IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 29, 2019, accepted July 22, 2019, date of publication August 5, 2019, date of current version August 21, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2933234

Enabling Drone Services: Drone Crowdsourcing and Drone Scripting

MAJED ALWATEER[1], SENG W. LOKE[2], AND NIROSHINIE FERNANDO[2]
[1]Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC 3086, Australia
[2]School of Information Technology, Deakin University, Geelong, VIC 3216, Australia

Corresponding author : Majed Alwateer (m.alwateer@latrobe.edu.au)

ABSTRACT Drones are rapidly finding their way into civilian applications, and are mostly networked, enabling their remote programming, and connectivity with humans. However, drones are limited by the weight they can carry and battery power resulting in limited resources. Moreover, some applications require utilising multiple drones to act in coordination. The combination of utilising nearby devices (i.e. with additional resources beyond the drone capability) and controlling multiple drones in a more convenient way has the potential to overcome these limitations. This paper proposes and examines programmable crowd-powered drones, involving two key concepts for combining drones and smartphones as a crowd-powered resource cloud. In particular, we focus on crowdsourcing for drone computations, and multi-drone service management using a new scripting language for coordinated flight paths of multiple drones. We describe our underlying model and experimentation with these concepts. We then extensively discuss the prospect of drones servicing communities within IoT ecosystems, as a future direction.

INDEX TERMS Drones, smart drone services, crowdsourcing, crowd computing, drone programming.

I. INTRODUCTION

Drones or unmanned aerial vehicles (UAVs) are flying vehicles that do not have on board a human pilot. They have been commonly used by the military for warfare and to perform a range of complex tasks. These tasks required substantial control and instructions, modern ground control computers, and effective locally available navigation systems. Previously, performing such tasks has been beyond the capacities of many, but rapid advances in technology with improvements in the size, intelligence and cost, have opened the door for many drone makers to come into existence such as Parrot,[1] DJI[2] and more. As a result, this technology is now available to the public with minimal effort, and the uses of drones have grown far beyond military applications [1]. Drones are now being used to save lives [2], deliver goods and medical supplies [3], surveying [4], filming [5], rescuing [6], building structures [7], pipe inspections [8], farming [9], and much more.

Due to their mobility, drones can autonomously travel into close proximity of other flying, moving or fixed objects. They can also carry materials or smart sensors to perform physical or digital tasks. This places drones in a unique position that enables them to rapidly provide various services, as data collectors or resource providers in hard-to-reach areas. For example, collaboration between drones and other mobile devices, such as smartphones, or even other drones, can help overcome the resource limitations of traditional mobile technologies and deal with complex situations. At present, mobile computing is at a level that its future promises interesting developments where ''tens to thousands of mobile nodes can cooperate in new ways, in order to provide new capabilities and applications, from scalable context-awareness to new distributed computational platforms'' [10]. The examples for this are: mobile crowdsourcing, crowdsensing, crowd-steering, participatory social systems, mobile device clouds, and cooperative Intelligent Transport Systems.

However there are many challenges in utilising drones to effectively deliver crowd-powered services; First, limited drone control with high-level commands that is user-friendly and comfortable for human interaction. Second, how drones collaborate (form a crowd) to handle or process tasks in parallel for fast evaluation without pilot intervention. Third, how data are collected by a single or a group of drones and how data are processed by a group of smart devices.

The associate editor coordinating the review of this manuscript and approving it for publication was Sayed Chhattan Shah.

[1]www.parrot.com
[2]www.dji.com

In this paper, we address the above mentioned challenges by focusing on two main requirements; the drone should be able to crowdsource to surrounding devices and the master should be able to control multiple drones, in our case, via a high-level scripting approach.

Our contributions in this paper can be summarised as follows:

- Propose and investigate the concept of crowd-powered drone services using three different scenarios.
- Allow a drone to crowdsource tasks to other nearby smartphones for crowd-powered data processing. While an Internet-enabled drone can send tasks to the Internet, i.e., crowdsource processing of tasks to the Internet, we show how a drone can crowdsource tasks (in our example, processing of images it takes) to the collection of local, nearby, devices.
- Develop a scripting framework that allows multiple drones to be automated, programmed and coordinated - this can be useful when multiple drones are used in an application and it is not possible for one human user to control/fly multiple drones manually; for example, to use three drones at the same time for an application, a human user could script the flight of two drones and control one manually on a critical path.
- Evaluate the scripting-based flight programming and coordination approach using various examples of single drone and multidrone flights paths.

This paper is structured as follows. §2 offers a review of related work. The concept and design of *crowd-powered drone services* is described in §3. We have used a potentially real-life scenario to explain the concept and design of crowd-powered drone services. Various aspects of the systems design are also described including system components, drone monitoring paths, and some other design considerations. Subsequently, §4 describes the prototyping and some experimentation we have conducted including a proof-of-concept implementation. Then §5 describes the scripting language we have proposed for coordinating flight paths of multiple drones. This includes the syntax of the language, the workings of the language, and examples of implementations for different flight paths. Finally, concluding comments and future work in §6.

II. RELATED WORK

Recent and ongoing advances in cyber-physical computing technology have changed people's preferences for computing, and there is demand for collective computing paradigms [11] that are more adaptive, and responsive to mobility. As shown in [12], neighbouring mobile devices can be used efficiently as a crowd powered resource to complement the remote clouds. As a type of mobile device, drones are excellent candidates to be used in crowd-computing scenarios due to their versatility [13]–[15]. Existing research on crowd computing [16], [17] has shown that mobile devices can be utilised in a social context to perform large-scale distributed computations, via a static farming method [18]. In crowd computing [19], human expertise is utilised to answer queries which are too complex for search engines and database systems. In Crowdsearch [20], mobile devices are used to perform image search, with the help of human validation via Amazon Mechanical Turk. Also AEROSEE[3] explores drones that are equipped with cameras to find injured or lost climbers and walkers. Images are crowd-searched by people in a web-based environment that work in cross-platform such as PC and smartphones.

The use of smartphones in a generic spatial crowdsourcing platform is discussed in [21], where queries are based on location information. Mobile phones are used to collect sensor data on Medusa [22], based on user-specified sensing tasks. In Rankr [23], an online mobile service is used to ask users to rank ideas and photos. These are primarily concerned with the aspect of crowdsourcing, with the use of mobile devices as tools for accessing an online crowdsourcing service hosted on a remote server. In contrast, Honeybee [24] focuses on offering local computation services to increase performance gain and energy savings. Therefore, we adopt Honeybee as the backbone of our crowd-powered data processing framework. Nevertheless, instead of using Honeybee, we could use other frameworks such as EdgeX,[4] Golem[5] or SONM.[6] However, we chose Honeybee given its ability to dynamically adjust to varying numbers of workers.

Drones have proven their utility in numerous applications in communication, photography, agriculture, surveillance and a variety of public services [25]. Deploying of such drones has issues of safety, security and privacy [26]. The work on [25], explains how Dragnet is emerging as a Cognitive Internet of Things for amateur drone surveillance. Also, the use of drones for spotting sharks is being purposed to monitor areas along the Australian northern NSW coastlines [27]. Operating a number of drones as a team requires strong Infrastructure and connectivity among drones [28], [29]. However, the use of scripting languages assists developers to do various tasks requiring minimal Infrastructure and no connectivity between the drones. For example, DicoScript [30] is proposed for specifying drone details and updating single drone missions during flight. On the other hand, our DroneScript is used to manage multiple drones by creating and maintaining multi-drone missions. Missions are constructed in a way that allow all commands to be sent to the intended drones simultaneously. As each drone has a unique ID, the master immediately identifies the target drone and send the missions accordingly. While there are other scripting languages for drones, some provide GUI-based easy programming methods, such as TYNKER and WORKBENCH,[7] they do not support the behaviour of programming multiple drones, or programming of complex flight paths in a succinct way as we do here.

[3] https://irevolutions.org/2014/02/17/crowd-computing-uav-imagery/
[4] www.edgexfoundry.org
[5] https://golem.network/
[6] https://sonm.io/
[7] https://edu.parrot.com/apps.html

Roldán *et al.* [31] use the drone to collect images in order to build a traffic map of the city, to manage traffic in real time. Areias *et al.* [32] present a platform that provides an abstraction layer between the end-user and the drone to provide drones as a service. It allows the end-user to communicate with the drone using high-level control operations via the platform, but not composing complex flight paths as we do in our work. A more detailed review focusing on issues of physical collision avoidance among drones and UAV-to-UAV communications (or flying ad-hoc networks) is given in [33].

Our approach of utilising a group of surrounding available drones to work together to do a specific task with minimal intervention emerges as a novel idea.

III. CROWD POWERED DRONE SERVICES: CONCEPT AND DESIGN

The objective of the drone services framework that we propose is to use crowd-powered devices to collect and process data in an efficient manner. Such a framework can have a number of applications and can be useful in potentially life-threatening situations. For example, the framework can be applied to find wreckage from airplane crashes, and also to help people stranded at a place due to a fire incident. Other scenarios can be imagined for search and rescue operations or in disaster situations, where multiple drones, even from different owners, can be pooled together to do a particular task, such as searching for survivors, looking for struggling swimmers in the ocean or spotting sharks near beaches. Using example scenarios, this section explains how data can be collected using a single drone or multiple drones as well as how data can be processed by a number of mobile devices that combine both human and machine intelligence.

