E 150: Project 2

Fayyad Azhari

October 10, 2019

1 Introduction

For this project, we aim to develop a simulation of UAVs flying through a developed model of terrain which littered with obstacle and targets. We are trying to let the UAVs get to the targets without crashing into each other or the obstacles. The main objective is to find the best setup to get to the targets as fast as possible, with minimal loss of UAVs. A combination of a genetic algorithm and machine learning is used to solve the optimization problem.

The algorithm works by repeatedly altering a set population of solutions. A fitness of the solution is computed and over several generations or repetitions the population moves towards a more optimal solution. The UAVs modeled can move in any direction and are idealized as point masses. Targets are mapped when agents are within a certain distance from them, and agents crash when they get to a certain distance from the obstacles. Agents can also crash if they get too close to another agent, and also if they leave a certain pre-defined region. A trial is terminated once all the targets are mapped or all agents crash. Through the process of the simulation we will try and observe how many of the UAVs crash, and how fast all the targets are reached. This paper will show how the dynamics of the drone is set up, the behavior of the control model and the behavior of the UAVs.

2 Background and Theory

Key Equations A fixed Cartesian coordinate system is used throughout the project and the unit vector is given by (e_1, e_2, e_3) which is a mutually orthogonal triad. The first key equations that used to model the dynamics of the agents is the position equation which is

$$\mathbf{r} = r_1\mathbf{e_1} + r_2\mathbf{e_2} + r_3\mathbf{e_3}$$

and the velocity equation is

$$\mathbf{v} = \dot{\mathbf{r}} = \dot{r_1}\mathbf{e_1} + \dot{r_2}\mathbf{e_2} + \dot{r_3}\mathbf{e_3}$$

and acceleration is

$$\mathbf{a} = \ddot{\mathbf{r}} = \ddot{r_1}\mathbf{e_1} + \ddot{r_2}\mathbf{e_2} + \ddot{r_3}\mathbf{e_3}$$

Each agent with a given number i interacts with all other agents, targets and obstacles at each moment of time. The Euclidean distance between agents and obstacles or targets is used to compute an interaction vector.

$$d_{ij}^{mt} = \|\mathbf{r}_i - \mathbf{T}_j\| = \sqrt{(r_{i1} - T_{j1})^2 + (r_{i2} - T_{j2})^2 + (r_{i3} - T_{j3})^2}$$

where d^{mt} is the euclidean distance,r_i is the position of the agent,T_j is the position of the Target

The unit vector between r_ii and T_j is given by:

$$\mathbf{n}_{ij}^{mt} = \frac{\mathbf{T}_j - \mathbf{r}_i}{\|\mathbf{T}_j - \mathbf{r}_i\|}$$

The interaction vector between agent and target is,

$$\hat{\mathbf{n}}_{ij}^{mt} = (w_{t1}e^{-a_1 d_{ij}^{mt}} - w_{t2}e^{-a_2 d_{ij}^{mt}})\mathbf{n}_{ij}^{mt}$$

The total interaction vector between agent i and all targets is the sum of all interaction vectors

$$\mathbf{N_i^{mt}} = \sum_{j=1}^{N_t} \hat{\mathbf{n}}_{ij}^{mt}$$

Likewise, the interaction vector between all the other objects is computed. A total interaction vector is computed with weights that are determined in the genetic algorithm.

$$\mathbf{N_i^{tot}} = W_{mt}\mathbf{N}_i^{mt} + W_{mo}\mathbf{N}_i^{mo} + W_{mm}\mathbf{N}_i^{mm}$$

Finally, the propulsive force direction n_i^* is given by:

$$\mathbf{n}_i^* = \frac{\mathbf{N}_i^{tot}}{\|\mathbf{N}_i^{tot}\|}$$

Each agent will follow Newton's second law of motion

$$m_i \mathbf{a}_i = \Psi_i^{tot}$$

where Ψ_i^{tot} is the total force vector which consist of propulsion force and drag force. The component of the force is defined by

$$\mathbf{F}_{p,i} = F_{p,i} \mathbf{n}_i^*$$

$$\mathbf{F}_{d,i} = \frac{1}{2} \rho_a C_{d,i} A_i \|\mathbf{v}_a - \mathbf{v}_i\| (\mathbf{v}_a - \mathbf{v}_i)$$

