Final Assignment B

Tracking and enclosing a target with perfect sensing

Martin Smed Metze - mamet15
Nicolaj Haarhøj Malle - nmall14
Thor Mørup Fischer - thfis15
Valthor Bjarki Gudmundsson - vaguu15

I. Introduction

Consider three rotorcraft Q_1, Q_2 and Q_3, and a moving target Q_t at a constant altitude $t_z > 0$ with velocity $\dot{p}_t = \begin{bmatrix} v_x & v_y & 0 \end{bmatrix}$. All our vehicles Q_1, Q_2 and Q_3 know the altitude of the target, and can measure their relative positions w.r.t the target, i.e., ${}^N(p_{\{1,2,3\}} - p_t)$, and their relative velocities ${}^N\dot{p}_{\{1,2,3\}} - {}^N\dot{p}_t$. Please, design an algorithm such that Q_1, Q_2 and Q_3 describe a circumference around Q_t, and Q_1, Q_2 and Q_3 can control their intervehicle angle in the circumference. The angle of Q_1 in the circle is just $atan2({}^N p_y, {}^N p_x)$, and the interangle is the subtraction of two angles coming from two vehicles, i.e., we control the relative angle between vehicles in the circle.

- This is a path tracking problem where the center of the circle is moving with a constant velocity.
- A reference on how to control the position of a team of vehicles on a circle [1]. It is just a consensus algorithm on the intervehicle angle of the circumference.

As a further objective, the implemented algorithms must be able to handle velocities along the gradient of the circle, essentially making the drones in the formation move in a circular pattern around the target drone.

In addition to the above tasks, this rapport explores the possibility of adding a number of drones to the configuration during simulation runtime. The guidance algorithms employed to solve the tasks must therefore be robust to sudden changes in the drone formation.

II. Guidance

A. Circle

The drones follow the target drone in a circle as described by the assignment. This is done by defining the circle function with radius r and Q_t's xy-coordinates as center coordinates. The function is seen in equation 1. This approach is inspired by the work in [2] for fixed wing UAVs where the guidance vector field makes the UAV orbit around its target as can be seen on figure 1.

$$f(x, y, r) = ((x - x_t)^2 + (y - y_t)^2 - r^2) := e_c \tag{1}$$

The difference between the approach in figure 1 and the approach used for this assignment is the use of quadcopters in simulation, and thus orbiting is not necessary. Placing Q_N drones around Q_t on the periphery of the circle is sufficient.

Equation 1 can be used to define level sets around Q_t. With a specific radius, the zero level set can be used to find the error vector towards that set. To find the normal vector pointing away from the zero level set, we use the gradient of equation 1. The gradient is found by first differentiating with respect to x and the again to y, see equation 2.

$$\nabla f(x, y) = \begin{bmatrix} 2 * (x - x_t) \\ 2 * (y - y_t) \end{bmatrix} \tag{2}$$

The error response contribution of the circle control thus becomes:

$$U_{circle} = K_c * (-e_c) * \nabla f(x, y) \tag{3}$$

As the results will indicate in section III, these errors are not Euclidean distances as they can be negative.

Even though orbiting around the circle is not necessary it is still implemented, however in all tests presented the gain for this response is set really low. One could simply add the orbiting by increasing this gain. The orbiting response is calculated by rotating the gradient in equation 2 to get the tangent. This is shown in equation 4.

$$U_{tangent} = K_t * \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \cdot \frac{\nabla f(x, y)}{||\nabla f(x, y)||} \tag{4}$$

Now add the response from the circle and the tangent and an orbiting response is made, see figure 5

