3 Aerial Swarm Search Without Global Information

The aerial swarm search algorithm considered here [22] is based on principles of sensor networks, which perform distributed processing of local information through wireless communication [7]. This work is based on the quadrotor robots we are developing for indoor swarming [17]. The robots are equipped with infrared distance sensors for obstacle detection [17]. A 3-D relative-positioning sensor gives the range and bearing to nearby robots and low-bandwidth short range communication, facilitating coordination [18]. Wireless LAN provides longer range communication. To prolong missions the robots can attach to ferromagnetic ceilings [19]. Alternatively, Gecko inspired dry adhesives [14] or mechanical perching could be utilised [8, 4], or simply robots could land (merely loosing their elevated perception capabilities).

Robots operate in two control states: "*Beacons*" or "*Explorers*". Beacons are static robots passively attached to the ceiling to conserve energy and form a robotic sensor network [19]. Explorers are flying robots, deploying into the environment guided by Beacons. Beacons sense their local environment and communicate with neighbouring Beacons to guide nearby Explorers. Explorers start clustered on the ground below a pre-deployed Beacon. When deployed, Explorers take off and follow the guidance signal of the nearest Beacon, flying from Beacon to Beacon across the network. Beacons next to unexplored space indicate adjacent locations where a new Beacon is required. Explorers that arrive at these locations attach to the ceiling and become Beacons. Beacons can revert back to Explorers once an area has been searched and redeploy to unexplored areas. We utilise depth-first search [3] which exhaustively explores a subarea of the environment before searching other unexplored areas. This avoids search duplication and unnecessary locomotion compared with stochastic methods, thus reducing the total swarm flight time [22]. Navigation in unknown environments is afforded by the hop-counts of local communication signals propagated across the network [22]. By exploiting the ceiling attachment capability, the swarm energy cost is reduced by 3–400% [22]. A video demonstrating the search behaviour in simulation is available online[1]. A second video demonstrating the current progress in developing this search strategy on real robots is also available[2], which shows entirely autonomous flight.

4 Deployment Strategies

To reduce energy costs and search time, the initial deployment of robots from the ground and the redeployment of Beacons from the ceiling once an area is searched are controlled. Three strategies are compared that are scalable, decentralised, and require low computational and communication resources. The strategies exploit environment information as it is acquired by the robots to reduce unnecessary locomotion [7, 2], and reduce diminishing returns and interference between robots [20, 9]. A video showing the three deployment strategies is available online[3].

4.1 Linear-Temporal Incremental Deployment (LTID)

The simplest strategy, labelled LTID, deploys robots one at a time with a fixed time interval between consecutive launches. This was used in our prior work [22] and is

[1] `http://lis.epfl.ch/~stirling/videos/Swarm_Search.avi`

[2] `http://lis.epfl.ch/~stirling/videos/Eyebot_Autonomous_Flight.mp4`

[3] `http://lis.epfl.ch/~stirling/videos/Deployment_Strategies.mp4`

similar to the linear dispatching presented by Chang *et al.* [2]. Longer inter-launch intervals (λ) slow deployment, but decrease the number of concurrent flying robots. This reduces spatial interference and unnecessary flight by exploiting environmental information acquired from the expanding Beacon network. Once a subarea of the environment has been searched, Beacons redeploy as Explorers to new unexplored areas. Before this redeployment commences, there may be multiple Explorers flying into this subarea where they are not required, which is reduced with longer inter-launch intervals. Thus, LTID reduces energy consumption by reducing interference and unnecessary locomotion.

To implement LTID, robots are assigned a unique ID $\{1,...,N\}$ and initially launch after $\lambda \times$ID seconds. Redeploying beacons also wait λ before detaching. LTID does not adapt online, but λ could be optimised *a priori* if the type of environment is known [22]. The advantages of LTID are its simplicity and no requirements for sensing, communication or significant processing. Therefore, LTID serves as a baseline strategy from which the other two strategies can be compared.