Terminal Velocity occur when the net force is equal to zero this will make the $\mathbf{F}_{p,i} = \mathbf{F}_{d,i}$ so that the terminal velocity can be calculated since v_a is given to be zero:

$$\frac{1}{2} \rho_a C_{d,i} A_i \mathbf{v}_i^2 = F_{p,i} \mathbf{n}_i^*$$

$$\mathbf{v}_i = \sqrt{\frac{F_{p,i} \mathbf{n}_i^*}{\frac{1}{2} \rho_a C_{d,i} A_i}}$$

Forward Euler time discretization is used to integrate instead of symbolic or numerical integration to save computational cost because it does not involves any equation solving but it requires a small Δt to function properly. Moreover, the error of using Forward Euler method is essentially proportional to the step size, if we want to halve our error we must double the number of steps taken across the same interval.

Advantage and Disadvantage of using large Δt is it will save computational cost but using large Δt will cause the accuracy to decrease

Point-mass Idealization will be an accurate assumption if the size is reasonably small and the size does not cause the agents to crash into the objects. If actual sizes are considered, then the objects might crash at a slightly different point but not too different from our simulations. Since the UAVs are idealized as point masses the effects of their rotation with respect to their mass center is considered unimportant to their overall motion. Therefore, the balance of linear momentum and angular momentum is ignored

Gravity, Lift or Buoyancy is ignored to make the problem less complicated. Neglecting these forces makes our model faster while not really compromising the integrity of the model. Including these forces would not theoretically be too complicated and do not lead to a huge effect however, they do increase the run time of the algorithm

3 Procedure and Methods

Cost Function is required to run the genetic algorithm that is made in Project 1. The cost function will take into account crashes, speed of mapping and number of targets mapped by the following equation:

$$\Pi = w_1 M^* + w_2 T^* + w_3 L^*$$

$$M^* = \frac{\text{Unmapped targets}}{\text{Total targets}}, \qquad T^* = \frac{\text{Used time}}{\text{Total time}}, \qquad L^* = \frac{\text{Crashed agents}}{\text{Total agents}}$$

where,

$$w_1 = 70, \qquad w_2 = 10, \qquad w_3 = 20$$

The weight reflect the relative importance of each term. Meanwhile, the design string that brings to the cost function contain 15 random undetermined constant lie in the interval of $0 \leq \mathbf{\Lambda} \leq 2$

$$\mathbf{\Lambda}^i = \{W_{mt}, W_{mo}, W_{mm}, w_{t1}, w_{t2}, w_{o1}, w_{o2}, w_{m1}, w_{m2}, a_1, a_2, b_1, b_2, c_1, c_2\}$$

Gradient based-method will be poorly suited to optimizing the Cost Function because it will involves a lot of computational cost especially to differentiate the equations involving the 15 variables. The function is also non-convex, non-differentiable or discontinuous which is impossible to use Gradient based-method. GAs are very good at finding a large set of good solutions even with high dimensional 15 spaced because they focus their attention around known good solutions, but still there is an element of randomness.

Control Model that is used with parameters that can be optimized via genetic algorithm. A good set of control parameters will enable agents to map targets quickly and completely without crashes. We expect it to work for small positive values from 0 to 2. The control model that is used will only control the direction of the thrust force but we did not control the magnitude. The interaction vector that is being control also have a good characteristic for this project because it is an exponentially decreasing functions that are very smooth and decrease rapidly.

If a,b or c Design Variables Became Negative it would be bad for the performance of the swarm. For example, by looking at interaction vector between an agent and a target formula:

$$\hat{\mathbf{n}}_{ij}^{mt} = (w_{t1}e^{-a_1 d_{ij}^{mt}} - w_{t2}e^{-a_2 d_{ij}^{mt}})\mathbf{n}_{ij}^{mt}$$

if a became negative $\hat{\mathbf{n}}_{ij}^{mt}$ will became an exponentially increasing function of d_{ij}. The interaction vector will tend to increase rapidly and explode at high value of d_{ij}.