$$U_{orbit} = U_{circle} + U_{tangent} \tag{5}$$

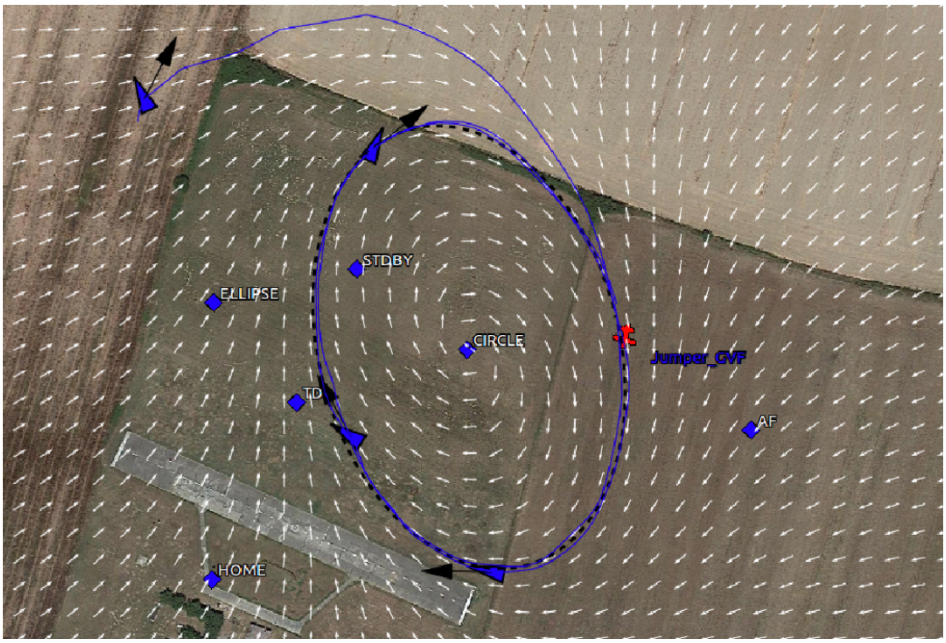

Fig. 1: An illustration of the guidance vector field in [2]

B. Formation

As assignment b states, the UAV's are supposed to use their interangle as control inputs. We take the liberty of processing this angle into a more intuitive value in meters, making sure the UAVs place themselves in the correct place with respect to their neighbours.

Firstly, the equal angle between the amount of drones is found using the chord of a circle. An example with 3 drones is $\frac{360}{3} = 120$ degrees between the unit vectors going from Q_t to all 3 drones.

The chord length each drone needs to maintain to each two of its neighbors can then be calculated as $r * sin(angle)$, where r is the radius of the formation to the target drone.

A fault with this formation strategy can occur when using more than 3 UAVs, as overlapping can occur. A simple example of overlapping is when 4 UAVs in a formation do not expand a geometric square but rather an L or simply a line, as visualized on figure 3

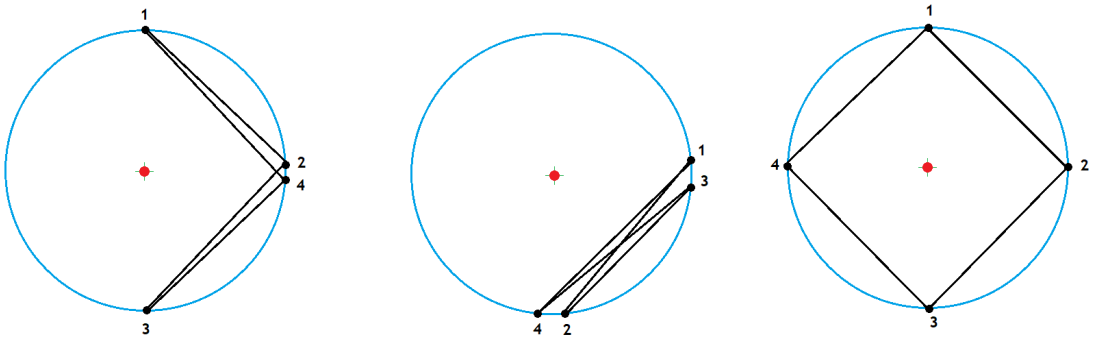

Fig. 2: Three different possible formation outcomes of the four drones. Going from left to right, the first two cases are actually just as correct in terms of the error response as all of the drones are on the circle, and each of them have the correct angle to Q_t and distance to its neighbours. The goal is creating a formation with equal angles and distances between all drones.

To circumvent this problem while still retaining the distributed way the system currently works with the neighbour approach where each Q_n only has two neighbours to communicate with, the angle between the vectors going from a drone towards its neighbours is used. In the cases described in figure 3, the angle between the lines going from Q_1 to Q_2 and Q_1 to Q_4 is very small, where the desired angle is actually 90°.