4.2 Single Incremental Deployment (SID)

SID is similar to LTID and deploys one robot at a time, but waits for the previous robot to become a Beacon before launching the next. This is similar to Howard *et al.*'s [7] approach in that the swarm waits for the previous robot to examine the newly discovered environment, but no centralised processing or map is required. SID reduces unnecessary flight time because the next robot will only (re)deploy once the Beacon network has sensed the environment and perceived if and where a new Beacon is required. Thereby, Explorers always fly directly to the desired deployment location. To implement SID, the swarm communicates if an Explorer is flying. This can be achieved by propagating local messages across the Beacon network. However, here we employ a simplified mechanism using long-range wireless communication. Beacons signal to the whole swarm if they perceive a flying Explorer and robots only (re)deploy if no signal is received. To ensure only a single robot deploys at a time, random timeouts are used. When no flying Explorer signal is present, robots wait a short random time period (typically 1–2 s). If after this period there is no flying Explorer signal, the robot can deploy.

Robots usually deploy more slowly than with LTID, increasing search times. Therefore, although SID may reduce flight energy consumption compared with LTID, the energy consumption of sensors and processors may be elevated due to the increased runtime. Additionally, there may be a robot that is closer to the desired destination due to the redeployment of Beacons, but since the launch selection is based on random timeouts, the closest robot is not guaranteed to deploy. Various strategies exist that would ensure the closest robot is selected [24]. SID is a fixed deployment scheme without adjustable parameters. SID requires no additional sensing or computation, but very low bandwidth communication is required for coordination.

4.3 Adaptive Group Size (AGS)

AGS is a novel strategy that adapts the density of flying robots, inspired by Liu *et al.*'s [9] rules to control robot foraging activity. However, we consider a search task rather than a foraging task (see Sect. 2), and we aim to explicitly avoid collisions. AGS initially rapidly deploys robots, every 2–3 seconds. Flying Explorers measure the density of neighbouring flying robots using their relative-positioning sensor [18] and will probabilistically land if the density is higher than a predefined threshold. This decreases the ratio of flying robots, reducing diminishing returns and interference. Robots which have landed launch again when there are no robots flying in the vicinity. The density of flying robots ρ is given by: $\rho = \sum_{i=1}^{N} \frac{4}{d_i}$, where d_i is the distance to neighbouring robot i. The constant 4 is a normalisation factor such that a single flying robot 4.0 m away (considered a safe flight separation) gives ρ the unit value 1. If ρ is greater than a threshold τ (typically 3.0–8.0) the robot will try to land. To prevent multiple robots landing simultaneously, robots wait a random timeout period (typically 1–2 s) while they signal their intention to land. After this timeout robots can land if no signal of a neighbouring robot's intention to land is received. Otherwise, the robot with the highest ID has priority in landing. Robots could attach to the ceiling [19] instead of landing, but this could interfere with the Beacon network.

AGS uses the perceived density of flying robots to avoid diminishing returns. For example, if robots land in high density areas where collision risk is considerable, interference is reduced. Furthermore, the perceived density implicitly encodes local environment information. If many robots are flying in a confined space the density will be high. Avoiding high densities reduces the deployment of robots to locations where they may not be required. Therefore, energy consumption is reduced by decreasing unnecessary flight time and interference. AGS can be optimised by varying the threshold τ. No significant processing or high bandwidth communication is required. However, a sensor is required to measure the density, which could be simple Time of Flight sensors or local communication instead of relative-positioning [18].

5 Experimental Method

Comparing strategies requires extensive simulation analysis since it is infeasible to gather sufficient data for statistical analysis with real flight tests given the large parameter space and the challenging logistics of conducting numerous flight experiments. Therefore, a realistic 3-D dynamics simulator was utilised [16], as discussed below in Sect. 5.1

Performance was measured over 100 trials with robots clustered in random starting locations in randomly generated maze-like corridor environments. Environments were constructed from 40 connected 3×3 m cells (see [22] for details). Fig. 1 shows a typical environment and the swarm deploying with Beacons and a flying Explorer. For the first experiments 20 robots were available to deploy. Subsequent experiments

