Multi-Agent Systems:

Discovering UAV's in an agricultural setting using decentralised ground stations

Arryon Tijsma, Noël Lüneburg

April 2015

1 Introduction

For many years now, the commercial sector has been researching and funding the replacement of man with automated machines. The most prominent examples are robots in factories. However, the operation space of these robots is often restricted to certain paths on the floor and in some cases they are attached to stationary objects.

Recently, outdoor exploration has been a significant candidate for automation. An example is the agricultural sector, where regular inspection of crops is important. Unlike factories, farmlands are large and have a wide open space above them from which the land can be observed unobstructed. It becomes then the ideal scenario for simple flying UAVs (which we will later refer to as drones).

By using cheap and small drones for exploration one has the option to combine multiple drones which can cooperate by communicating to execute certain tasks as a group. Drones normally require high-detail wireless connections to transmit data, however, these connections have a limited range. We propose the use of ground stations distributed in a field to serve as landmark guides and data recipients for the drones. Communication between all ground stations will link the entire system together, creating a decentralised or distributed knowledge system. We have developed an application that simulates a field containing drones and ground stations in order to research the acquisition and sharing of knowledge.

Our research question is: how to describe spatial knowledge within a decentralised network of agents using migrating drones as sensors?

2 Simulation

Simulation setup

The simulation takes as input the amount of drones and the amount of ground stations. These are then both distributed randomly while making sure the range of each ground station does not overlap another. This is done for simplicity, although a drone could in theory connect with multiple stations or just the one closest by.

Each drone starts by hovering in a stationary position, which either falls outside or inside some station's range. When a drone is located inside a station's range, that station immediately connects to the drone and will then communicate to the other stations that it knows the position of that drone. Communication between stations can potentially fail or take some time which means that whenever a drone reaches a station it can take a little while before every other station is aware of this.

Heuristics

Some drones will not initially be positioned inside any station's range and so they need to be guided to one of the stations in order for their position to be known. This is done through a station's ability to send out "satellite" drones. A drone with a known location will be sent away from the station to scout for lost drones. Whenever such a satellite drone gets close enough to such a drone the satellite will tell the drone which station it came from and how to get there. The lost drone will then follow that direction until it arrives at the station.

Whenever a satellite drone has flown for too long it will lose track of the path from its current position to its associated station, and will now be lost. It will still try to return to its previously associated ground station, but will connect with every station it encounters along the way. Each station will send out a satellite drone at certain interval. Only one drone is sent out for every station at a time to prevent losing drones more often than obtaining new ones.