This problem is solved by switching the the Q in question with one of its neighbours. The current approach is switching Q_n with Q_{n+1}. This rearrangement of the formation is an attempt to create the correct formation between all Q's in the

system, and not necessarily the correct solution in the first rearrangement attempt.

C. Control scheme

Each drone has multiple contributions to its velocity controller's response, based on the error signals:

1) its implicit path to the desired radius/circle formation error: e_c
2) two chord length error terms (neighbor drones distance errors) e_c
3) a small tangent contribution (move around in circle) e_c

These guidance signals are multiplied by a unique proportional gain each, one for the correction of the formation error, one for the circle distance correction and one for the tangent scale. The sum of these intermediate responses are fed through a sigmoid function. This sigmoid function is tuned to cause a higher response per unit of "intermediate" response at low levels and enforces an upper limit on the response level. The upper response level also serves the secondary purpose of limiting how high the set velocities of the drones can get.

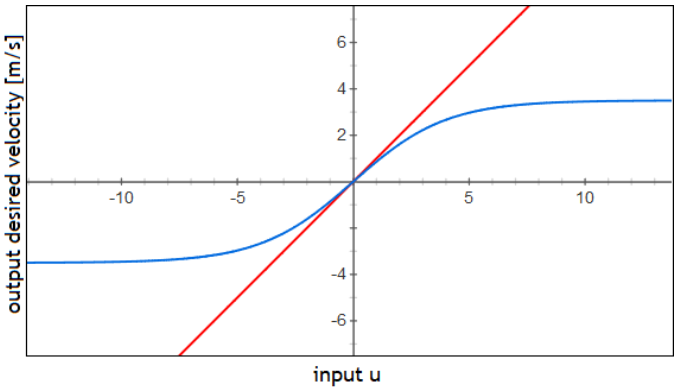

Fig. 3: The sigmoid function output (blue) and input (red) used with a range of 7m/s (+/- 3.5m/s) and a curvature of 0.5. This result in rather linear like desired velocity at low velocities, but as the response u gets higher, the desired velocity exponentially converges on the upper limit of 3.5m/s

The final target velocity is calculated as:

$$v_t = \frac{7}{(1 + e^{(-0.5*u)}} - 3.5 \tag{6}$$

For the purpose of this assignment the Lyapunov velocity controller provided by Hector Garcia from [1] is used to control the velocities of each drone. The Lyapunov controller is given the response of the sigmoid function as the desired/target velocity.

D. Velocity

Since the velocity of Q_t is known to all Q_n in the system, they can simply add this velocity to their own. This way, they will follow Q_t's velocity precisely, but maybe a bit unrealistically since this might result in very high velocities of the following Q_n's. Since the velocity of Q_t is constant and relatively low, it is not a concern for this assignment.

III. Simulation

A. Tests

Several different tests have been performed to evaluate the performance of the system:

1) Three drones hovering around Q_t with zero velocity.
2) Three drones hovering around Q_t where a fourth drone is added at t=100s.
3) Three drones following Q_t while flying with a constant velocity and trying to adjust the formation during flight.
4) Four drones following Q_t while flying with a constant velocity and trying to adjust the formation during the flight.
5) Four drones' path taken when tracking a moving target

B. Results

1) : The aim of this test is to verify the formation and circle guidance and thus Q_t will not be moving. As it is for all further tests, Q_t starts at xy-position (0,0) and all remaining drones will spawn uniformly random between 0 and 1 in x and y. The remaining drones will try their best to place themselves on the perimeter of the circle and also go into formation.