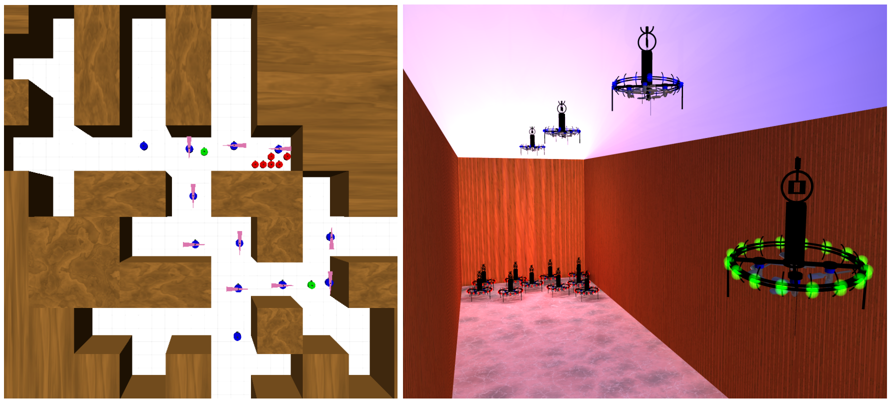


Fig. 1 *Left*: Typical randomly generated maze environment. *Right*: The swarm deploying with robots on the ground, a flying Explorer and Beacons on the ceiling

assessed the scalability performance, so the number of robots was increased from 20 to 30. We measured search time, coverage area and swarm energy consumption, calculated with an energy model of the rotor thrust-power curve of a real quadrotor helicopter [19], shown in Fig. 2. This model facilitated the accurate prediction of flight endurance within a 1% error. The model was extended to include the energy used by sensors and processors, detailed in Table 1. This creates three electronics power consumption rates: when the robot was flying (high power); when a Beacon on the ceiling (medium power); and when resting on the ground (low power). This equates to a power consumption of 120 W for flying Explorers, 5 W for Beacons, and 0.5 W for robots resting on the ground. These rates were validated on real flying robots [19] and are similar to other rotorcraft, e.g. [23]. Moreover, the performance trends are robust to changes in model parameters since the power rates are differentiated by an order of magnitude.

Table 1 Power consumption of components used to develop the energy model for aerial robots

Component	Power (W)	Comment
Total rotor power	100–120	Depends on payload [19]
Flight computer	2.44	Sensors & microprocessor [19]
Microprocessor	0.125	Microchip PIC32 40MHz
802.11a WiFi	1.5, 1.22, 0.01	Send, receive & sleep power [15]
Infrared distance sensor	0.165	Sharp GP2Y0A02YK
Ultrasonic altitude	0.015	MaxBotics LV-MaxSonar-EZ4
3-D Relative positioning	~ 7	For 20 robots [18]

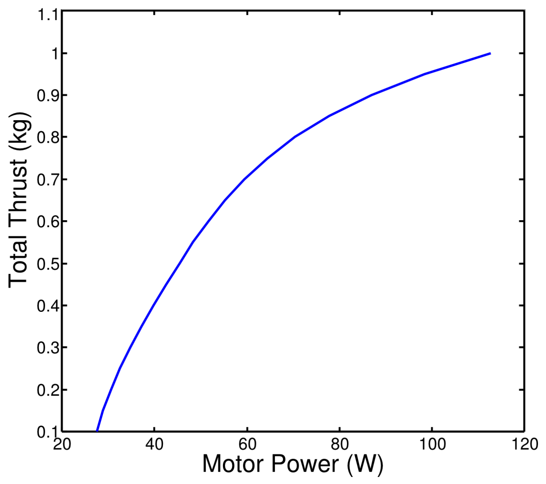


Fig. 2 Thrust-power curve of the quadrotor propulsion system validated in [19]

