Documentation
of the
Team Project
Swarm Intelligence

Cooperative Height Control

by Kanwal Jahan, Jawad Ahmad,
Fabian Witt and Steven Brandt

1. Introduction and task

This team project has the goal to develop a first swarm behavior of the finken copters. For this task a simple base configuration of the copters is used. With this configuration the copters are able to hold a given height autonomously but only with a statically coded height. Furthermore the four copters are fixed in a cage at the edges of a square with thin lines that they can only move up and down. This reduces the implementation of the swarm behavior to finding the correct height.

The mentioned swarm behavior stands for cooperative holding of the same height for all four copters. This means that the copters will take off and then they should find each other at a not directly specified height. Rather this is a searching process with only predefined values for the upper and lower height boundaries of the searching area. This searching process is the first basic approach explained in the *second chapter* where it is called oscillation.

The oscillation process is needed to search for other copters and if a copter found another one, he should hold this height. Finding another copter in this context means that the copter measures an obstacle with one of its four sonar sensors (angle of 90 degrees between each one) at a distance between 30 and 60 centimeters. Then the copter knows there is something near me and we assume that this is another copter. This is the next basic behavior of the copters which is defined in the *third chapter*.

Now one copter knows what to do alone but because of one copter is not a swarm we extend the process to all four copters in *chapter four*. Generally the same code is running on all four copters except on the decision which sonar sensors should be used to detect obstacles.

So far a random search process is implemented with holding the current height if another copter was found. If an already found copter is lost the random search process starts again. Since this is not a really cooperative behavior we developed a finite state machine to define what needs to be done if a found copter is lost. This finite state machine allows us to implement a real swarm behavior. This means that the copters that have found each other will search together for further copters, on the other hand if a copter gets lost the

remaining group should not break up rather it should search together for the lost one. This finite state machine is described in the *fifth chapter*.

Theoretically everything works now. What the copters do in real and if they converge to a real swarm is explained in *chapter six*.

Finally a conclusion of the hole development process is made in the last *chapter seven*. Furthermore there are mentioned some general problems and possible future work.

2. Oscillation

The oscillation process is the first basic behavior which is needed for the cooperative height control. Everything starts with a defined take off phase. In this phase the copter will start flying and after it leaves the ground it will fly upwards to 1 meter. After it reaches this height and stays there for 10 seconds, it will switch to the search phase.

In the search phase the copter will automatically oscillate between the upper and the lower search boundary as shown in figure 1. Since we use an infrared sensor for measuring the height we get only correct values for the height between 0.4 and 0.9 meters. That is the only reason why we have chosen this values for the boundaries.

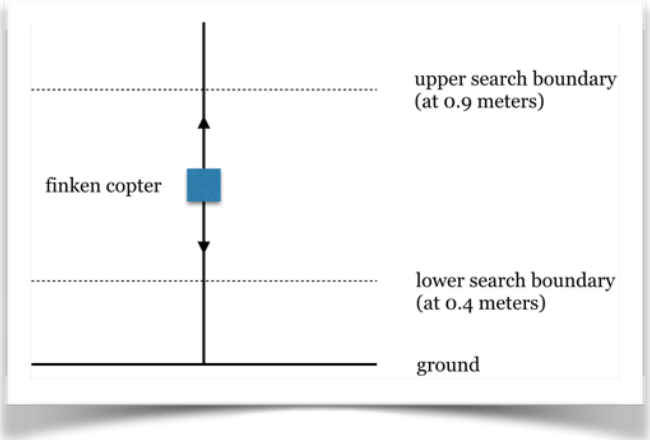

Figure 1: Oscillation process for one copter. (Own illustration.)

The function in which we have implemented the oscillation process is a periodic one. Therefore we have to care about how often the height increases or decreases. We can adjust how often this function would be called in every second with the frequency. The „search_neighbor“ variable indicates weather to oscillate or not. One step deeper the decision of moving upwards or downwards is made by the variable „go_down“. Finally if the upper or lower boundary is not reached we need to change the height by the variable „height_changing_rate“ which is manually defined to 0.005 meters. Combined with the frequency of the periodic function (5 Hz) it ends up to change the height 0.025 meters every second. If one boundary is reached the direction of searching is changed to the opposite.

In figure 2 you can see the code snippet where the oscillation process is programmed. The variable declarations and definitions are excluded since they are easy.

```
void finken_oscillating_model_periodic(void)
{
    if ( finken_oscillating_mode ) {

        if( search_neighbor ){

            // check weather go up or down
            if ( check_direction == true ) {

                // if we are above middle, go downwards and otherwise
                if ( finken_sensor_model.distance_z >= middle ) {
                    go_down = true;
                } else {
                    go_down = false;
                }
                check_direction = false;
            }

            if ( go_down ){
                if ( finken_system_set_point.distance_z > height_oscillating_down ){
                    finken_system_set_point.distance_z -= height_changing_rate;
                } else {
                    finken_system_set_point.distance_z = height_oscillating_down;
                    go_down = false;
                }

            } else {
                if ( finken_system_set_point.distance_z < height_oscillating_up ){
                    finken_system_set_point.distance_z += height_changing_rate;
                } else {
                    finken_system_set_point.distance_z = height_oscillating_up;
                    go_down = true;
                }
            }
        }
    }
}
```

Figure 2: Code of the oscillation process. (Own illustration.)