A. SCENARIOS

Jack and his friends are on holiday in a remote archipelago. They are fond of flying drones and have taken their drones on their holiday for recreational use. It is summertime on the archipelago, and there are a lot of tourists around. A fire has started on one of the island due to a lightning strike. The location of some people who might be in danger on the island that has caught fire, is not known. Fire rescuers have appealed to the public to help locate the people in need in whichever way they can. Jack and his friends have decided to help by utilising crowdsourced devices and techniques. We have presented three approaches in which the crowdsourced devices can be utilised, as described below.

1) SINGLE CROWD - SINGLE DRONE APPROACH

In this approach, one master device will control one drone to collect data, and the collected data will be processed by worker nodes connected to the master device (see Figure 1). Applying this approach to the island fire scenario described previously, Jack will use his smartphone, as the master device, to control his own drone, which may help in locating people trapped on the affected island. Commands and instructions are sent to the drone; these include taking a large number of aerial photographs of the island from different perspectives. Once the drone has completed its mission, data will be transmitted to the master device for processing. The next task would be to go through all the photographs to identify anyone who might be in a dangerous situation. Even though images are geotagged, doing such a task manually will take a long time.

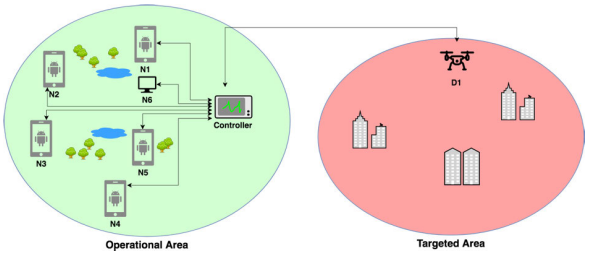


FIGURE 1. Single crowd - single drones approach.

It is proposed that all collected data (i.e., photographs) be shared with a number of people (nodes) connected to the master device, using a work-sharing technique called Honeybee [24]. This technique can share the workload with other people by dividing the major task into smaller jobs. The users are able to review each photograph and help with the massive task efficiently. Jack will be notified when a participant has completed a task, and he will be able to review the findings from the processed photographs one by one. As the fire might spread too fast, using multiple drones will enable speeding up of the photograph reviewing process. The next subsection discusses the Single Crowd - Multiple Drones approach.

2) SINGLE CROWD - MULTIPLE DRONES APPROACH

In this approach, the master device controls a number of drones by creating a network of drones, and the collected data (e.g., photographs) are distributed between worker nodes, as shown in Figure 2. Applying this approach to the island fire scenario described previously, Jack can request his friends or other people who have a drone to connect themselves to the master device. Multiple drones can then be utilised to take aerial photographs efficiently. The master device can deploy the drones in non-overlapping regions of the island. The same task would have taken one drone much longer time. This approach is much better than the Single Crowd - Single Drones approach, as it can save valuable time in potentially life-threatening situations like a fire. Again, the collected data can be distributed to the worker nodes connected to the master device, in the same way as described in the Single Crowd - Single Drones approach. Wind, dry conditions and other factors might put neighbouring islands in a highly threatening situation with respect to the fire. Again, using more drones might help in reducing the amount of required time to serve the affected area. However, using multiple drones means more data to be processed, which may result in delaying the rescue mission. To overcome this issue, we can consider the Multiple Crowd - Multiple Drone approach as described below.

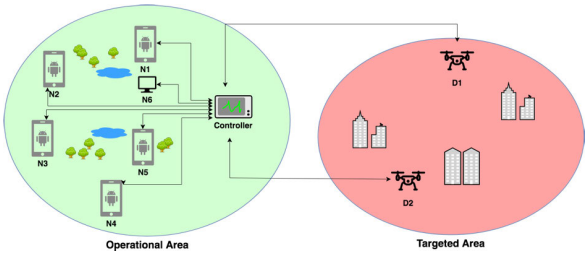


FIGURE 2. Single crowd - multiple drones approach.

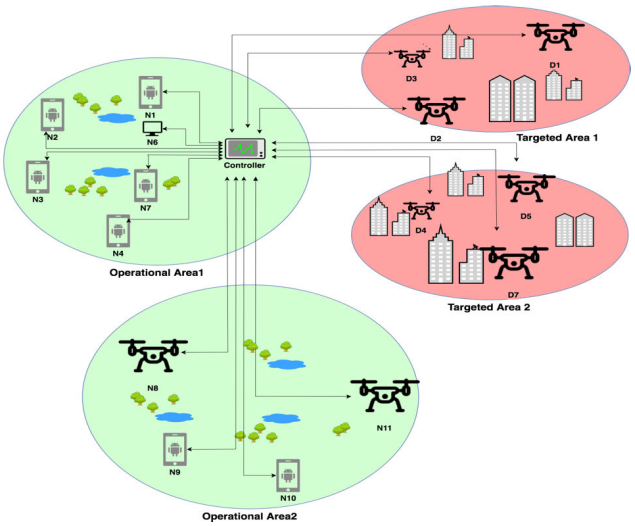


FIGURE 3. Multiple crowd - multiple drones approach.

3) MULTIPLE CROWD - MULTIPLE DRONES APPROACH

The Multiple Crowd - Multiple Drone approach, as shown in Figure 3, is an extension of the Single Crowd - Multiple Drone approach. In this approach, the master devices control many drones, and the collected data is then processed by a number of nodes connected to the master devices. This approach has the advantage of increasing the number of participating drones (to serve or collect data) and worker nodes. It is proven that drones can also be part of worker nodes [34] for data processing. This should allow system scalability through aggregation of a large number of participants in both collecting and processing tasks. As with the previous approach, this might be expensive or hard to run, but it may save valuable time in conceivably hazardous circumstances.

B. SYSTEM DESIGN

While using (non-mobile) infrastructure resources is also a valid method, there are situations when infrastructure may be damaged, inaccessible or not existing. Our work focuses on offloading to the 'crowd' (i.e., nearby mobile devices) in such situations where offloading to non-mobile fixed infrastructure is not feasible, and is intended to complement the infrastructure resources, if and when they do exist. This section describes the design of a system for the above scenarios.

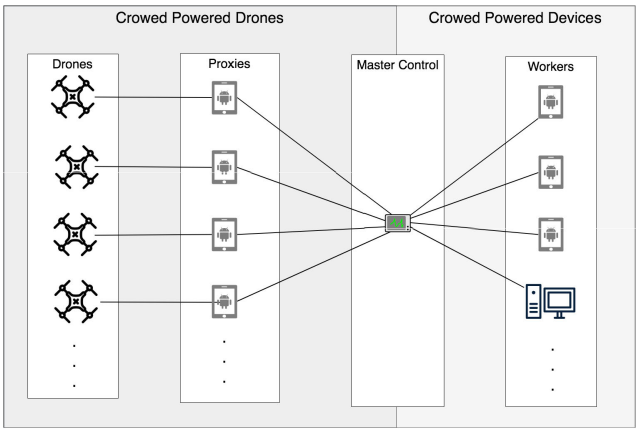


FIGURE 4. Crowd powered drone network architecture.

1) SYSTEM COMPONENTS

The main components of the proposed framework are: Drone nodes ($D_{0,\ldots,i}$), Worker nodes ($N_{0,\ldots,i}$), and a Master controller (see Figure 4). Here, we focus on two key concepts: drone crowdsourcing, and multi-drones service management (via scripting).

First, we developed an application that allows the drone to work with a framework for mobile crowd computing to enable crowd-powered drone services. The implementation and the tests that we have performed, are discussed in detail in §V of this paper. The idea of this service is to allow the user to control one or multiple drones to collect data using a pre-assigned master controller. However, a different aspect of the work which could be optimised is that instead of a pre-assigned master, the master could be changed as in [35]. These data are processed by the crowd using two methods:

- Automated Tasking
- Human Tasking

Second, we developed a scripting language to manage, program and support a framework consisting of multiple drones. The scripting language is focused on programming high-level path behaviours, leaving to local on-drone sensors and controls to avoid obstacles (which we do not focus on in this paper). This scripting language can support different tasks including movement tasks, e.g., take off or land, and processing tasks, e.g., take a photo or crowdsource tasks. Instructions will be sent from the master to all connected drones, each drone may have a similar or different set of commands (or instructions). Once a drone has received an instruction, it will carry it out and transmit the collected information back to the master. The master will then distribute the processing tasks to the worker nodes that are connected to it.