5.1 Simulation and Flight Dynamics

A custom 3-D dynamics simulation was developed using the Open Dynamics Engine[4] (ODE). An input force vector F_{tot} is applied to a rigid body with mass m giving accelerations: $\dot{V} = \frac{1}{m}F_{tot}$. Angular accelerations and torque were neglected since they are stabilised by the flight controller [17]. Gaussian noise was added to simulate turbulence and imprecise control, with standard deviations (s.d.) set empirically (2.5×10^{-2} N) as the platform has not yet been characterised. However, the altitude fluctuation was modelled from previous work in [17]. F_{tot} is given by:

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

where ε_N is a Gaussian noise vector for roll, pitch and thrust standard deviations: $\varepsilon_N \sim \mathscr{N}(0,\hat{\sigma})$. F_g denotes the platform weight with $F_g = [0,0,m\cdot g]^T$. F_c is the control force vector formed from desired pitch f_p and roll f_p forces, combined with the altitude control f_a from a PID controller [17]: $F_c = [f_p, f_r, f_a]^T$. Drag force is calculated with:

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

where ρ is the specific air-density, V is air-speed, A is the frontal reference area and C_d is the estimated drag coefficient.

Sensor noise was Gaussian with s.d. measured from characterisation experiments: 2.5 cm for the ultrasound altitude sensor and 5 cm for infrared distance sensors used for obstacle avoidance. The relative-positioning sensor has been characterised in [18]: the range s.d. is 17 cm and bearing s.d. is 6.1°.

[4] `www.ode.org`

6 Results

To compare performances both the total swarm energy consumption and search time metrics are examined separately. Subsequently, a multi-objective function is used that linearly combines energy and time into a simple single parameter-free metric, the Energy-Time-Product (ETP), measured in Joule-Seconds (Js) [2]. The ETP is inspired by the Power-Delay Product frequently used in electronics engineering. Both energy and time metrics are taken to have equal unit weighting, which assumes the equal importance of these factors and also avoids arbitrary parameterisation. Alternative weighting of parameters is discussed in Sect. 7. With AGS and LTID we varied the inter-launch interval with $\lambda = 6$, 8, 10, 12, 18 and 24 seconds, and the density-threshold τ from 3.0 to 8.0, respectively. Finally, scalability performance is examined by increasing the robot group size. Shapiro-Wilk tests indicated small deviations from normal-distributions, so Kruskal-Wallace χ^2 and Spearman's Rho r_s non-parametric tests were used to examine the statistical significance of any effects. Medians are shown with standard deviations in parenthesis.

6.1 Overall Comparisons

Importantly, there was no significant difference in median coverage area (99.4%) across all strategies ($\chi^2 = 18.42$, df = 12, $p = 0.1$), permitting fair comparisons. The mean coverage area was 95.7%. Comparing all strategies over all parameters, the fastest was AGS with $\tau = 6.0$ with a median search time of 307.9 s (57.6) (Fig. 3(a)). The most energy efficient was AGS with $\tau = 4.0$, with a median of 178.5 kJ (33.8) (Fig. 3(b)). The slowest strategy was SID taking 1013.9 s (118.6), 229.3% slower than AGS with $\tau = 6.0$. The least energy efficient was LTID with $\lambda = 6$ with a median of 253.4 kJ (59.2), requiring 42.0% more energy than AGS with $\tau = 4.0$. Comparing ETP performances (Fig. 3(c)), LTID suffers from a tradeoff between search time and energy-efficiency achieving its lowest ETP of 800.5×10^5 Js (26.5) with $\lambda = 10$ s. Since AGS showed low energy consumption with fast search times it produced the lowest overall ETP of 579.7×10^5 Js (183.1), with $\tau = 6.0$. Finally, because SID suffers from slow deployment the median ETP was high, 1866.3×10^5 Js (459.8).