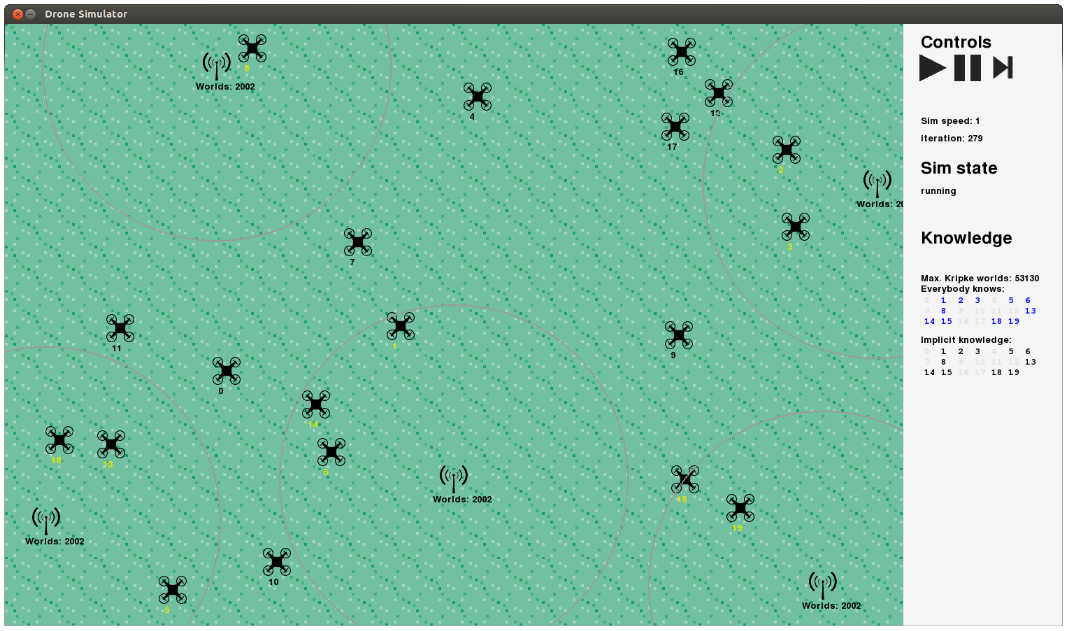
Eventuality

With communication between ground stations, we assume that eventually messages will arrive to every ground station. Therefore, we let every ground station wait until it has received all communications of all other ground stations. We simulate the eventuality by introducing a random delay between 0 and five seconds between the communication of stations.

Between drones however, exists the possibility that a drone flies too fast or is too busy navigating, to be able to tell an idling drone to go to its associated ground station. We aid this possibility implicitly by iterating the logic of our simulation at a lower rate than the visual part of it. That way, it can happen that a drone is visually flying past and out of range of another drone before the simulation has a chance to communicate with that drone. The total area in which the drones can fly is restricted by an outer border to ensure that each drone will eventually reach a station. Whenever a drone reaches a border it will bounce back.

Goal

The goal of the program is to locate all drones in order for a task planner (not part of the application) to plan a new global task, taking into account every drone location. This requires a definition of drones being lost and drones with a known location. We say a drone is lost whenever it is positioned outside of the range of every ground station. Whenever it is hovering stationary within the range of a station, the corresponding station knows the location of that drone. In a real environment this could for instance be achieved using a GPS module on a drone that transmits its position to the station when in range.



```
Controls

Sim speed: 1
iteration: 279

Sim state
running

Knowledge

Max. Kripke worlds: 53130
Everybody knows:
 0   1   2   3   4   5   6
 7   8   9  10  11  12  13
14  15  16  17  18  19

Implicit knowledge:
 0   1   2   3   4   5   6
 7   8   9  10  11  12  13
14  15  16  17  18  19
```

3 Multi-Agent System logic

Epistemic properties

The picture above shows an example scenario of the simulation. Each drone is associated with a number which represents its name. If the number is shown in black it means that none or not all of the stations know the location of that drone. If the number is shown in yellow the drone is known to the station it is closest to and this fact has been communicated between all stations.

In the frame on the right of the picture the 'everybody knows' and 'implicit knowledge' are shown. 'Everybody knows' lists the drones of which all stations know their locations. 'Implicit knowledge' is weaker and lists the drones that are known when combining the knowledge of each individual station. Implicit knowledge of drone d at station s holds whenever station s knows the location of drone p and all other stations consider it possible that drone p can be present at station s.

Kripke worlds

The configuration of connections between drones and ground stations is modeled using Kripke worlds. The connectivity of the drones with the stations is represented per world as a *global state*. The global state $S = (s_1, ..., s_m)$ is a tuple of values from local states, where in our simulation, the value of the local state of each ground station is the number of drones connected to that ground station. Say we have two ground stations and two drones, the global state is a tuple of two values, and the amount of kripke worlds possible will be 6, defined by the possible combinations of global states: $(0,0), (1,0), (0,1), (1,1), (2,0), (0,2)$.

Kripke worlds relate to the uncertainty for agents in an environment, and all the possible options available. When starting the simulation, nothing is known about the positions of the drones, so all possible Kripke worlds adhering to our restrictions (only one associated ground station per drone) are possible for all stations. If there is only one Kripke world possible for all agents, that means there is no uncertainty, which is what we strive for.

For each agent, the possible Kripke worlds can be reduced by gaining knowledge, and increased by losing knowledge. Say we have two drones and two stations, and we connect one drone to station 1. The Kripke world where station 1 has a local state of zero has become impossible, so we've reduced the number of Kripke worlds. Conversely, if the drone gets disconnected from the station, that world has become possible again.