2) DRONE MONITORING PATHS

One aspect of programming drone behaviour is the flight path. The implementation of the autonomous drone missions allows performing many organised operations, especially with the aid of automated piloting [36] and battery charging [37]. Figure 5 shows some examples of autonomous drone monitoring paths. Depending on the requirements of

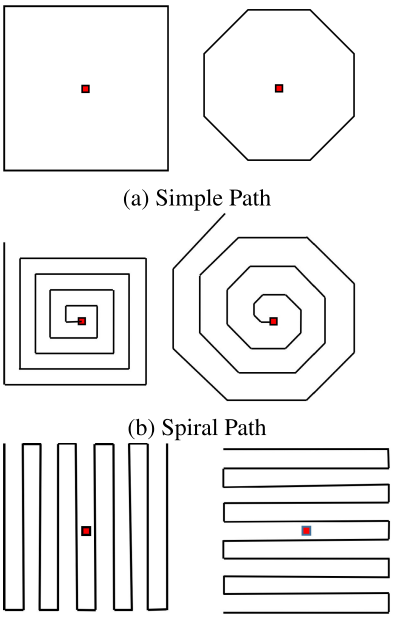


FIGURE 5. Drone monitoring paths.

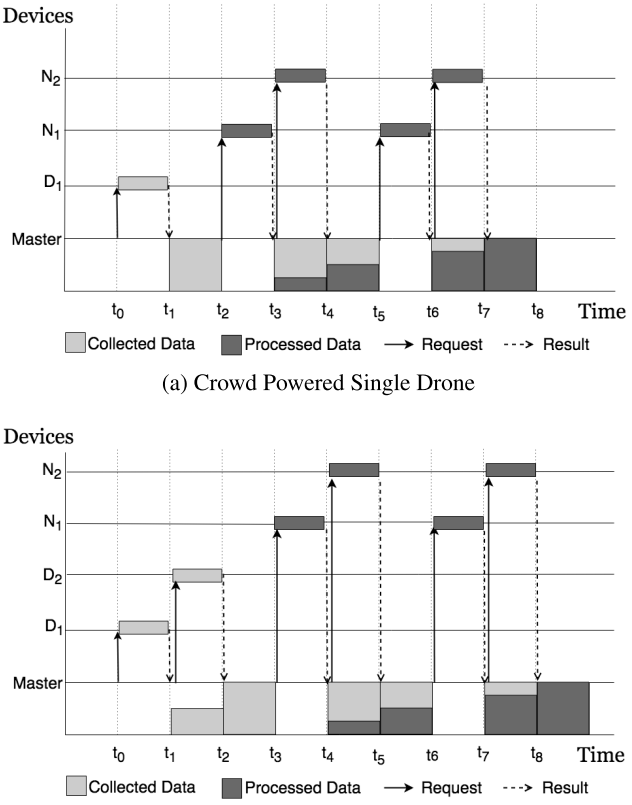


FIGURE 6. Sequence diagrams of single and multiple drones.

each mission, a drone can monitor any targeted area by flying along: a simple path (Figure 5a) to perform simple tasks such as for 2D data, a spiral path (Figure 5b) for intensive photo-covering of an area, or a detailed path (Figure 5c) such as to collect 3D data. Further details are provided in §IV.

To be able to offer the best solution to the problem which the framework will try to solve, the drone monitoring path can be (1) selected from a set of available options (e.g., simple, spiral, detailed or other), (2) it can be prescribed by the user through a drawing or (3) it can be determined using an optimisation algorithm. A range of available options can be fed into the system so the users can select the path which they think is most suitable for a given mission. The options can be developed by considering a number of use cases of the framework. However, since it is not possible to consider all the possible use cases, there should be an option for the users to program a path they think is most suitable for the situation - which we consider in this paper as shown later. Finally, there can also be an option for the users to request a custom path for the problem by running an optimisation algorithm. The optimisation algorithm will require inputs from the user in the form of objectives and constraints (not considered in this paper).

3) DESIGN CONSIDERATIONS

To design such an architecture, we will explain the workflow. First, using the same scenario discussed in §III-A.1 which can be demonstrated in Figure 6a. At time t_0 Jack uses his phone as the master controller to send commands to his own drone D_1 to monitor the targeted area. Jack selects the monitoring path using a list of preset options that are available. At time t_1 he receives the collected aerial images and initialises the total job queue (j). Then using his phone's Wi-Fi Direct, he scans the area looking for available worker devices. The master can accommodate n worker nodes. The results of the Wi-Fi Direct scan indicate the availability of two people who are willing to participate with their devices.

The two workers connect to the master, who then transmits a set of jobs to workers (N_1 and N_2 at times t_2 and t_3, respectively). Workers process the given task based on its requirements and their preference for how they would like to contribute. The options in which the participants can contribute are; offering their machines' resources, or physically or manually interacting with the system (i.e., a mobile app). Once a worker finishes a set of jobs, his/her device sends the results to the master and is given the option to undertake more jobs if offered by the master. As shown in Figure 6a, the master receives results from workers N_1, and N_2, at time t_3 and t_4. The process goes on until the master receives the last jobs which is in this case at t_7, when all the collected data has been processed.

Second, using the scenario discussed in §III-A.2 which can be demonstrated in Figure 6b. This scenario is an extension of the previous scenario and uses more than one drone. At time t_0, Jack uses his phone as the master control to send commands to two drones D_1 and D_2. The monitoring path is selected in the same way as previously described. At time t_1 he receives the collected aerial images from D_1, and at time t_2

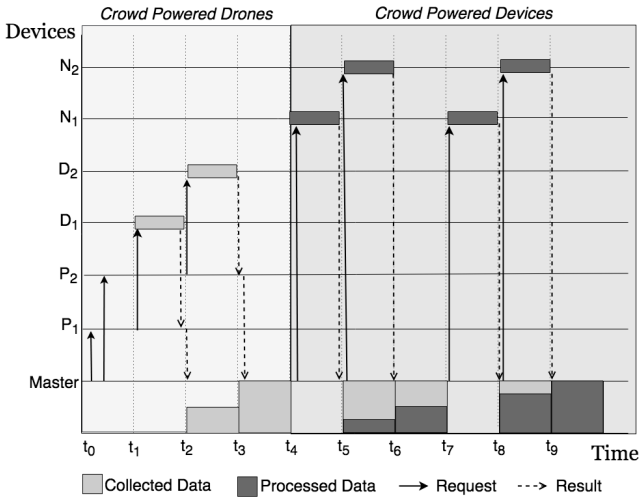


FIGURE 7. Sequence diagram of crowd powered multiple drones (with proxies).

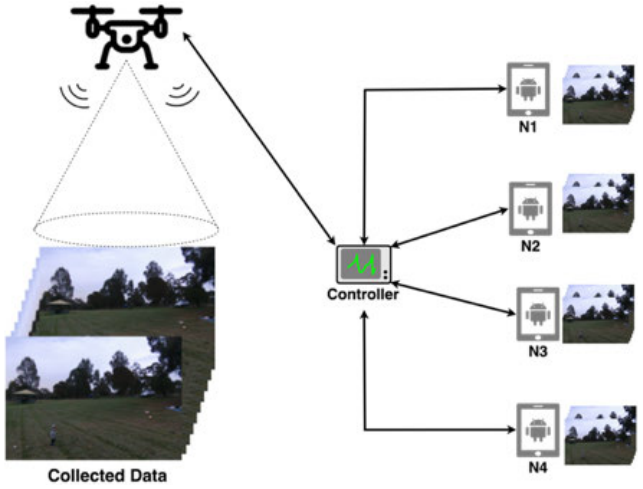


FIGURE 8. High level diagram of the experimental scenario.

he receives the collected aerial images from D_2. The jobs are queued progressively as the images are received by the master from the drones. The process in which the jobs are distributed to the workers and notifications received back by the master are the same as described previously.

In the real world, each drone belongs to one owner who is directly connected to it and can operate it via some controller such as a mobile device. Using these controllers as proxies should allow one or multiple drones to receive commands from a channel (e.g., from a remote server). The channel can be a form of a coordinator of the whole process. It would be preferable for the channel to be present locally (as close as physically possible) as opposed to in another physical location (e.g., another city or country) as this can be less expensive, most likely promise faster connectivity and require no additional infrastructure to build a local cloud. Also, the channel should be compatible with the various individual devices, which might be using different platforms.

As in the scenario, one can imagine people bringing their own drones to be pooled together and controlled by a common master coordinating the search and rescue mission. As shown in the Figure 7, at time t_0, the master sends commands to all connected proxies. Each of the proxies $(P_1, P_2, \ldots, P_n)$ are connected to a single drone $(D_1, D_2, \ldots, D_n$, respectively). When the proxies receive a command from the master, they execute the command. Proxies allow a local level of control.

This way the master commands the drone's indirectly through the proxies. The process to receive feedback works through the same pathway. The information collected by the individual drones is fed back to the proxies, and then the proxies send the information to the master. Then, jobs are queued progressively as the information is received by the master from the proxies. The process in which the jobs are distributed to the workers and notifications received back by the master are simple Honeybee workers scenario, as detailed later.