Strategy for “removing” mapped targets and crashed agents is by setting them equal to ’inf’ if the euclidean distance is smaller than the given range and skip through the ’if’ statement to avoid applying net force with ’inf’. And every time step loop, the drones or target that is flagged with ’inf’ is removed to decrease the run-time. The following is the code snippets:

Listing 1: Removing target after mapped

```
    % inside the loop of each agents with each target
    if dist <= agent_sight
        target = [ inf, inf, inf];
        Tar_loc(d,:) = target;
        continue
    end
```

Listing 2: Removing crashed agents

```
    % inside the loop of each agents with each other
    if dism <= crash_range
        %removing both agents after crashed
        drone2 = [ inf, inf, inf];
        drone  = [ inf, inf, inf];
        if e>=c
            Dro_dum(e+1,:) = drone2;
        else
            Dro_dum(e,:) = drone2;
        end
        continue
    end
```

Listing 3: Removing flagged drones and targets

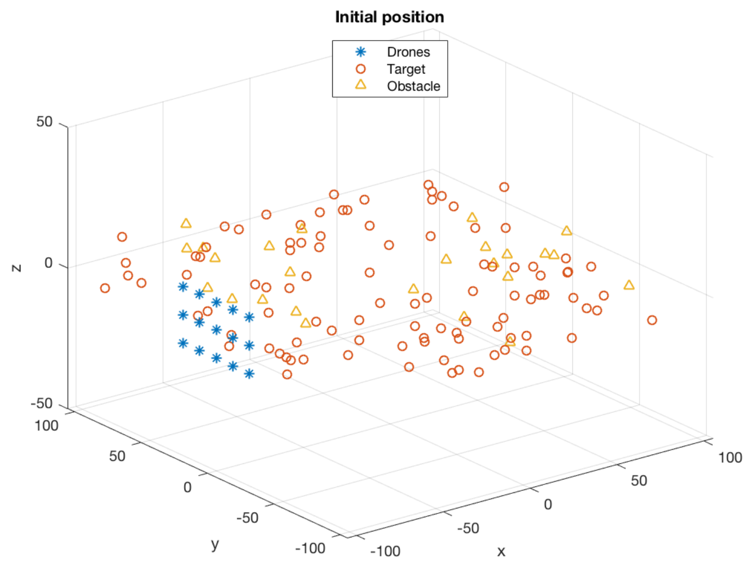
```
    % inside the loop of each time-step
    Dro_loc = Dro_dum;

    [dri,~] = find(~isfinite(Dro_loc));
    [tri,~] = find(~isfinite(Tar_loc));

    Dro_loc(dri,:) = [];
    Tar_loc(tri,:) = [];
```

Initial Location of the agents is placed thoughtfully (not randomly) so that the drones are not immediately crashing to each other and to make sure the drones are can mapped the targets as efficiently as possible. The following is the code snippet used to arrange the drone:

Listing 4: Initial position of drone

```
    Dro_loc = [-100+zeros(1,15);
    linspace(-25,25,5),linspace(-25,25,5),linspace
        (-25,25,5);
    -10+zeros(1,5),zeros(1,5),10+zeros(1,5)]';
```

The figure below shows the initial location of the drone in the 3D space with the obstacles and the targets



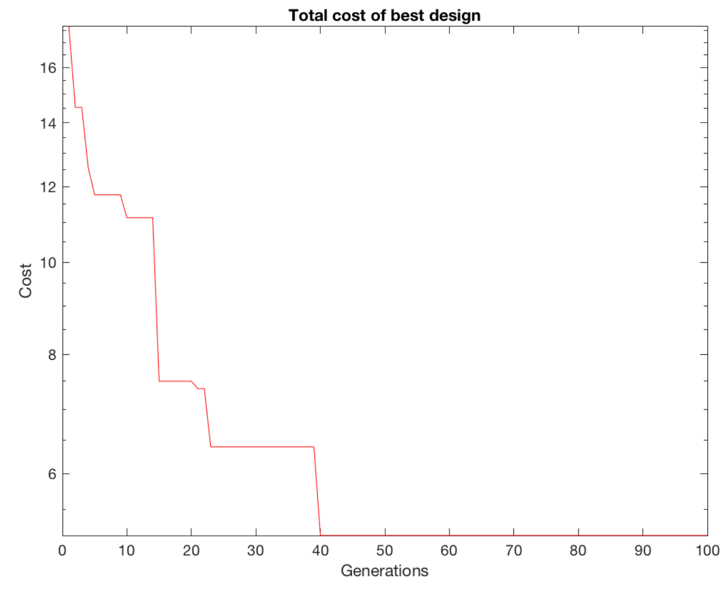
Significant Choices that I made in creating the code is increasing the drag by a factor of 5. At first, the drones were not able to map all the targets in the period of 60s. When looking at the animation, the drones overshoot the target making a spiral around it instead of mapping them. By increasing the drag, the drones moves slower decreasing the chance of overshoot.