Derivation of knowledge

When we want to know if 'Every agent knows that p', we need to have some formula p with an associated truth value, for each of the accessible Kripke worlds, for all agents. Otherwise we won't be able to infer the predicates 'Everybody knows' and 'It is implicit that'.

In our simulation, this is done by considering the drones to be the logical formulas, and the knowledge about the position of the drone as the truth value for the formula. If an agent knows where a drone p is, $K_i(p)$ but also $I(p)$ hold. Otherwise $K_i(\neg p)$ holds. If all agents know where drone p is, $E(p)$ holds and we have achieved a high level of knowledge about that drone.

So what is the impact on not knowing a drone's location on the possible Kripke worlds? Say station 1 doesn't know where drone 3 is. It can assume that either the drone will make a connection with one of the other ground stations eventually, or something might be wrong with it meaning it will never connect with any station. This situation equals, if everything else stays the same, a number of possible Kripke worlds equal to the number of ground stations, plus one.

4 Results

We have run simulations to determine the behavior of the drones using the provided heuristics and to determine how fast the number of Kripke worlds converges to one, until all drone positions are known. Since there is probability involved we ran the simulation fifteen times for each of the combinations of drones versus ground stations.

Generation of the cartesian product for the initial Kripke worlds is a consuming task that grows exponentially with the number of ground stations added. Therefore, combinations of many drones and many ground stations are not suitable for our simulation. Table 1 lists the configurations tested and shows their convergence results.

The table shows that the possible Kripke worlds grow exponentially at higher values of $drones \cdot stations$. It shows that with more ground stations, drones are discovered more easily, which is logical. It also shows that if more drones are present in the same area, they are located faster, probably because the chances another drone will get in their vicinity is larger.

5 Conclusion and discussion

Our research question was: how to describe topological/spatial knowledge within a decentralised network of agents using migrating drones as sensors?

As an answer to this question, the simulation proves quite useful. It seems with the correct strategy, it is fairly simple to simulate a situation where drones will gravitate towards various ground stations until their position is known and communicated between all ground stations. Using more drones will not guarantee a faster convergence to full knowledge but adding more stations will significantly speed up the process. The reason behind this is quite simple: The drones will have less space where they can be lost as more ground station will cover more areas in which locations of drones are known.

In the simulation we applied simplifications that can impact the performance in real life. For instance, sensor input can make navigation much harder for a

Drones	Stations	Kripke worlds	Mean	Median	Std. dev
10	2	66	669.0	608	295.0
10	3	286	622	685	207.8
10	4	1001	643.7	530	476.9
10	5	3003	175.5	146	121.8
10	6	8008	146.1	121	86.8
20	2	231	748.0	669	333.4
20	3	1771	**925.5**	**843**	519.1
20	4	10626	202.9	165	88.0
20	5	53130	151.5	155	59.6
30	2	496	842	705	420.3
30	3	5456	582.4	545	239.3
30	4	46376	117.2	164	77.8
30	5	**324632**	187.9	160	114.8
40	2	861	296.2	293	74.14
40	3	12341	297.1	247	203.8
50	2	1326	326.3	305	117.5
50	3	23426	223.3	224	62.2
60	2	1891	234.7	276	123.5
60	3	39711	230.9	233	49.1

Table 1: Displays for each configuration of number of drones and ground stations the initial number of Kripke worlds and the mean, median and standard deviation of the iterations required to converge to one Kripke world. Bold values indicate the maximum value for the corresponding column.

drone. While communication and navigation can in theory be done by GPS and WiFi modules, accurate GPS is costly and heavy, so a drone might have to rely on its vision to navigate to other drones or ground stations, making the real life task more difficult.

Another assumption we made is that all ground stations can communicate with each other eventually. Even with a wired connection, fault tolerance is desirable, so a ground station shouldn't wait for all communication to finish before updating its knowledge.

An improvement that goes hand in hand with the last remark would be not to know beforehand how many ground stations there are, but instead having the ability to add more stations on the fly and integrate their knowledge into the system. This would make the whole approach more flexible, also allowing for easily replaceable systems in case of failure.