IV. PROTOTYPING AND EXPERIMENTATIONS ON DRONE CROWDSOURCING FOR DRONE DATA PROCESSING

A. OVERVIEW

The focus here is to process and analyse the data collected by the drone in two ways; using algorithms-based (computational resources of available devices), and using human intelligence who accompany their devices. We have conducted some experiments to allow for a comparison between the two methods. All experiments are conducted using Wi-Fi and Wi-Fi direct to simulate real-time jobs on a portable cloud. Working with low-end devices, Wi-Fi would be a superior alternative, when there is a choice of Wi-Fi available.

The framework ought to have the capacity to change to different conventions, such as 4G or Bluetooth, relying upon user abilities and other circumstances. Furthermore, Honeybee [24] was used as a crowd computing framework to enable mobile devices to share work, and utilise local resources in the mobile context. However, in our experiments, the master control does not participate in processing the data, it only distributes the jobs to connected devices for processing as shown in Figure 8. Using a single master may increase the risk of single point of failure. Thus, we assume that if the master fails, one of the connected proxies can takeover to eliminate the single point of failure. We also assumed that there is at least one device who is willing to share its resources with the master.

B. PROOF OF CONCEPT IMPLEMENTATION

We have assumed a scenario where the master controller has sent a drone to capture 100 photographs on an island with the ultimate objective of identifying people who might be stranded on the island. Each of the photographs taken by the drone needs to be reviewed to identify people (i.e. processing of the data).

The chunk size of each job assigned to each worker is 5 (i.e. the number of images that are allowed to be transmitted at a given time as a part of one request), further details about

TABLE 1. Summary of participating devices.

Device	Type	Role	OS
Drone	Parrot Bebop 2	Collect	Linux
Master	Nexus6, 2.7 GHz Krait 450	Control	Android
Worker1	Nexus7, 1.2 GHz Cortex-A9	Process	Android
Worker2	Nexus7, 1.2 GHz Cortex-A9	Process	Android
Worker3	Nexus7, 1.2 GHz Cortex-A9	Process	Android
Worker4	Nexus7, 1.2 GHz Cortex-A9	Process	Android

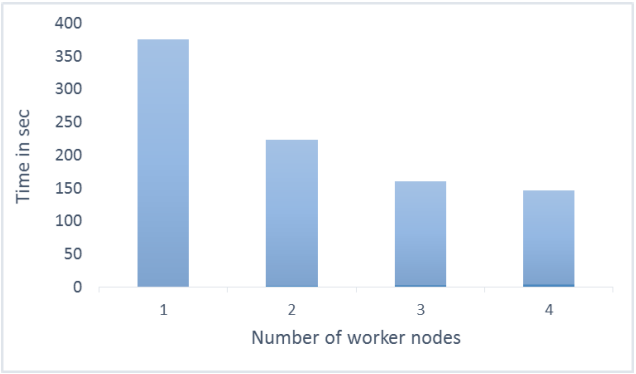


FIGURE 9. Time for machine-based processing of 100 images collected by the drone.

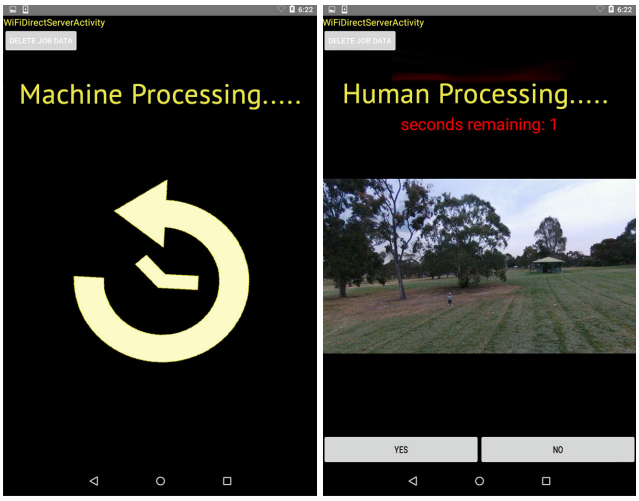
Honeybee parameters are given in [24]. Table 1 summarises the participating devices and the roles they play in the experiment. We carried out implementation of both aspects; the experiments and results are described below.

1) MACHINE-AIDED DATA PROCESSING

The photographs collected from the drone are reviewed using an Android built-in face recognition algorithm. Such kind of algorithms are usually computationally expensive (use high CPU cycles and memory), therefore, it would be better to distribute the task of processing a massive number of photographs to multiple machines (i.e., worker nodes). This approach of distributing the workload will allow for more efficient and faster processing. Each worker mode that is engaged can process some photographs using the Android built-in face detection algorithm and send the results to the master (illustrated in Figure 10a). Figure 9 shows the number of images processed by worker nodes and the total time taken to complete the task.

2) HUMAN-AIDED DATA PROCESSING

Humans are incredible resources when it comes to pattern recognition. According to [38], it usually takes a human less than a second to identify an object in an image. However, it takes time to provide an informative feedback. Consequently, a group of 10 participants were chosen to find out a reasonable amount of response time for identifying humans in images. Aerial images are categorised into ''obviously no one'', ''clearly there is someone'', and a ''mix of both'' (i.e., unsure). Participants were told to click on a button, as soon as they decide if there is or there is not a human in the image, as shown in Figure 10b. The average time taken for a



(a) Machine aided processing (b) Human aided processing

FIGURE 10. Drone data processing applications.

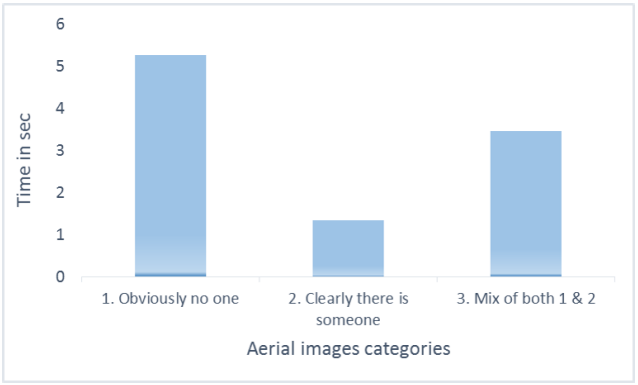


FIGURE 11. Average time taken for a person to process an image.

human to process an image is shown in Figure 11. The results indicate that it took people the longest time to review images which had no one, and least time was taken to identify images which clearly had someone. We have computed and reported the average of all the results from three sets of experiments, which is roughly three seconds.

Humans can be utilised instead of machines for the scenario described in §IV-B.2. The master controller distributed the task of processing 100 images to four worker nodes (i.e., humans) using Honeybee. Each human that is engaged can process some photographs through a manual review (by answering ''yes'' or ''no'' (see Figure 10b) to the following question corresponding to each image: ''Do you see a person in this image?'') and send the results to the master. If any worker node doesn't interact with the given image within three seconds the chosen answer can be set to be either ''yes'' or ''no'' (or ''unsure''). The choice of the default option depends on the context. For example, if time and resources are not seen as critical factors but certainty is, then the default value should be a ''yes''. In time critical situations, where human lives are at stake, and when resources are most needed, wasting time looking for someone based on a photo

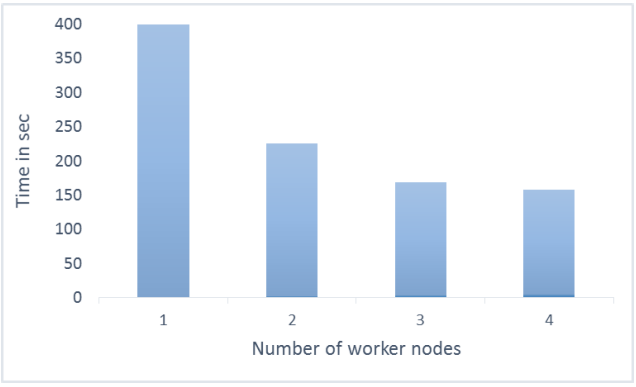


FIGURE 12. Time for humans to process 100 images collected by the drone.

(which might not have a human) can cost more human lives. In this example, we set "no" as the default value if no answers are given by the user for each photo. Figure 12 shows the number of images processed by each human-worker node and the total time taken by each human-worker node to complete the task. It can be seen that the performance is quite similar in this case with the machine-based processing.

C. DISCUSSION

The time taken by each of the worker nodes to process one photograph depends on the image complexity. In both experiments, all worker nodes use a similar type of device which results in a similar processing and computational powers. In the context of machine-aided data processing, the number of photographs processed by the worker nodes varied depending on the sequence and/or availability of time from each node. In our experiment, all the worker nodes had more than enough time available to cumulatively get through the small task of processing 100 images. Using the human-aided data processing, the time taken by each of the worker nodes to process one photograph is around 3 seconds. In this instance too, the human-worker nodes had more than enough time available to cumulatively get through the small task of processing 100 images (approx. 300 seconds or 5 minutes).