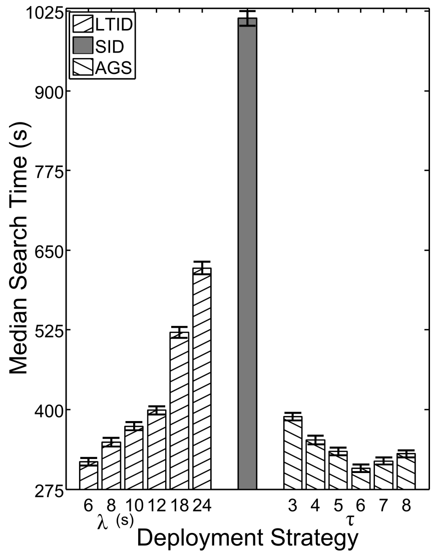
With LTID, increasing the time between consecutive robot launches (λ) significantly decreased the median swarm energy consumption ($r_s = -0.47$, df = 598, $p < 0.001$) and increased median search time ($r_s = 0.80$, df = 598, $p < 0.001$). Increasing λ from 6 s to 24 s reduces the median energy consumption by 27.6% and increases median search time by 95.6%. Energy consumption is deceased by reducing robot deployment to unnecessary locations and decreasing interference, decreasing flight energy. Although energy consumption decreased as λ increased, the search time increased more strongly, so the median ETP increased.

SID has no tunable parameters. SID had the slowest search time since only a single robot flies at a time. Robots deployed only when and where necessary, which minimised unnecessary flight energy over all deployment strategies, confirmed with multiple comparisons at the $p < 0.001$ level (using Wilcoxon ranksum tests). However, SID did not achieve the lowest energy consumption because of the energy consumption of the Beacons' sensors and processors over the long search duration. Therefore, even although the power consumption of electronics is small compared with the rotor power it is important to consider within a complete energy model. The median ETP was high due to the slow search time.

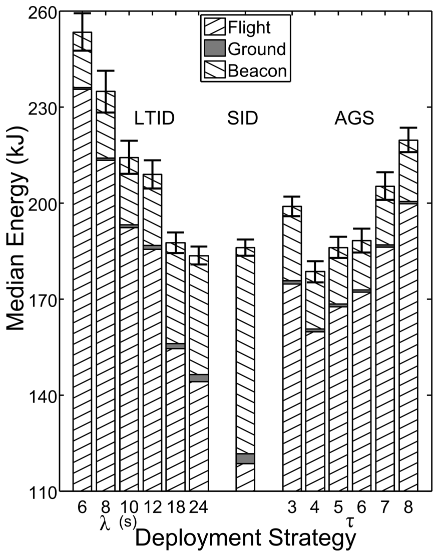
With AGS, varying the threshold τ significantly affected the median search time ($\chi^2 = 116.3$, df = 5, $p < 0.001$) and median swarm energy consumption ($\chi^2 = 102.1$, df = 5, $p < 0.001$). Both energy and time metrics form a $\cup$-shape. This is because at low thresholds flying robots have a higher probability of landing when encountering neighbours, which incurs a time and energy cost. Conversely, increasing τ reduces the effect of controlling the group size and so increases interference and unnecessary flight, thereby increasing energy consumption and search time. Therefore, there is an optimal threshold ($\tau = 6.0$) that minimises the ETP.

6.2 Scalability Performance

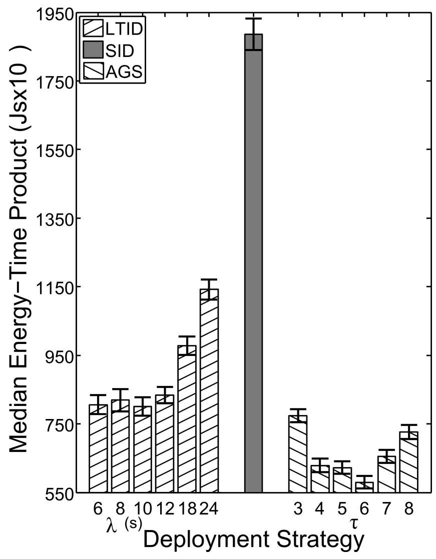
To assess scalability performance the median ETP was compared when the number of robots was increased from 20 to 30. For LTID and AGS the parameters that minimised the ETP for 20 robots (λ = 10 s and τ = 6.0, respectively) were the same for 26 and 30 robots, so were used for all group sizes. Since increasing the swarm size can increase the expected coverage area [22] and associated search time and flight energy, we restricted results to trials that achieved 100% coverage, ensuring fair comparisons. Results are shown in Fig. 3(d). Increasing the robot group size significantly increases the median ETP for both AGS ($r_s = 0.28$, df = 598, $p < 0.001$) and LTID ($r_s = 0.41$, df = 598, $p < 0.001$), but not for SID ($r_s = 0.007$, df = 598, $p > 0.86$). LTID increases at a higher rate compared to AGS. AGS minimises the median ETP over all tested group sizes. The median ETP of SID is approximately constant because only a single robot flies at a time, so there is no unnecessary flight time. However, SID never becomes competitive even for large group sizes. The ETP trends for both AGS and LTID mask a decrease in median search time (9.6% and 9.1%, respectively) and an increase in median energy cost (15.6% and 29.4%, respectively). This is due to the increased parallelisation afforded by additional robots accelerating the search and consequently increasing flight energy. Importantly, for all strategies the median ETP (and energy consumption) *per robot* decreases as the group size increases, indicating good scalability performance.