[1] https://github.com/noether/pycopter

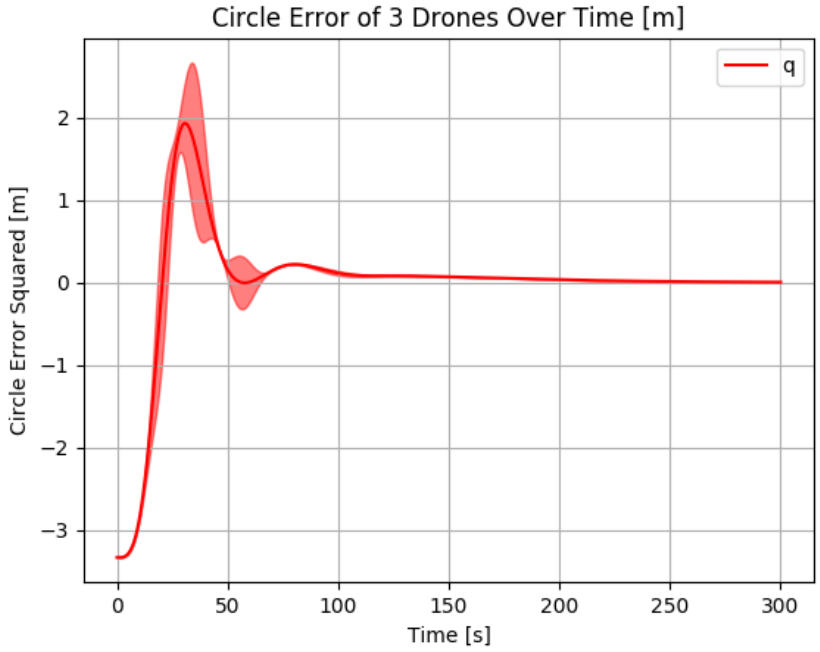

Fig. 4: A figure depicting the average value of all three drones' distance error e_c to the circle perimeter, with the target drone being stationary. The less intense shade of red is the variance of the formation error to the mean error based on 50 test runs

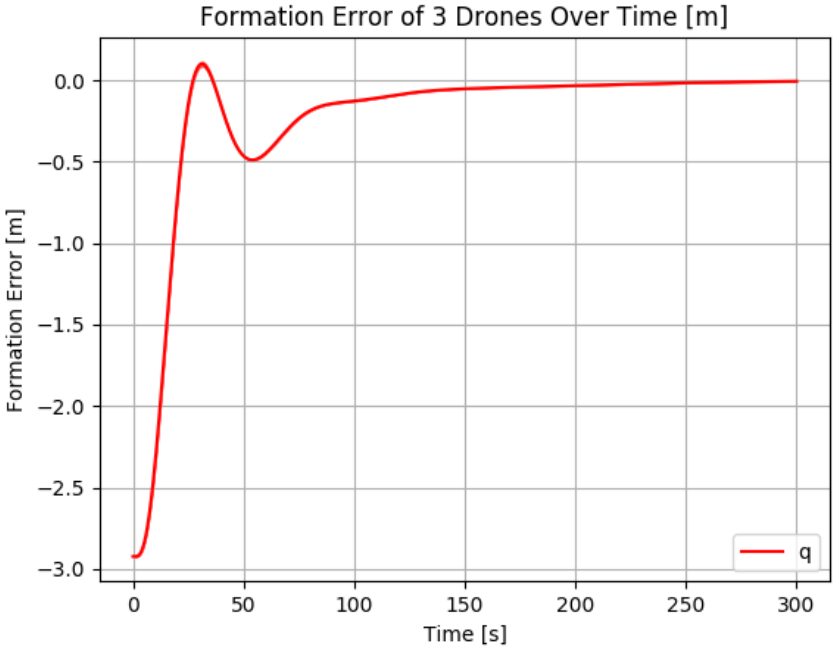

Fig. 5: The average value of the formation error e_f, with the target drone being stationary.

2) : In this test, the fourth drone is added in a random position with the same altitude and parameters as the other drones, at the 100 second mark. This drone will take its place as the fourth drone on the list, thus becoming neighbours with Q_3 and Q_1, breaking their bond. If this place does not suit Q_4 in terms of the minimum angle between neighbours, the places are swapped. It is very apparent on figure 7 that a neighbour swap occurs instantaneously after Q_4's addition to the fleet due to how the total formation error sharply decreases right after its sharp increase.