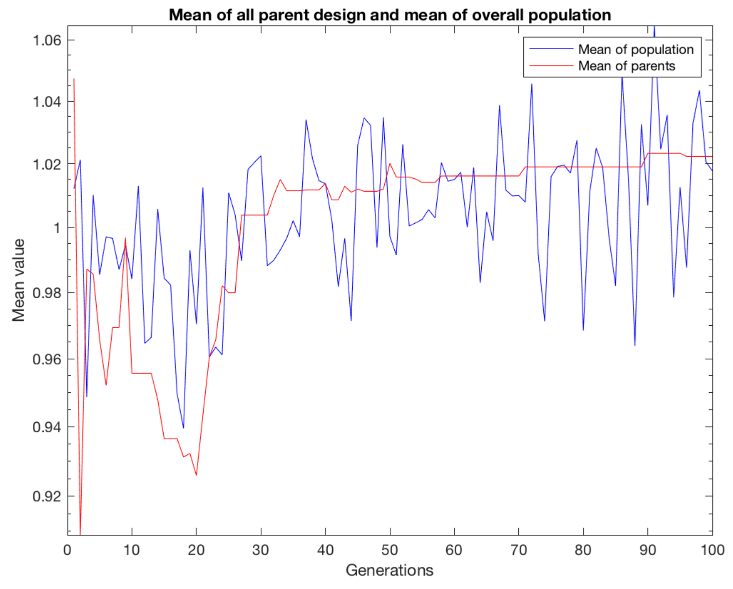
Listing 5: Increase drag

```
    thrustf = Fpi_s.*nstar;
    dragforce = 5*0.5*rho_a*Cdi*Ai*norm(-vel(:,c)).*-
        vel(:,c);
    % multiply 5 to drag force
    netforce = thrustf+dragforce;
```