The sequence in which the worker node joined the work-share was a more important determinant of the number of images that a node was able to process. The first worker node that joined the work-share earliest and was able to process more images than other nodes. Similarly, each subsequent joiner was able to process fewer and fewer images. Machines are generally faster than humans. This was true for our experiment as well. As shown in Figure 9 and Figure 12, the machine-aided data processing performed slightly better than the human-aided data processing by a few seconds (also, for each image, humans were also given a 3second limit to respond, after which a default value is set) - it must be noted too that with more powerful mobile devices, the machine-aided data processing would be much faster while the human performance would not change (further work is needed to compare the accuracy of human and machine processing, but is not within the scope of this paper).

```
Mission {Drone ID: D1,
         Action Take off: T,
         Action Forward: F50,
         Action Left: L10,
         Action Backward: B50,
         Action Right: R10,
         Action Land: Q }
```

Listing. 1. A single mission.

V. MULTI-DRONES SERVICE MANAGEMENT VIA SCRIPTING

A. OVERVIEW

The management of multiple drones and the collection and distribution of data requires the use of multiple distributed systems. The efficacy of the usage of the proposed application requires seamless and robust integration between the multiple systems [30]. In this section, we describe a scripting language that enables the master controller to manage multiple drones (which might use different flight patterns). In our approach, drone scripting is used to specify the high level control but we assume that there is low level control to deal with certain situations that may arise while flying. If there are many interruption, we assume that a drone may notify the master about the difficulties of fulfilling the path pattern and may return home. For example, if there are many obstacles and the flight path had to keep changing a number of times until its outside certain bounds, then it has to stop and go back home.

We also provide a sample implementation using (Parrot SDK)[8] to demonstrate the concept, i.e., a scripting language for drones, which we call *DroneScript*. Finally, we present our experiments using Sphinx[9] simulation tool, compared with real life flights and mathematical estimates.

B. DRONESCRIPT

DroneScript is a scripting language that enables programmers to easily and quickly create missions for drones using commands issued from the master controller. The essential language elements are *actions* and *missions*. Its syntax is very simple and includes a set of missions and actions separated by commas. Each action is represented by a letter followed by a numerical value to parametrize the respective action. Actions can be defined as the process of changing the drone status, movement or activity. Each drone has its own ID.

A drone has a number of basic movements that can be performed, e.g., Take Off, Left, Right, Forward, Backward, Up, Down and Land. Drones can also perform different type of activities such as taking photos or orbiting around a given target location. Table 2 lists the basic functions that are available in the scripting language. Missions are a combination of actions which are controlled by a constraint. Listing 1 shows an example of a complete mission.

The implemented scripting language not only allows controlling a number of drones to fly predefined paths but also

[8]https://developer.parrot.com/docs/SDK3/
[9]https://developer.parrot.com/docs/sphinx/whatissphinx.html

TABLE 2. Basic commands for Drone Script.

Action	S	Numerical values/ Constraints (C)	e.g
Take off	T	Hovering time in seconds (optional)	T
Left	L	Go (left) distance in meters	L10
Right	R	Go (right) distance in meters	R10
Forward	F	Go (forward) distance in meters	F8
Backward	B	Go (backward) distance in meters	B7
Up	U	Go (up) distance in meters	U10
Down	G	Go (down) distance in meters	G7
Land	Q	Hovering time in seconds (optional)	Q
PIO	P	Predefined location ID	P1
Drone	D	Drone ID	D2
Rotate	Z	(+) right and (-) left degree rotation	Z90
Image taking	I	The interval between images in seconds	I1
Video	V	The length of the video in seconds	V2
Hover	H	The time drone should be idle in seconds	H10

allows commands to be received during the flight. For example, if the drone has already been sent to a location and if it is hovering over a certain location, additional instructions can be sent to the drone for further execution. This way, the mission can be dynamic and open to receiving more actions during flight. Using such a scripting language allows missions to be specified for individual drones, and also allows flying multiple drones simultaneously as well as allowing drones to autonomously accomplish different type of missions - manual piloting of multiple drones is difficult but multiple drones can be scripted. Additionally, the scripts can be analysed to ensure that the task is done in the most efficient manner and without any accidents - for example, to check if no other drone is repeating the same task or preventing a collision between drones. The scripting language is a high level language but it is designed in a way so that it can be translated to lower level languages (e.g., Java APIs) and be compatible with a range of operating-systems/platforms. The same actions in DroneScript can translate into different low-level API calls of different drone platforms, so that DroneScript can be agnostic to the underlying drone platform.

C. COMMANDS AND PROCESSING

DroneScript is based on a time-based approach that calculates the required time to complete each action of a mission as it performs, in relation to delivering a variety of drone services.. It is useful to formulate the completion time of a mission to assess and comprehend the ideal outcome for the situation at hand - a drone script hence allows some reasoning about flight paths (e.g., to estimate the completion time of a path or the energy required) before they are executed.

The completion time T of each action A in a mission M should be known in order to fulfil the needs of the service providers and their clients. Clients need to know an estimate of how long it will take for their service request to be completed. Also drones have some constraints that need to be considered such as the battery level. Therefore, the master device is able to calculate or predict the T required to complete a M or part of it (i.e. A's), which is useful for

```
buffering(BebopDrone mBebopDrone){
    mBebopDrone.goTo(0,0,0,0);
    try {
        Thread.sleep(1000);
    } catch (Exception e) {
        Log.e("Buffering exceptions:", e);}
}
```

Listing. 2. Stabilising time function.

TABLE 3. Time (as estimated via the formula, as determined in simulated drone runs, and as determined in actual drone flights) taken for each basic action in seconds.

Action	Example	Formula	Simulation	Actual
Take off	T(T)	1 + 2b = 3	3.003	3.010
Land	T(Q)	1 + 2b = 3	3.011	3.011
left	T(L100)	d/s + 2b = 22	22.011	22.012
Right	T(R100)	d/s + 2b = 22	22.011	22.011
Forward	T(F100)	d/s + 2b = 22	22.010	22.011
Backward	T(B100)	d/s + 2b = 22	22.011	22.011
Up	T(U10)	d/s + 2b = 4	4.011	4.011
Down	T(D10)	d/s + 2b = 4	4.010	4.012
Rotate	T(Z90)	4 + 2b = 6	6.011	6.011

end-users. Also, each drone needs to know the required time to complete a mission in advance in order to decide if it is capable of doing so (considering application-specific time constraints).

Let n be the number of actions for a given drone D in a given mission, i.e. the mission is denoted by $M(D, A_1, \ldots, A_n)$. Therefore, we can calculate the total time T of a mission as follows:

$$\sum_{n=1}^{n} T(A_n)$$

Each action A has a different T, for example, if a client requests a service that requires a drone to perform a left action with a side length of 50 meters, the $T(L50)$ will be greater than performing the same action with only 10 meters of side length T($L10$). To provide more insight into calculating the time taken for each action:

- Table 3 summaries the basic actions and their time formulas with examples (note that the actual flight times assume no wind and other conditions which would affect flight).
- Drone speed (s) = 5 meters per second
- Stabilising time (b) = 1 second (“buffering” time, i.e., each function or command requires a waiting time before and after processing to allow the drone to complete its movements and be stabilised in its position as shown in Listing 2)
- Distance (d) either given by the user or by calculating the distance between the current location of the drone and the target location.

Using these basic commands, next section demonstrates the ability of drones travelling over different predefined paths and a pre-calculated processing time.

```
left(mBebopDrone,d);
rotateright(mBebopDrone,90);
left(mBebopDrone,d);
rotateright(mBebopDrone,90);
left(mBebopDrone,d);
rotateright(mBebopDrone,90);
left(mBebopDrone,d);
```

Listing. 3. Square function.

D. PATH PATTERNS

Path planning is an important primitive for autonomous mobile robots to ensure that robots can perform missions successfully. Depends on the task the drone is executing, static path planning can be as simple as flying in a straight line from point to point or more complex such as orbiting around an object that is to be viewed at different altitudes. In order to examine the basic commands, this section proposes three different flight path patterns using *DroneScript* to carry out a complete mission; simple path, spiral path, and detailed path. Each example shows the used algorithm to create a drone's path and a formula to calculate the required time to accomplish the task. Then followed by an example with the results obtained, some sample results are given below.

1) SIMPLE PATH EXAMPLES

- **Square:** A square is a geometric shape with four equal side d and four interior equal angles (90-degree). Drone follow a flight path similar to a square of side length d to perform simple tasks such as monitor a house or a block of land. In order to calculate processing time prior the mission, (1) calculate the expected time for the flight to travel along the complete path. Listing 3 shows the code to follow a flight path similar to a square of side length d and rotate by 90 degrees.

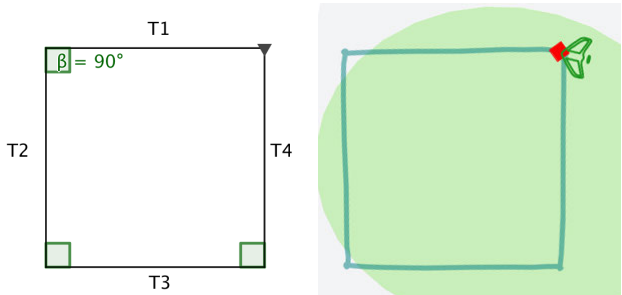
$$4T(L(d)) + 3T(Z(90)) \tag{1}$$

Example 1: A square shape with a side of 100 **T(S100)**
  - Result using the formula (1) is 108 seconds.
  - Result using the simulator is 108.008 seconds.
  - Result using the actual drone is 108.09 seconds.
  - Figure 13 shows the targeted geometric shape 13a and the observed shape 13b.