The second sharp change in formation error on figure 7 happens at t = 150s, 300s and 450s. This happens due to the

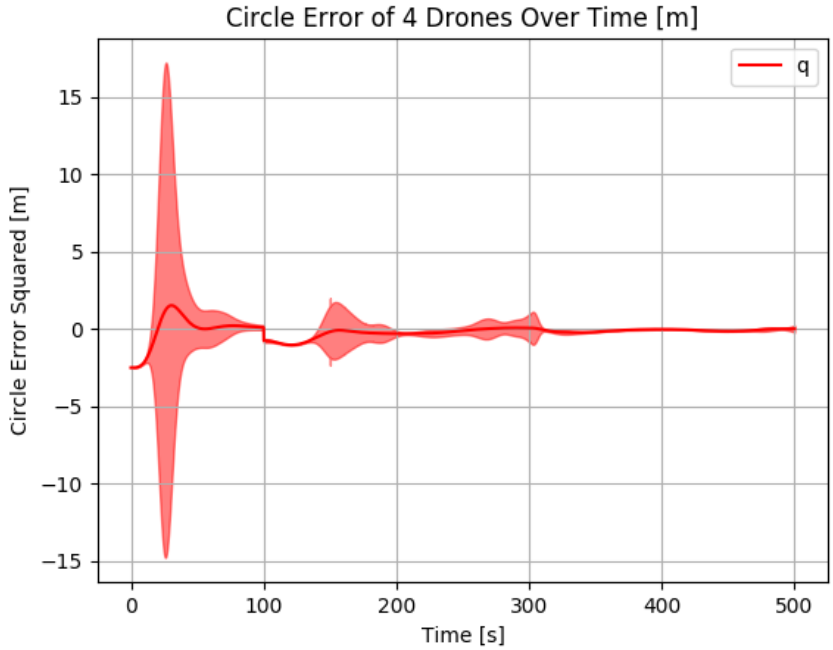

Fig. 6: The average value of the four drones' distance error e_c to the circle perimeter, with the target drone being stationary.

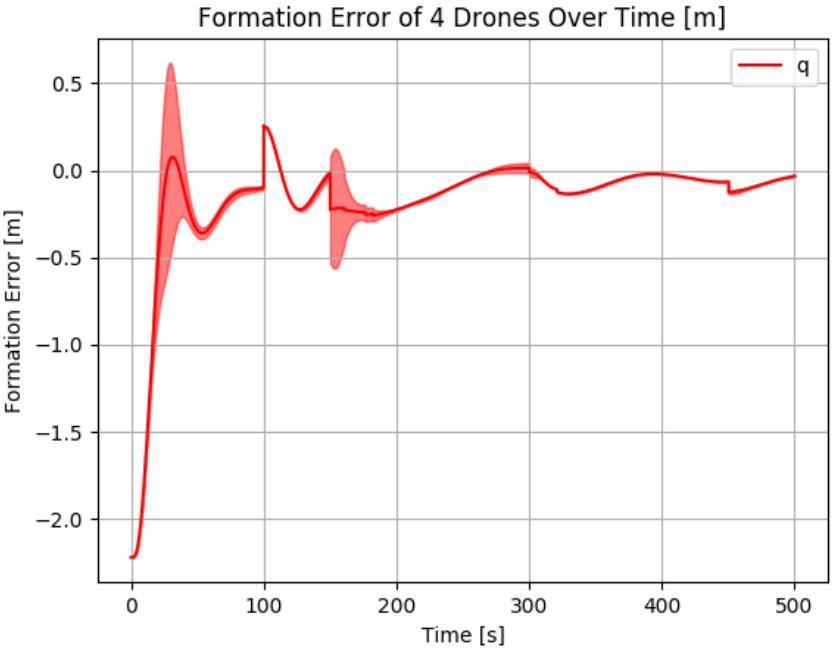

Fig. 7: The average value of the four drones' formation error e_f, with the target drone being stationary.

cooldown the Q's are given after a neighbour rearrangement which is set to 150 seconds when a drone spawns and if a drone has been a part of a rearrangement. This is to ensure that the fleet can converge to the correct formation before rearrangement can be initiated again as the convergence might be slow.

3) : This is a repetition of test 1 except now Q_t moves with a constant velocity of $\binom{0.25}{0.25}$, moving north-east with 3 drones in circular formation. Figures 8 and 9 depict the results from 50 simulations with the mean and variance of circle and formation error respectively. The results indicate no issues in the formation or circle errors and both converge to zero.