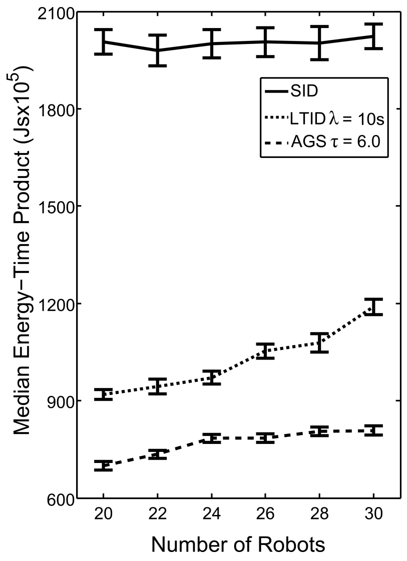
(a) Median Search Time



(b) Median Energy



(c) Median ETP



(d) ETP Scalability

Fig. 3 Median **a**) search time, **b**) swarm energy cost and **c**) Energy-Time-Product (ETP) tested with 20 robots over 100 trials. Standard error bars (standard deviation divide by square root of sample size) are shown. The energy results show the constituent ground, beacon and flight costs. The inter-launch interval λ of LTID and the flying robot density-threshold τ of AGS were varied. **d**) Median ETP performance as the group size increases. AGS has the lowest energy consumption, fastest search time, lowest ETP and good scalability performance

7 Conclusion and Future Work

In this paper we compared three strategies to deploy flying robots for a search task in unknown environments using a complete energy model for electronics and motors validated on real flying robots. All strategies were characterised by nominal computation, memory and communication requirements, and were necessarily simple to facilitate implementation on aerial systems without further complexifying autonomous flight control. To summarise:

- LTID demonstrated that slowing deployment facilitates a significant reduction in energy consumption up to 27.6%, but this increased search time by 95.6%. This tradeoff can be optimised by adjusting the inter-launch interval. No communication or additional sensing is required and the implementation is simple. Therefore, LTID serves as a useful benchmark strategy.
- SID ensures only one robot flies at a time and leads to low energy consumption, but a very high search time. This indicates that mitigating deployment of robots to unnecessary locations by exploiting acquired environmental information significantly reduces flight energy. However, the increased energy consumption of sensors and processors prevents SID achieving the best overall energy-efficiency. No additional sensing is required, but very low-bandwidth communication is used for coordination.
- The AGS strategy results showed that, by controlling the density of flying robots, the swarm energy consumption can be reduced while also achieving rapid search. AGS require a sensor to measure the local robot density.

With optimal parameters, AGS ($\tau = 6.0$) has an ETP 27.6% better than LTID ($\lambda = 10\,\mathrm{s}$) and 69.3% better than SID. All strategies showed good scalability performance, with a decreased median ETP *per robot* as group sizes increased; SID displayed constant performance, but AGS consistently achieved the best ETP.