- **Octagon:** An octagon is a geometric shape with eight equal side d and eight interior equal angles (45-degree). Similar to the square example, drone follow a flight path to perform simple tasks but with shorter rotation angle. The expected time taken for the flight to complete the path can be calculated using the formula in (2). Listing 4 shows the code to follow a flight path similar to an octagon of side length d and rotates by 45 degrees.

$$8T(Ld) + 3T(Z45) \tag{2}$$

Example: An octagon shape with a side of 100 **T(E100)**
  - Result using the formula is 182 seconds.
  - Result using the simulator is 182.470 seconds.

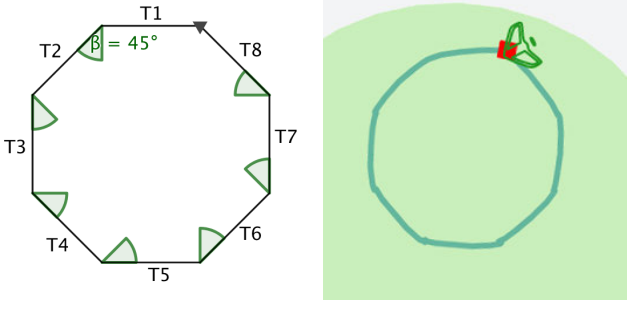


(a) Expected (b) Observed

Figure 13. A square targeted geometric shape.

```
double edge = getSideLength(radius);
for(int i = 1;i<8;i++){
  left(mBebopDrone,edge);
  rotateright(mBebopDrone,45);
}
left(mBebopDrone,edge);
```

Listing. 4. Octagon function.



(a) Expected (b) Observed

Figure 14. A square targeted geometric shape.

  - Result using the actual drone is 182.472 seconds.
  - Figure 14 shows the targeted geometric shape 14a and the observed shape 14b.

2) SPIRAL PATH EXAMPLES

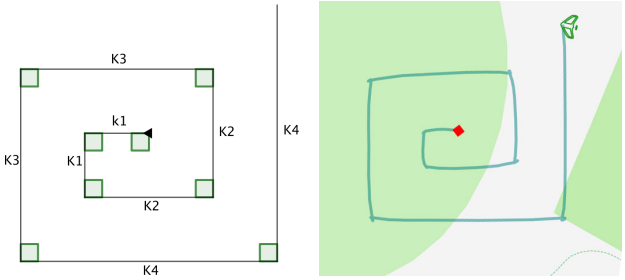
- **Square Spiral:** Square spiral is an extension of a square but only two sides of the shape are equal in what we call a spin unit (k). Then in each spin the shape double its side length. Using the pattern will allow more complexity and may allow more details, in the context of providing drone monitoring services. The expected time taken for the flight to complete the path can be calculated using the formula (3) (which includes k the sum of stabilising times). Listing 5 shows the code to follow a flight path similar to half a square going in a loop.

$$\sum_{n=1}^{k}(2T(L20)n) + 2T(Z90)) + k \tag{3}$$

Example: A Spiral square with a spin unit of 4 **T(M4)**
  - Result using the formula is 148 seconds.

```
for(int i = 1;i<=size;i++){
  rotateright(mBebopDrone,90);
  left(mBebopDrone,i*20);
  rotateright(mBebopDrone,90);
  left(mBebopDrone,i*20);
  buffering(mBebopDrone);
}
```

Listing. 5. Square spiral function.



(a) Expected (b) Observed

Figure 15. A spiral square targeted geometric shape.

```
for(int i = 1;i<=size;i++){
rotateright(mBebopDrone,45);
left(mBebopDrone,i*20);
rotateright(mBebopDrone,45);
left(mBebopDrone,i*20);
rotateright(mBebopDrone,45);
left(mBebopDrone,i*20);
rotateright(mBebopDrone,45);
left(mBebopDrone,i*20);
buffering(mBebopDrone);
```

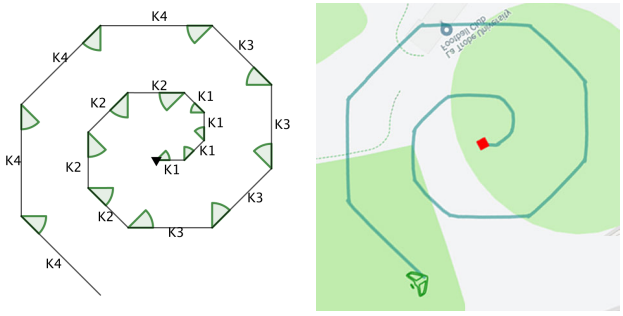
Listing. 6. Octagon Spiral function.

  - Result using the simulator is 148.015 seconds.
  - Result using the actual drone is 148.017 seconds.
  - Figure 15 shows the targeted geometric shape 15a and the observed shape 15b.

- **Octagon Spiral:** Octagon spiral is an extension of an octagon but only four sides of the shape are equal in what we call a spin unit (k). Then in each spin the shape double its length side. The expected time taken for the flight to complete the path can be calculated using (4). Listing 6 shows the code to follow a flight path similar to half an octagon going in loop.

$$\sum_{n=1}^{k}(4T(L20)n) + 4T(Z45)) + k \tag{4}$$

Example: A spiral octagon with a spin unit of 4 **T(N4)**
  - Result using the formula is 292 seconds.
  - Result using the simulator is 292.023 seconds.
  - Result using the actual drone is 292.028 seconds.
  - Figure 16 shows the targeted geometric shape 16a and the observed shape 16b.

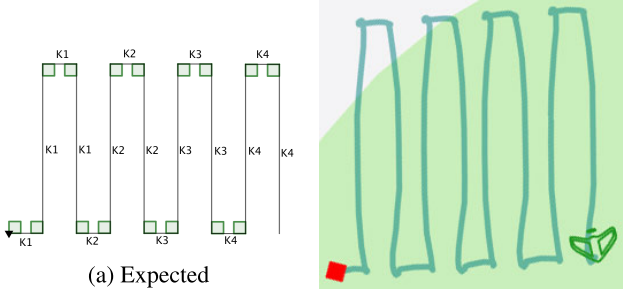


(a) Expected (b) Observed

Figure 16. A spiral octagon targeted geometric shape.

```
for(int i = 1;i<=size;i++){
rotateleft(mBebopDrone,90);
forward(mBebopDrone,10);
rotateleft(mBebopDrone,90);
forward(mBebopDrone,50);
rotateright(mBebopDrone,90);
forward(mBebopDrone,10);
rotateright(mBebopDrone,90);
forward(mBebopDrone,50);
```

Listing. 7. Detailed path function.



(a) Expected (b) Observed

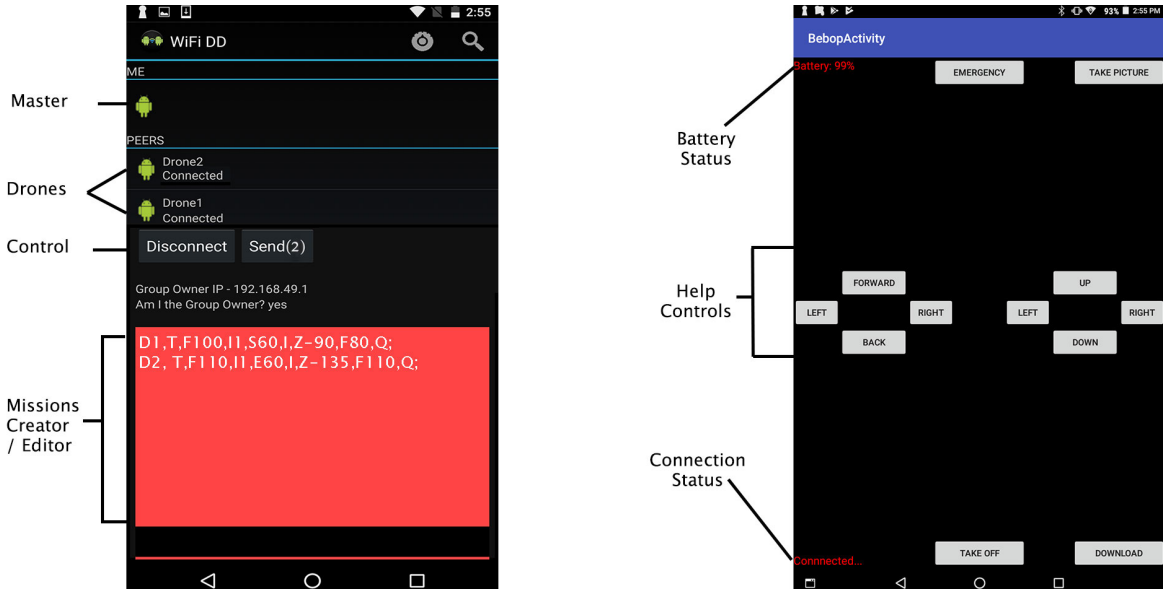
Figure 17. A detailed targeted geometric shape.