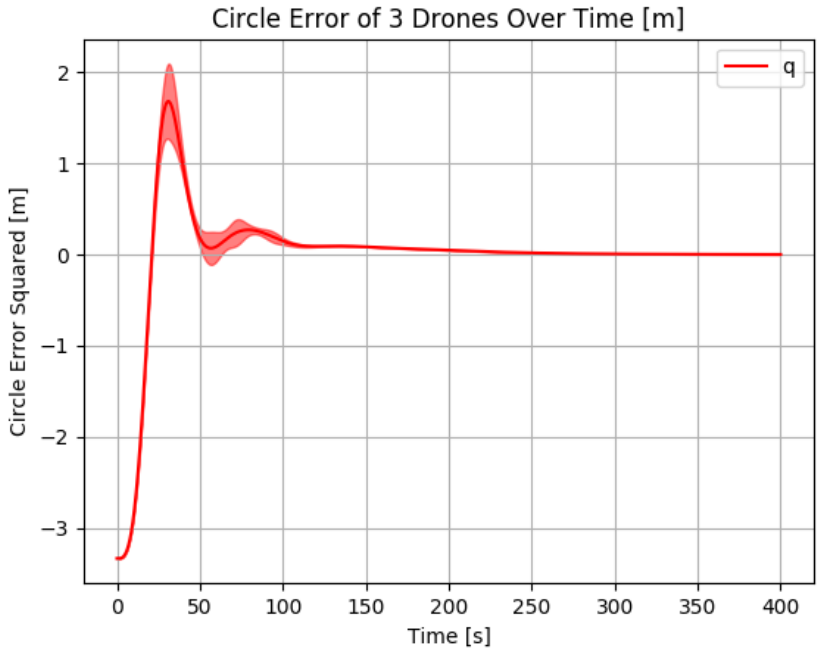

Fig. 8: The average value of the three drones' error e_c to the circle perimeter, with the target drone moving

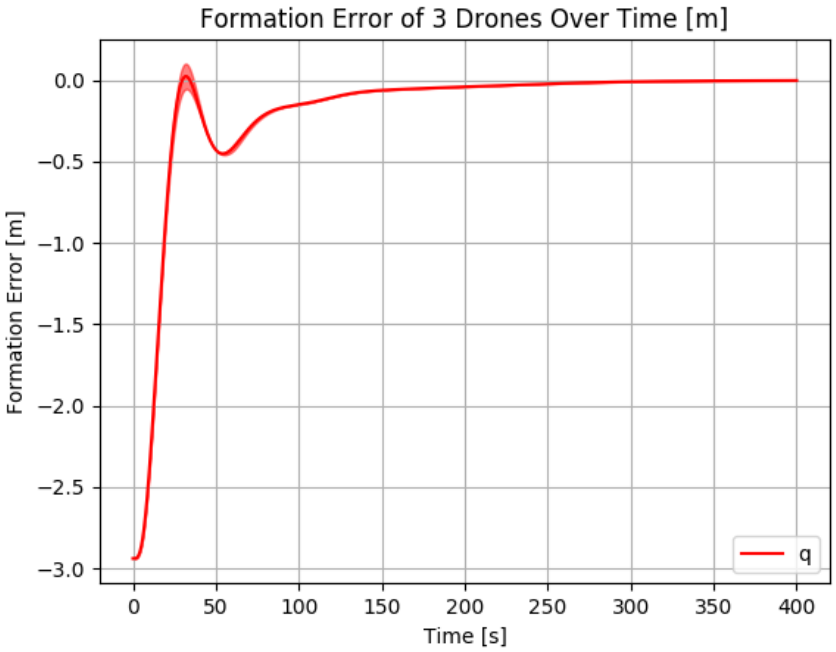

Fig. 9: The average value of the three drones' formation error e_f, with the target drone moving

4) : More interestingly, these tests regard simulations where four drones are following Q_t while flying with the same constant velocity as the previous test. The four follower-drones are all present from t=0. The addition of the fourth drone at t=100 more often than not resulted in crashes due to huge error-signals for the fourth drone, spawning quite far from the formation.