When comparing algorithm performance, the choice of metrics is crucial. Previous researchers often examined either the time cost [20] or energy consumption [9, 7] independently. Alternatively, multi-objective functions are used, combining multiple weighted metrics, e.g. Hayes [6]. However, this is not straightforward due to the choice of metrics, weighting and formulation. Additionally, although comparisons are simplified, individual metric trends are obfuscated. To clearly understand the underlying trends, we have shown both energy and time costs independently as well as the ETP to allow selection of the best energy-time efficient strategy. The ETP has equal weighting of time and energy costs providing a simple metric. However, relative weightings could be easily applied to the provided energy and time results, depending on their relative importance in different applications. The ETP facilitated comparisons between different robot group sizes during the scalability tests.

This work was confined to one type of corridor environment of a fixed size. Properties such as size and complexity or the existence of open areas may affect the performance of the three strategies and response to parameters (λ and τ). These effects were examined with LTID previously in [22]. Summarising, the gains in energy efficiency with LTID are more pronounced in higher complexity environments with more corridor junctions. This is because robots will redeploy to new areas more frequently and experience greater interference. Therefore, it is expected that these characteristics will generalise to SID and AGS. Such complex environments are common in buildings such as offices, especially in disaster situations. It would also be feasible to autonomously optimise the control parameters based on perceived environmental conditions, a subject of future work. Finally, the effects of small obstructions (e.g., lights) on the relative-positioning sensor was neglected. Current testing indicates the sensor is robust to small obstructions and experiences only slight attenuation, while large obstacles are handled algorithmically [22].

Currently we have developed the autonomous flight behaviours of the underlying swarm search behaviour, validating the feasibility of the approach. In the future we aim to verify the presented results with real flying robots. We are also investigating methods to extend these strategies to further reduce flight energy, e.g. by selection of the closest robot to the desired destination with strategies amenable to swarm robotics [24]. Additionally we will test all strategies in more varied environments aiming to draw more general performance predictions.

In conclusion, aerial swarms are gaining interest due to their suitability for many applications such as search or disaster mitigation because they can rapidly cover obstacle-rich terrain [22]. The work presented here facilitates the future deployment of flying robots with limited autonomy to successfully cover larger environments, while understanding the impact on time costs. The three presented strategies provide different performances with different sensing and communication requirements, facilitating selection according to robot capabilities and application requirements.

Acknowledgements This work is part of the "Swarmanoid Project", a Future Emerging Technologies (FET IST-022888) project funded by the European Commission. The authors would like to thank Sara Mitri and Thomas Schaffter for their valuable feedback and assistance.

References

1. Brue, S.L.: Retrospectives: The law of diminishing returns. The Journal of Economic Perspectives **7**(3), 185–192 (1993)
2. Chang, H.J., Lee, C.S.G., Y.-H., L., Hu, Y.C.: Energy-time-efficient adaptive dispatching algorithms for ant-like robot systems. In: International Conference on Robotics and Automation, ICRA '04, vol. 4, pp. 3294–3299. IEEE, Piscataway (2004)
3. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. MIT Press, Cambridge (1990)
4. Cory, R., Tedrake, R.: Experiments in fixed-wing UAV perching. In: Proceedings of the Guidance, Navigation, and Control Conference, pp. 1–12. AIAA, Reston (2008)
5. Dias, M., Zlot, R., Kalra, N., Stentz, A.: Market-based multirobot coordination: A survey and analysis. Proceedings of the IEEE **94**(7), 1257–1270 (2006)
6. Hayes, A.T.: How many robots? Group size and efficiency in collective search tasks. In: Proceedings of the 6th Int. Symp. on Distributed Autonomous Robotic Systems, DARS '02, pp. 289–298 (2002)