3) DETAILED PATH EXAMPLE

For more comprehensive coverage of an area, this pattern allows the drone to cover broad and complex areas of interest, to provide different types of tasks such as generating collaborative maps and 3D models. The expected time taken for the flight to complete the path can be calculated using (5). Listing 7 shows the code to follow a flight path similar to a structured zig-zag pattern.

$$\sum_{n=1}^{k}(4T(Z90) + 2T(F10) + 2T(F10)) + k \tag{5}$$

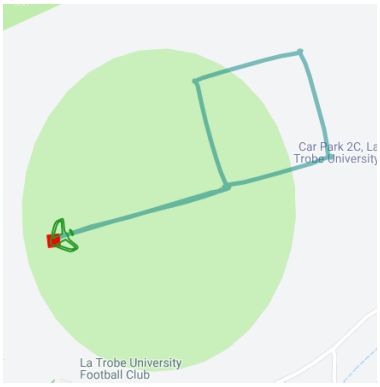
Example: A detailed path with a spin unit of 4 **T(x4)**
- Result using the formula is 228 seconds.
- Result using the simulator is 228.027 seconds.
- Result using the actual drone is 228.030 seconds.
- Figure 17 shows the targeted geometric shape 17a and the observed shape 17b.



(a) Master UI (b) Proxy UI

Figure 18. Drone data collecting applications.

TABLE 4. Multiple drones missions.

Drones:	
D1	D2
Missions:	
M1{D1,T,F100,I1,S60,I,Z-90,F80,Q}	M2{D2, T,F110,I1,E60,I,Z-135,F110,Q}
Map:	
	
Total Time taken:	
T(M1) = 132.027 seconds	T(M2) = 190.499 seconds
Collected Data:	
76 aerial images	133 aerial images

E. MULTIPLE DRONE EXAMPLE

Let us consider a scenario where there are three mobile devices available, one phone (as a master controller shown in Figure 18a), and two tablets (as proxies shown in Figure 18b) and we have two drones. Using the same scenario of the fire on the island described previously, the script to collect photos from the island can be executed using the phone. The instructions from the script will be relayed to the two tablets by the master controller which are the respective local controllers for the two drones. The tablets can then send the instructions using the script to the two respective drones. The drone can go on separate paths, collect photographs and relay them to the proxies.

Table 4 shows the flight paths (using different path patterns) followed by the two drones. The master controller sends the commands to the first and the second tablets (proxies) to instruct their respective drones to follow path resembling a square and an octagon. The two tablets then relay the instructions to their respective drones. Drone 1 and 2 follow the paths they have been instructed to follow as shown in the table below in approx. 132 and 191 seconds, respectively. Furthermore, drone 1 and 2 captured 76 and 133 photos, respectively. The photos are then relayed back to their respective proxies. The notion of the proxies has a number of advantages, e.g., to allow local personalised control if the commands somehow cannot be carried out, and to extend the range of communication between the master controller and the drones.

F. DISCUSSION

We note also that some drones have built-in obstacle avoidance and so our script provides a ''high-level'' paths or behaviour for the drones, and the drones can follow the paths, taking into account low-level variations and changes. Also, such drone scripts can be executed like Cron jobs enabling repeated complex drone behaviours to be easily created, or triggered by context conditions.

By analysing DroneScripts, the behaviour of drones can be predicted, e.g., time estimations for complex drone missions comprising multiple drones and composite behaviours can be analysed and determined beforehand, by improving the time estimates of component behaviours. Decisions about missions, whether they can be carried out or not, and charging or billing models for drone services can then be based on such estimates.

VI. CONCLUSION

The prevalence and usage of both smartphones and drones is on the rise. Smartphones are constrained by resources and capabilities to undertake large tasks (e.g. computationally expensive tasks) and individual drones are constrained by physical limitations (e.g. number of photos they can take or area they can cover).

A review of related work has indicated that the idea proposed is unique and exploring the synergies of smartphones/wearables and drones would be valuable contribution in this domain. In this concept paper, we have proposed a framework for drone crowdsourcing and multi-drones service management via scripting. We demonstrated the concept using an example of an island where there has been a fire and how drones and other devices assisted with locating the people who were stranded in dangerous situations. But many scenarios can be imagined for our framework. We have discussed the various aspects of design of such a framework including the system components, path specification and monitoring. The data processing aspects of the concept have also been described. A scripting language which can control and manage the systems has also been proposed.

Future work will involve exploring further applications of crowdsourcing from drones, including providing human input and machine resources for drones performing complex tasks in smart city environments (e.g., guiding drones as they perform tasks such as guiding tourists, emergency settings, patrol, or inspection). Also, an aspect of future work is investigating how to avoid single point of failure by using a redundant master. There are other challenges not addressed in this article, mainly energy [33], low-level collision avoidance, and multi-drones autonomous cooperation when performing tasks that require multiple drones [39]. We will be working on developing some cooperative strategies and dealing with the energy limitation. Note that a key contribution of this paper is the idea of high-level scripting of multiple drones by a user via DroneScript.

We will also investigate implementing DroneScript across a range of different drone platforms, so that drones from different manufacturers can work together, and also exploring further multi-drone services scripted via DroneScript. The scripting language enables a means of estimating drone mission times and resources - we can extend this to estimate energy and other resources (for drone actions) required for a given mission. Also, interesting is to explore the notion of mission in [40] where instead of mobile software agents carrying out cyber-world missions, we have drones carrying out physical-world missions. Our path pattern approach is a high-level control operation, with the system relying on the low level drone sensing and control to deal with certain situations that may arise such as avoiding immediate obstacles along the way. However, if the drone deviates too far from the intended path (e.g., go outside certain threshold bounds), it has to be reported back to the master and the drone has to come home, this will be an important avenue for future work.

We envision smart civilian drone services in the future, and we have only contributed towards two aspects of this in this paper: involving human and machine processing of drone data and automating multidrone flights. Further work is required to explore other aspects, from AI-based smarter drones to smarter human control of drones.

REFERENCES

[1] M. Alwateer, S. W. Loke, and A. M. Zuchowicz, ''Drone services: Issues in drones for location-based services from human-drone interaction to information processing,'' *J. Location Based Services*, vol. 13, no. 2, pp. 94–127, 2019.

[2] A. Claesson, L. Svensson, P. Nordberg, M. Ringh, M. Rosenqvist, T. Djarv, J. Samuelsson, O. Hernborg, P. Dahlbom, A. Jansson, and J. Hollenberg, ''Drones may be used to save lives in out of hospital cardiac arrest due to drowning,'' *Resuscitation*, vol. 114, pp. 152–156, May 2017.

[3] C. A. Thiels, J. M. Aho, S. P. Zietlow, and D. H. Jenkins, ''Use of unmanned aerial vehicles for medical product transport,'' *Air Med. J.*, vol. 34, no. 2, pp. 104–108, 2015.

[4] L. Tang and G. Shao, ''Drone remote sensing for forestry research and practices,'' *J. Forestry Res.*, vol. 26, no. 4, pp. 791–797, Dec. 2015.

[5] E. Natalizio, R. Surace, V. Loscri, F. Guerriero, and T. Melodia, ''Filming sport events with mobile camera drones: Mathematical modeling and algorithms,'' HAL, Lyon, France, Res. Rep. hal-00801126, 2012.