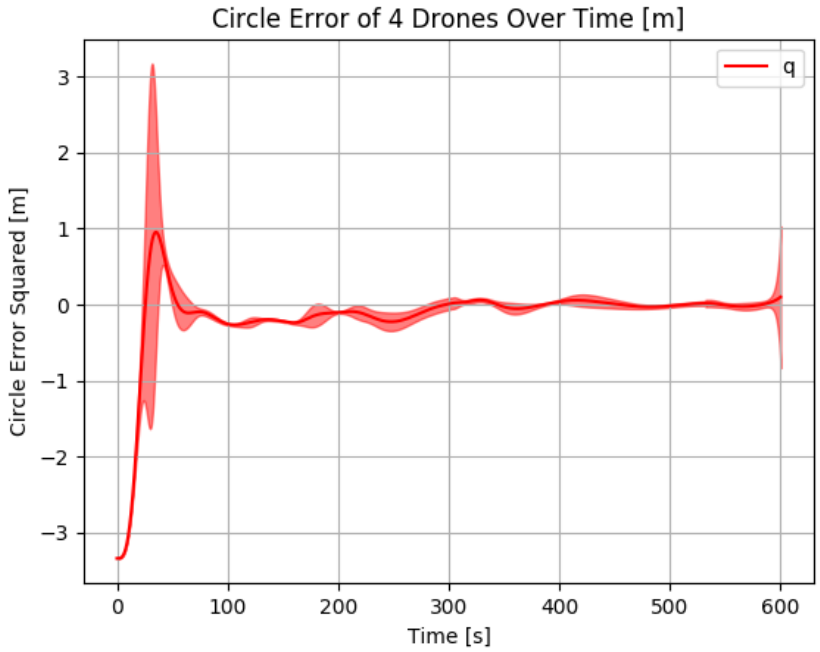

Fig. 10: The average value of four drones' error e_c to the circle perimeter, with the target drone moving

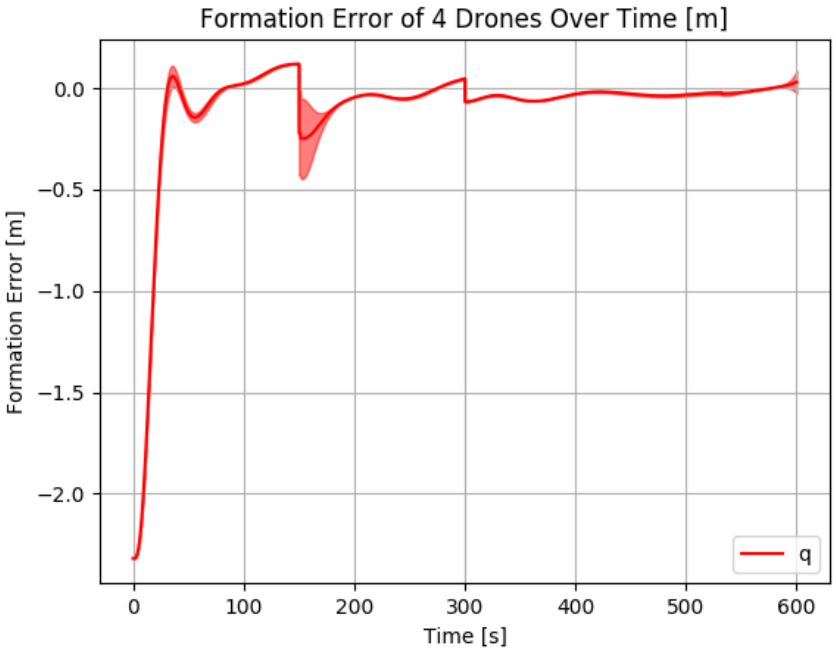

Fig. 11: The average value of the four drones' formation error e_f, with the target drone moving

Compared to the previous test, the chance of the fleet being in an incorrect formation is enough to often prompt a neighbour rearrangement at t=150s and 300s. It seems that after two rearrangements, the fleet has found the correct formation with both the mean and variance converging to zero.

5) : Visualization of the paths taken by four drones with a moving target and an in formation rotation around the target.