The only problem that occurs during this procedure was the infrared sensor. It measures only correct values in a really short range (between 0.4 and 0.9 meters). Fortunately it is enough to get a useful oscillation process. Thus the copters have a smaller searching area so they should converge faster but in future the sensors should be replaced by better ones.

3. Height holding of an obstacle

The second task for the copter is to detect its neighbors during the oscillating process. Sonar sensors look for any obstacle or object in a defined search space. Since all four copters are in known environment, which restricts the copters to stay at a defined distance from each other. So the search space for sonars to look for the other copter has been restricted to a total of 30 cm (Range: between 30 cm to 60 cm).

During the vertical oscillation the copter reads its sonars with a frequency of 12 Hz, if they have detected any object in the defined horizontal search space. When the sonar reading is positive the copter stops oscillating and tends to stay at its position maintaining the same height as its neighbor. If the reading is negative it keeps oscillating vertically.

Once readings of one sonar are positive and a neighbor is detected, readings to be positive from another sonar sensor are checked with the same frequency (12HZ) in order to find the second neighbor too.

3.1. Issue and fix

Previously readings were read from the sonar sensor at a frequency of 30 HZ that is after every 3.33 mSec. This rate was too fast for the sonar sensor to process. The frequency has been reduced to 12 Hz and readings are taken after every 83 mSec.

4. Extension to four copters

The same algorithm has been run on each copter. So the approach used is, to achieve the cooperative height control while every individual copter is on range defined intelligent search. During this search every single copter is trying to achieve its local goals which brings out the emergence of swarm behavior where all four copters stay at the same height.

There are the following goals for each copter to consider. First is to oscillate vertically and read enabled sonars after every 83 milliseconds. Once one sonar reads the obstacle, it is considered as a neighbor due to our controlled environment. This shows two copters have found each other and are expected to maintain the connection. This communication is explained in the Chapter 5. The next thing to do is, to keep reading the sonars until second obstacle/neighbor is detected, too. Once each copter has found two neighbors it should maintain its position, so all copters end up holding the same height.

4.1. Issue and fix

There are four sonar sensors on each copter mounted at a right angle to each other. To run the process faster only two sonar sensors are read since the local goal is to find two neighbors at maximum while the other two are disabled. Which sonar is enabled on which copter depends on the installation of the copters in the test cage.

5. Finite state machine

We used a finite state machine to represent the behavior of the copters. This defines the different state of the copters. It is important for the later swarm behavior that although everyone uses the same definition of machine, but the machine runs independently on each copter. All possible states and the corresponding transitions were modeled.

As a result, three states were defined:

- State 0: no copter found
- State 1: one copter found
- State 2: two copters found

Due to the given environment, each copter can only find two other copters. At each state two variables are considered:

- search_neighbor
- check_direction

The variable search_neighbor indicates whether the copter continue searching for neighbors (search_neighbor equal to „true“), or whether to keep the current position (search_neighbor equal to „false“).

For a joint search without communication, the flight direction must be determined. If the variable check_direction is „true“ the current position will be measured compared to the middle of the search space. If the copter is above the middle, it searches downwards and if it is below the middle, it will search upwards. This ensures that two copters will fly in the same direction.

With this assumption, there are thus nine state transitions. The following graph shows the finite state machine with its three states and nine state transitions.