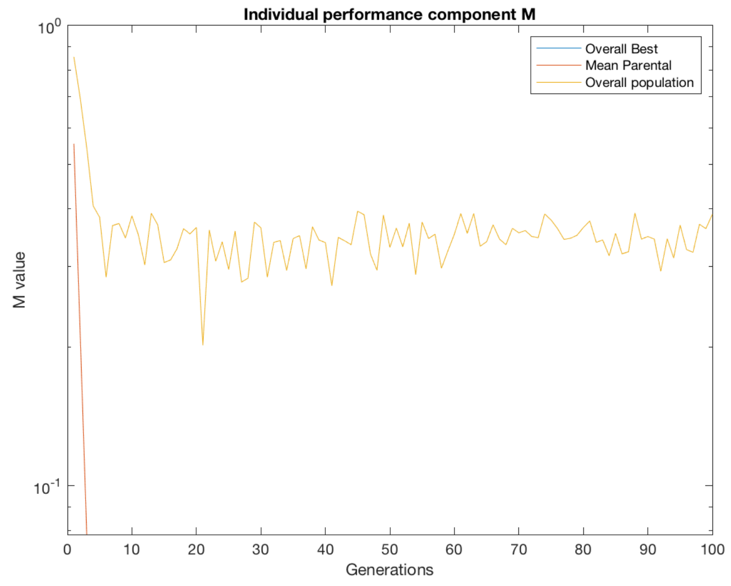
4 Procedure and Methods

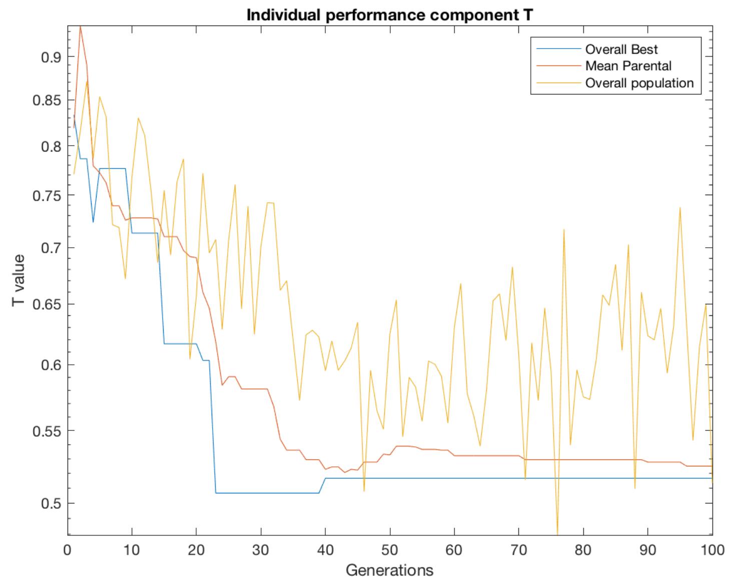
Convergence Plot showing the total cost of the best design, the mean of all parent designs, and the mean of the overall population for each generation is shown below:

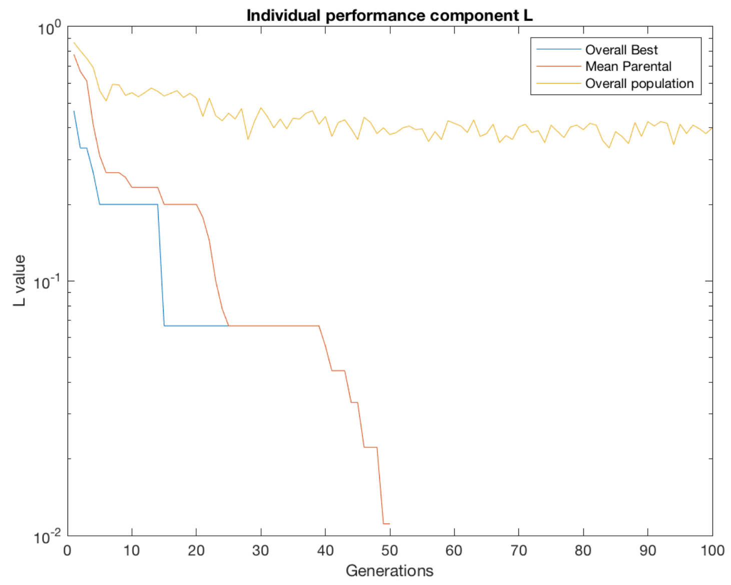




Individual Performance components M,T and L ,for the overall best performer, mean parental population, and overall population is shown below:







From the figure it can be observed that all the individual performance converging to zero except for the T value. This shows that the minimum cost value mostly come from T which is the the fraction of use time over the total time. This is true considering the speed limitation of the drone to mapped all the target as fast as possible. From the cost function we also include the lowest weight on time since it should have the lowest priorities.

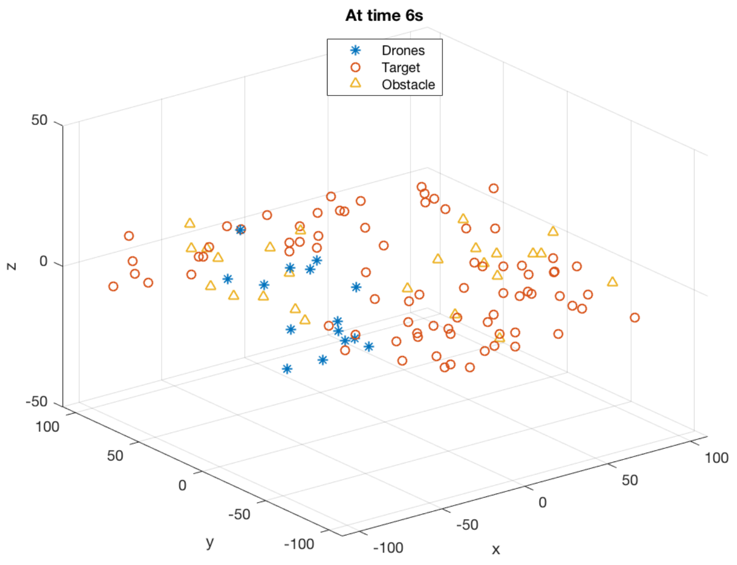
Best-performing 4 Designs is reported on the table below:

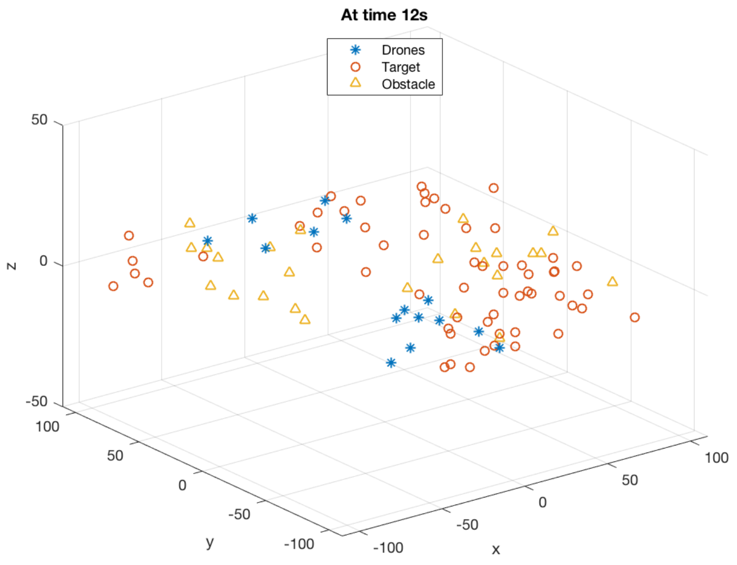
Design	Λ_1	Λ_2	Λ_3	Λ_4	Λ_5	Λ_6	Λ_7	Λ_8	Λ_9	Λ_{10}	Λ_{11}	Λ_{12}	Λ_{13}	Λ_{14}	Λ_{15}	Π
1	1.11	1.18	1.25	1.24	1.21	0.83	1.27	1.32	1.03	0.57	0.15	0.80	1.81	0.31	1.47	5.17
2	1.12	1.13	1.28	1.26	1.21	0.85	1.27	1.23	0.80	0.51	0.15	0.80	1.82	0.31	1.46	5.20
3	1.11	1.14	1.27	1.24	1.21	0.84	1.27	1.26	1.00	0.52	0.15	0.79	1.82	0.31	1.46	5.23
4	1.12	1.15	1.25	1.26	1.21	0.84	1.27	1.24	0.82	0.56	0.15	0.80	1.82	0.31	1.47	5.23

Table 1: Best 4 Designs

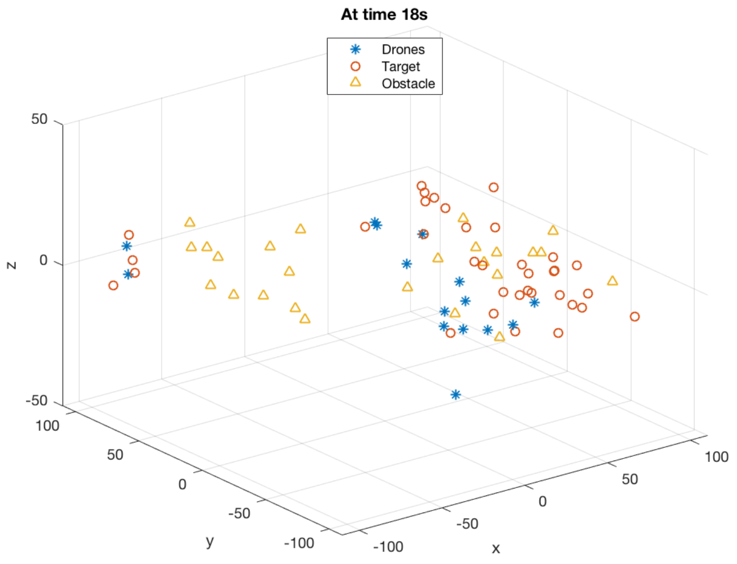
The Best-performing Designs have a really small variation between each parameters and they seems to be converging around their best design values. There is no parameters that have negative values which might be caused by our setup of random parameters interval that is between 0 to 2. Most of the parameters are far from zero and we can see that from the plots previously shown that the mean of the parents design is converging to a value around 1. The smallest parameters that is closest to zero is the parameter no 15 which belongs to c_1. c_1 is the variable that control the attraction between an agent with other agents which needed to be small.