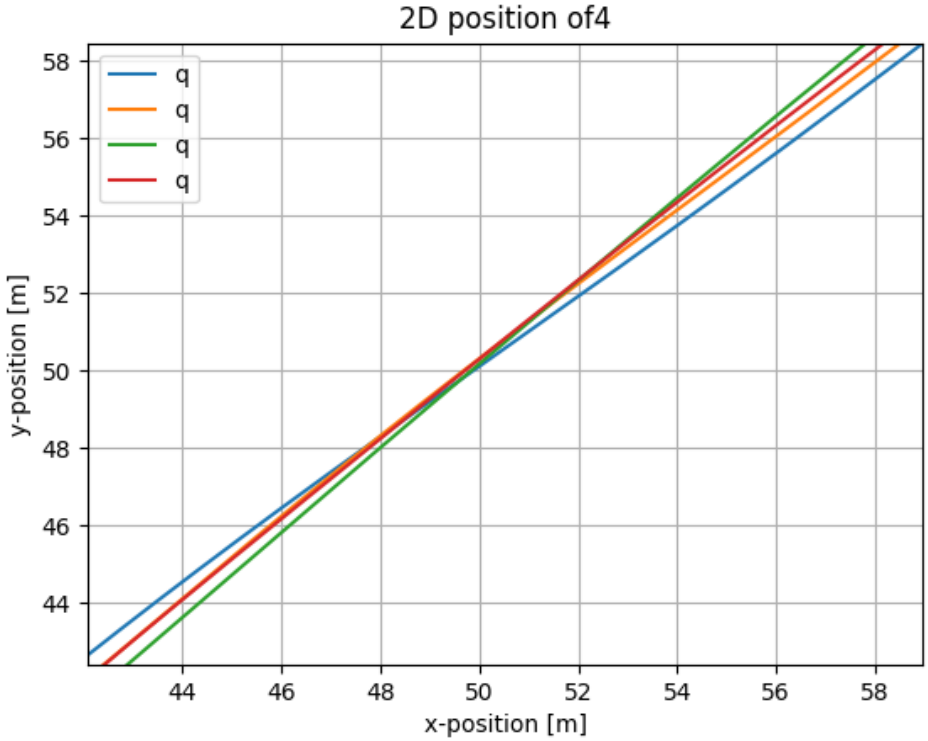

Fig. 12: zoom in on section of path taken by four tracking drones

The line crossings are due to the in formation rotation around the target drone.

IV. Discussion

Currently, the difference between a moving target Q_t and a still target is minimal with respect to the circle and formation error.

The current approach of assuming perfect sensor information with no time delay is rather unrealistic. To make the simulation better resemble reality, imperfection in sensing with a time delay could be introduced. This would introduce the need of a navigation system to estimate on a per drone basis the target drone's velocity and position as well as their own position. This would in turn also make the tracking problem more difficult as the error could generally be expected to be higher.

The proposed solution to overlapping by switching neighbours based off an angle between neighbours seems like a good solution for smaller formations like 4 drones, but does not scale well with a larger amount of drones. The drawback of this method is also the guesswork of it. It is not guaranteed to find the correct formation with one rearrangement of neighbours, and not even after the second one. This might go on for a while until the correct formation has been reached (in particular for high amounts of drones).

The system has been tested with formations of 5 and 6 drones and almost every test results in a crash. There currently is a fault with the formation guidance signal that causes very high signals resulting in crashes and runaways from the circle, especially with higher number of drones. Debugging this problem was unsuccessful and cases with 4 drones were prioritized.

V. Conclusion

A distributed guidance algorithm of a fleet of more than three drones around a moving target drone Q_t has been developed and tested in simulation. Results indicate good performance with up to four drones, being able to rearrange their places in the formation by communicating with their neighbours. Results of tests with more than 4 drones indicate sub-optimal performances as the number of possible formations sharply increases with the amount of drones in the system, resulting in repeated rearrangement attempts and divergence from zero error.

REFERENCES

[1] H. G. de Marina, Z. Sun, M. Bronz, and G. Hattenberger, "Circular formation control of fixed-wing uavs with constant speeds," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 5298–5303.

[2] H. G. de Marina, Y. A. Kapitanyuk, M. Bronz, G. Hattenberger, and M. Cao, "Guidance algorithm for smooth trajectory tracking of a fixed wing uav flying in wind flows," pp. 5740–5745, 2017.