7. Howard, A., Mataric, M.J., Sukhatme, G.S.: An incremental self-deployment algorithm for mobile sensor networks. Autonomous Robots **13**(2), 113–126 (2002)
8. Kovac, M., Germann, J.M., Hrzeler, C., Siegwart, R., Floreano, D.: A perching mechanism for micro aerial vehicles. Journal of Micro-Nano Mechatronics **5**(3–4), 77–91 (2009)
9. Liu, W., Winfield, A., Sa, J., Chen, J., Dou, L.: Strategies for Energy Optimisation in a Swarm of Foraging Robots, *LNCS, Swarm Robotics*, vol. 4433, pp. 14–26. Springer, Berlin (2007)
10. Mei, Y., Lu, Y.H., Hu, Y., Lee, C.: Deployment of mobile robots with energy and timing constraints. IEEE Transactions on Robotics **22**(3), 507–522 (2006)
11. Mei, Y., Lu, Y.H., Hu, Y.C., Lee, C.S.G.: Determining the fleet size of mobile robots with energy constraints. In: International Conference on Intelligent Robots and Systems, IROS '04, vol. 2, pp. 1420–1425. IEEE Press, Piscataway (2004)
12. Mei, Y., Lu, Y.H., Hu, Y.C., Lee, C.S.G.: A case study of mobile robot's energy consumption and conservation techniques. In: Proceedings of the 12th International Conference on Advanced Robotics, ICAR '05, pp. 492–497. IEEE, Piscataway (2005)
13. Moscibroda, T., von Rickenbach, P., Wattenhofer, R.: Analyzing the energy-latency trade-off during the deployment of sensor networks. In: Proceedings of the 25th International Conference on Computer Communications, pp. 1–13. IEEE Press, Piscataway (2006)
14. Murphy, M., Aksak, B., Sitti, M.: Gecko-inspired directional and controllable adhesion. Small **5**(2), 170–175 (2009)
15. O'Hara, K.J., Nathuji, R., Raj, H., Schwan, K., Balch, T.: AutoPower: toward energy-aware software systems for distributed mobile robots. In: International Conference on Robotics and Automation, ICRA '06, pp. 2757–2762. IEEE, Piscataway (2006)
16. Pinciroli, C.: The swarmanoid simulator. Tech. Rep. TR/IRIDIA/2007-025, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2007)
17. Roberts, J., Stirling, T., Zufferey, J.C., Floreano, D.: Quadrotor using minimal sensing for autonomous indoor flight. In: Proceedings of the 2007 European Micro Air Vehicle Conference and Flight Competition, EMAV '07 (2007)
18. Roberts, J., Stirling, T., Zufferey, J.C., Floreano, D.: 2.5D infrared range and bearing system for collective robotics. In: Proceedings of the International Conference on Intelligent Robots and Systems, IROS '09, pp. 3659–3664. IEEE, Piscataway (2009)
19. Roberts, J., Zufferey, J.C., Floreano, D.: Energy management for indoor hovering robots. In: Proceedings of the International Conference on Intelligent Robots and Systems, IROS '08, pp. 1242–1247. IEEE, Piscataway (2008)
20. Rosenfeld, A., Kaminka, G.A., Kraus, S.: Coordination of Large-Scale Multiagent Systems, Part 1, chap. A Study of Scalability Properties in Robotic Teams, pp. 27–51. Springer (2006)
21. Rybski, P., Larson, A., Lindahl, M., Gini, M.: Performance evaluation of multiple robots in a search and retrieval task. In: Proceedings of the Workshop on Artificial Intelligence and Manufacturing, pp. 153–160. AAAI Press, Menlo Park (1998)
22. Stirling, T., Wischmann, S., Floreano, D.: Energy-efficient indoor search by swarms of simulated flying robots without global information. Swarm Intelligence **4**(2), 117–143 (2010)
23. Valenti, M., Bethke, B., How, J.P., Farias, D.P., Vian, J.: Embedding health management into mission tasking for UAV teams. In: American Control Conference, pp. 5777–5783. IEEE, Piscataway (2007)
24. Wang, G., Cao, G., Porta, T.L., Zhang, W.: Sensor relocation in mobile sensor networks. In: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM '05, vol. 4, pp. 2302–2312. IEEE Press, Piscataway (2005)
25. Zlot, R.M., Stentz, A., Dias, M.B., Thayer, S.: Multi-robot exploration controlled by a market economy. In: International Conference on Robotics and Automation, vol. 3, pp. 3016–3023. IEEE Press, Piscataway (2002)