Series of plots showing how the best design moves the agents is shown below

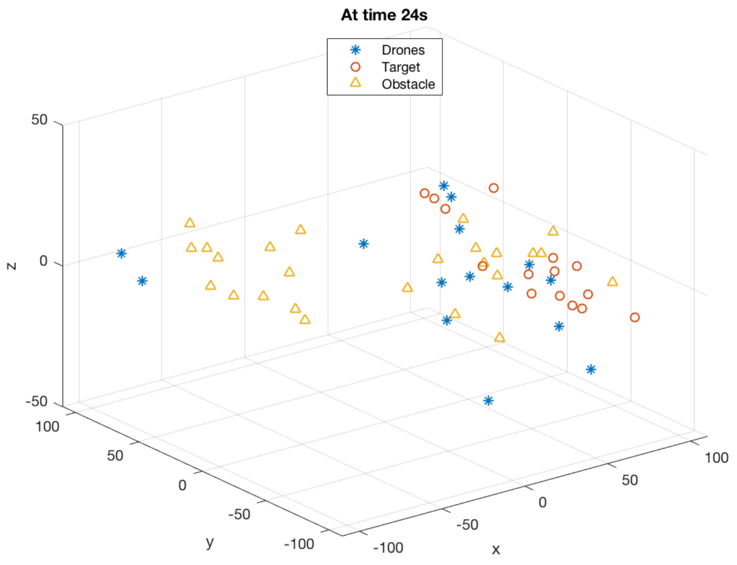


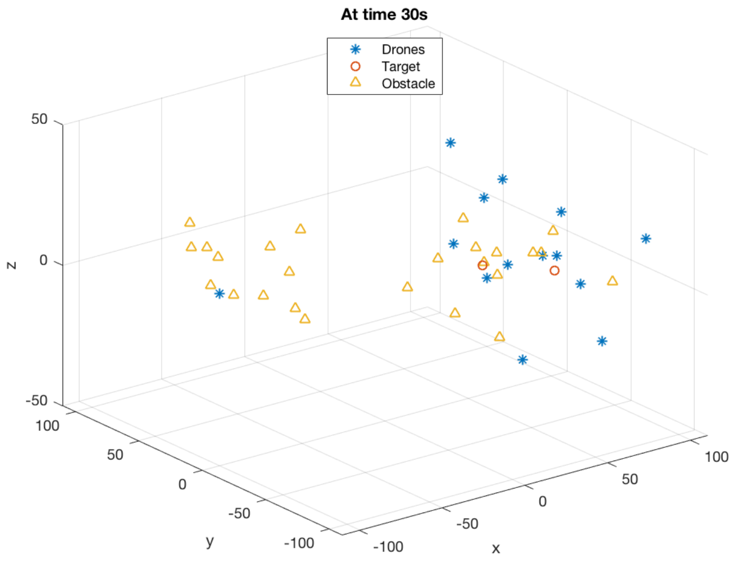


At time 12s
Drones
Target
Obstacle
50
0
-50
z
100
50
0
-50
-100
y
-100
-50
0
50
100
x



At time 18s
Drones
Target
Obstacle
50
0
-50
z
100
50
0
-50
-100
y
-100
-50
0
50
100
x


At time 24s
Drones
Target
Obstacle
50
0
-50
z
100
50
0
-50
-100
y
-100
-50
0
50
100
x


At time 30s
Drones
Target
Obstacle
50
0
-50
z
100
50
0
-50
-100
y
-100
-50
0
50
100
x

5 Conclusion

The aim of the project which is to generate a model of a terrain needing to be mapped is successfully achieved. The series of plots shows that the model created successfully mapped all the targets in the least amount of time without any crashes. The cost function is minimized using the genetic algorithm and the optimal solution is computed.