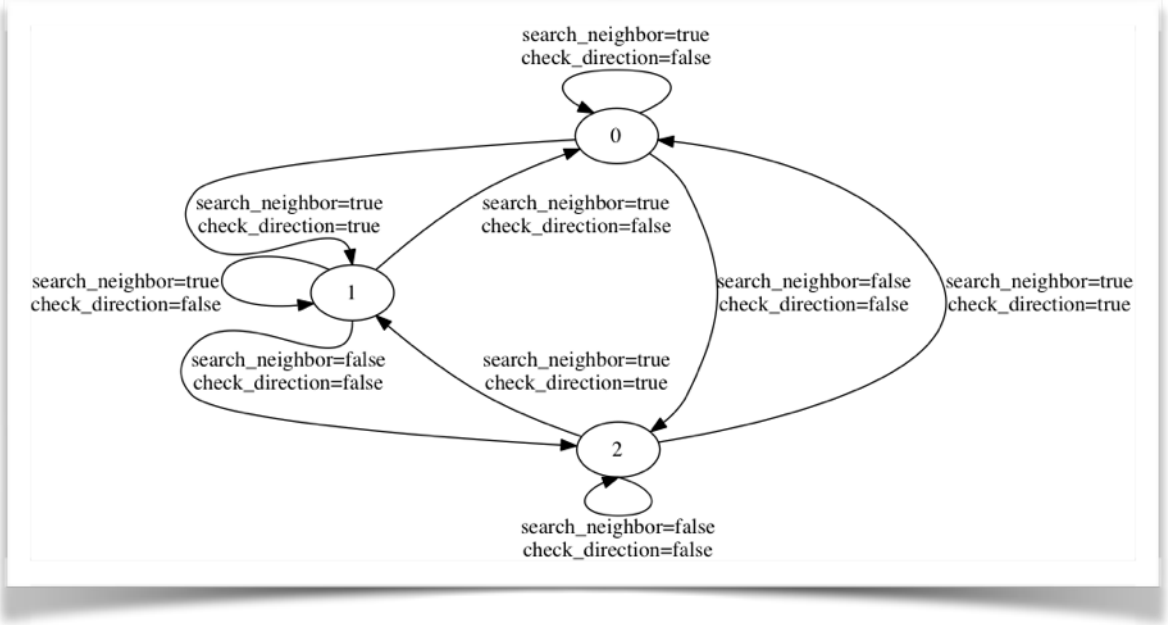

Figure 3: Finite state machine.
(Own illustration.)

The finite state machine is designed only for the defined environment and the described behavior. In theory the swarm can be extended to any number of copters, but so far only the behavior of four copters has been implemented and tested.

The variables search_neighbor and check_direction are both independent in the number of individuals, as well as the dimension of the search space. Since the Copters in our case have fixed positions, we prevent interference by selecting explicitly the lateral distance sensors in the implementation. In an extension or modification of the test environment, therefore the selection of the sensors must be adjusted.

6. Resulting behavior

The copters almost achieve the expected behavior which is

1) Oscillate from 40 cm to 90 cm vertically since the readings of IR sensor are currently accurate in this range.
2) A horizontal range of 30-60 cm for sonar sensor of under observation copter is defined to detect the neighbor copter.
3) The task to achieve the swarm behavior of four copters is sub-divided in a way that each copter has to find its two neighbors.
4) Different states are defined and reached depending upon the number of neighbors a copter has.

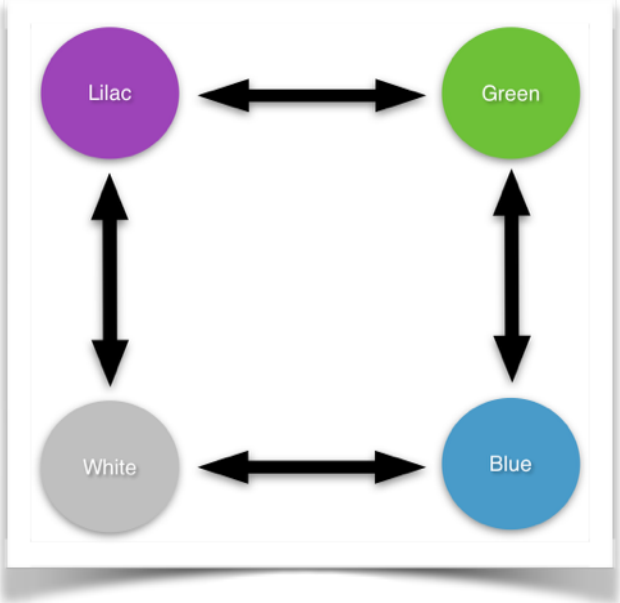

Figure 4: Test setting.
(Own illustration.)

6.1. Observation

Once state 2 is achieved, the height and swarm behavior should be maintained by the copters indefinitely. Actually, due to different battery levels and consumption of the copters, sometimes one copter loses the connection to the other ones and then it tries to find it again. This cycle goes on until the copters are out of battery.

7. Conclusion

The task of the team project was to program the collective behavior of decentralized and self-organized copters in achieving the cooperative height holding. There are four agents which are locally interacting by searching two neighbors. This leads to emergence of our desired swarm behavior. The agents, in our case copters, are performing two tasks:

1) Oscillating:
   Moving in a predefined vertical range (40 cm to 90 cm) in a step wise manner where each step is of 2.5 cm/sec.

2) Detecting Neighbors:
   Anything found in the set horizontal range of sonar sensors, mounted on the copters, is considered as a neighbor.

The algorithm is defined to reach the optimal solution, which is to achieve state 2 (the state where every individual copter has succeeded to find its two neighbors). This local behavior of each copter generates the wanted global swarm behavior.