[6] S. W. Loke, M. Alwateer, and V. S. A. A. A. Don, "Virtual space boxes and drone-as-reference-station localisation for drone services: An approach based on signal strengths," in *Proc. ACM 2nd Workshop Micro Aerial Vehicle Netw., Syst., Appl. Civilian Use*, 2016, pp. 45–48.
[7] S. Daftry, C. Hoppe, and H. Bischof, "Building with drones: Accurate 3D facade reconstruction using MAVs," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2015, pp. 3487–3494.
[8] A. Shukla, H. Xiaoqian, and H. Karki, "Autonomous tracking and navigation controller for an unmanned aerial vehicle based on visual data for inspection of oil and gas pipelines," in *Proc. 16th Int. Conf. Control, Automat. Syst. (ICCAS)*, Oct. 2016, pp. 194–200.
[9] M. Bacco, A. Berton, E. Ferro, C. Gennaro, A. Gotta, S. Matteoli, F. Paonessa, M. Ruggeri, G. Virone, and A. Zanella, "Smart farming: Opportunities, challenges and technology enablers," in *Proc. IEEE IoT Vertical Top. Summit Agricult.-Tuscany (IOT)*, May 2018, pp. 1–6.
[10] S. W. Loke, *Crowd+Cloud Machines*. Cham, Switzerland: Springer, 2017, pp. 11–25. doi: 10.1007/978-3-319-54436-6_2.
[11] G. D. Abowd, "Beyond weiser: From ubiquitous to collective computing," *Computer*, vol. 49, no. 1, pp. 17–23, Jan. 2016.
[12] N. Fernando, S. W. Loke, and W. Rahayu, "Computing with nearby mobile devices: A work sharing algorithm for mobile edge-clouds," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 329–343, Apr./Jun. 2019.
[13] A. Fotouhi, M. Ding, and M. Hassan, "Flying drone base stations for macro hotspots," *IEEE Access*, vol. 6, pp. 19530–19539, 2018.
[14] N. H. Motlagh, M. Bagaa, and T. Taleb, "UAV-based IoT platform: A crowd surveillance use case," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 128–134, Feb. 2017.
[15] E. Salisbury, S. Stein, and S. Ramchurn, "Real-time opinion aggregation methods for crowd robotics," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2015, pp. 841–849.
[16] D. G. Murray, K. Nilakant, J. Crowcroft, and E. Yoneki, *Task Farming Crowd Computing*. Hoboken, NJ, USA: Wiley, 2013, ch. 13, pp. 491–513. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118511305.ch13
[17] K. Parshotam, "Crowd computing: A literature review and definition," in *Proc. South Afr. Inst. Comput. Scientists Inf. Technologists Conf. (SAICSIT)*, New York, NY, USA, 2013, pp. 121–130. doi: 10.1145/2513456.2513470.
[18] D. G. Murray, E. Yoneki, J. Crowcroft, and S. Hand, "The case for crowd computing," in *Proc. SIGCOMM Workshop Netw., Syst., Appl. Mobile Handhelds (MobiHeld)*, New York, NY, USA, 2010, pp. 39–44. doi: 10.1145/1851322.1851334.
[19] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin, "CrowdDB: Answering queries with crowdsourcing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, New York, NY, USA, 2011, pp. 61–72. doi: 10.1145/1989323.1989331.
[20] T. Yan, V. Kumar, and D. Ganesan, "CrowdSearch: Exploiting crowds for accurate real-time image search on mobile phones," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, New York, NY, USA, 2010, pp. 77–90. doi: 10.1145/1814433.1814443.
[21] Z. Chen, Rui Fu, Z. Zhao, Z. Liu, L. Xia, L. Chen, P. Cheng, C. C. Cao, Y. Tong, and C. J. Zhang, "gMission: A general spatial crowdsourcing platform," in *Proc. VLDB Endowment*, vol. 7, no. 13, pp. 1629–1632, 2014.
[22] M.-R. Ra, B. Liu, T. F. La Porta, and R. Govindan, "Medusa: A programming framework for crowd-sensing applications," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, New York, NY, USA, 2012, pp. 337–350. doi: 10.1145/2307636.2307668.
[23] Y. Luon, C. Aperjis, and B. A. Huberman, "Rankr: A mobile system for crowdsourcing opinions," in *Mobile Computing, Applications, and Services*, J. Y. Zhang, J. Wilkiewicz, and A. Nahapetian, Eds. Berlin, Germany: Springer, 2012, pp. 20–31.
[24] N. Fernando, S. W. Loke, and W. Rahayu, "Honeybee: A programming framework for mobile crowd computing," in *Proc. Int. Conf. Mobile Ubiquitous Syst., Comput., Netw., Services*. Berlin, Germany: Springer, 2012, pp. 224–236.
[25] G. Ding, Q. Wu, L. Zhang, Y. Lin, T. A. Tsiftsis, and Y.-D. Yao, "An amateur drone surveillance system based on the cognitive Internet of Things," *IEEE Commun. Mag.*, vol. 56, no. 1, pp. 29–35, Jan. 2018. doi: 10.1109/MCOM.2017.1700452.
[26] K. W. Smith, "Drone technology: Benefits, risks, and legal considerations," *Seattle J. Environ. Law*, vol. 5, no. 1, p. 1, 2015.
[27] O. Chang. (2016). *NSW Premier Mike Baird has Launched a $250,000 Shark-Spotting Drone*. [Online]. Available: https://www.businessinsider.com.au/nsw-premier-mike-baird-has-launched-a-250000-shark-spotting-drone-2016-2
[28] J. Lee, K. Kim, S. Yoo, A. Y. Chung, J. Y. Lee, S. J. Park, and H. Kim, "Constructing a reliable and fast recoverable network for drones," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.
[29] S. Yoo, K. Kim, J. Jung, A. Y. Chung, J. Lee, S. K. Lee, H. K. Lee, and H. Kim, "Poster: A multi-drone platform for empowering drones' teamwork," in *Proc. ACM 21st Annu. Int. Conf. Mobile Comput. Netw.*, 2015, pp. 275–277.
[30] A. El-Sayed and M. ElHelw, "Distributed component-based framework for unmanned air vehicle systems," in *Proc. IEEE Int. Conf. Inf. Automat. (ICIA)*, Jun. 2012, pp. 45–50.
[31] J. J. Roldán, P. Garcia-Aunon, E. Peña-Tapia, and A. Barrientos, "SwarmCity project: Can an aerial swarm monitor traffic in a smart city?" in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (UNAGI)*, Mar. 2019, pp. 862–867.
[32] B. Areias, N. Humberto, L. Guardalben, J. M. Fernandes, and S. Sargento, "Towards an automated flying drones platform," in *Proc. VEHITS*, 2018, pp. 529–536.
[33] N. H. Motlagh, T. Taleb, and O. Arouk, "Low-altitude unmanned aerial vehicles-based Internet of Things services: Comprehensive survey and future perspectives," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 899–922, Dec. 2016.
[34] M. Alwateer, S. W. Loke, and W. Rahayu, "Drone services: An investigation via prototyping and simulation," in *Proc. IEEE 4th World Forum Internet Things (WF-IoT)*, Feb. 2018, pp. 367–370.
[35] W.-S. Jung, J. Yim, and Y.-B. Ko, "Adaptive offloading with MPTCP for unmanned aerial vehicle surveillance system," *Ann. Telecommun.*, vol. 73, nos. 9–10, pp. 613–626, 2018.
[36] F. Flammini, R. Naddei, C. Pragliola, and G. Smarra, "Towards automated drone surveillance in railways: State-of-the-art and future directions," in *Proc. Int. Conf. Adv. Concepts Intell. Vis. Syst.* Cham, Switzerland: Springer, 2016, pp. 336–348.
[37] C. H. Choi, H. J. Jang, S. G. Lim, H. C. Lim, S. H. Cho, and I. Gaponov, "Automatic wireless drone charging station creating essential environment for continuous drone operation," in *Proc. IEEE Int. Conf. Control, Automat. Inf. Sci. (ICCAIS)*, Oct. 2016, pp. 132–136.
[38] S. Thorpe, D. Fize, and C. Marlot, "Speed of processing in the human visual system," *Nature*, vol. 381, no. 6582, p. 520, 1996.
[39] A. S. Aghdam, M. B. Menhaj, F. Barazandeh, and F. Abdollahi, "Cooperative load transport with movable load center of mass using multiple quadrotor UAVs," in *Proc. IEEE 4th Int. Conf. Control, Instrum., Automat. (ICCIA)*, Jan. 2016, pp. 23–27.
[40] G. T. Jayaputera, S. W. Loke, and A. B. Zaslavsky, "Design, implementation and run-time evolution of a mission-based multiagent system," *Web Intell. Agent Syst.*, vol. 5, no. 2, pp. 139–159, 2007. [Online]. Available: http://content.iospress.com/articles/web-intelligence-and-agent-systems-an-international-journal/wia00110

MAJED ALWATEER received the B.S. degree from Canterbury University, New Zealand, in 2012, and the M.S. degree in computer science from La Trobe University, Australia, in 2014, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Information Technology. He is also a member of the Distributed Systems and IoT Research Group, Deakin University, Australia. His research interests and expertise are primarily in the areas of pervasive computing, mobile computing, drone computing, the Internet of Things, and service oriented architecture, working with technologies like Android development, AnyLogic, and software engineering.

SENG W. LOKE was a Reader and an Associate Professor with the Department of Computer Science and Information Technology, La Trobe University. He is currently a (Full) Professor of computer science with the School of Information Technology, Deakin University, where he co-directs the IoT cluster in the School. His research interests include pervasive (ubiquitous) computing and mobile computing, the Internet of Things (IoT), focusing on issues concerning systems and information, with current emphases on complex cooperation among things (including smart vehicles viewed as smart things, i.e., the Internet of Vehicles, the Internet of Drones, and so on), crowd-powered mobile computing, mobile big data, mobile big systems, the social impact of mobile technology innovation, mobile/physical web/cloud development, and how they might interact. Some of his work can be categorised under mobile cyber-physical systems.

NIROSHINIE FERNANDO received the Ph.D. degree in computer science from La Trobe University, Melbourne, Australia, in 2015. She is currently a Lecturer of software engineering with the School of Information Technology, Deakin University. Her research interests include the IoT, edge and fog computing, software engineering, and pervasive computing.