EE3L11: Bachelor Thesis

Two dimensional swarming

Neighbour detection

by
Lars de Kroon, 4383095
Jeroen van Uffelen, 4232690

June 13, 2022
Version 1.0

Abstract

Robot swarming has been done for quite some years now. However, creating a very large swarm can become very difficult, since the networking becomes complex. With small swarms, a robot can broadcast information, or a central node can coordinate the robots. However, when a swarm becomes very large, these techniques fall short. Broadcasting all information is too much to be evaluated by every single robot, and a central node can't control it as well. In this research, we have looked at how nature works and tried to find a solution by mimicking its behaviour. In nature, animals stay in a swarm, based on their instinct combined with following their neighbours' behaviour. This document is part of a bigger research which implements a bird-flocking algorithm with cars. For bird-flocking, the speed and heading of neighbours need to be known. Detection of neighbours with robots is technically challenging. We have researched a way to detect neighbours based on a hardware filter based on RSSI, in combination with a software filter.

Preface

The thesis is written as a part of the graduation for the bachelor Electrical Engineering in TU Delft. The project was commissioned by the department "Embedded and Networked Systems". The original goal of this project was to create a platoon with cars getting into and staying in a formation. However, because of the short time and misjudges technical challenges, the goal was changed to implement a 2D bird-flocking system with cars.
We would like to express our gratitude to our daily supervisors dr.ir. Ashutosh Simha and ir. Suryansh Sharma as well as our head supervisor prof. Venkatesha Prasad. Furthermore we want to thank dr.ir. Chris Verhoeven for some extra focus and guidance in the end stage of the project. Finally we would like to thank our co-researchers: Melle Minten, Sven Dukker, Ivar Hendrikson and Thijmen Hoenderboom for an enjoyable and educational collaboration.

Contents

1 Introduction

1.1 Swarming

In robotics, swarms are collaboratively working robots. In this research, the behaviour of robots is inspired on how swarms with animals in nature work. In swarms, a lot of small entities work and move together without knowing the purpose of the whole swarm. Just like in nature, there are a lot of different swarm behaviours, for example, bird flocking, bee swarming, fish schooling, etc. However, the common denominator of all the behaviours is that the individuals only make decisions based on their neighbours and their instinct[1]. The instinct of all individuals is the same.

1.1.1 Nature swarms

Animals follow their neighbours based on a lot of different ways. This can be a chemical sensing process [2], an audio process [3], a visual process [4], a vibration process[5] etc. All these systems are very complicated and perfected through years of evolution.

1.1.2 Robot swarms

Robots try to emulate the behaviour of animal swarms, however, there are some technological difficulties in imitating the behaviour of the animals. The main difference is that animals have very sophisticated 'sensors', and very efficient processing of those. Nowadays it is still difficult and very expensive to build this advanced detection and processing in robots. Because swarms exist out of a lot of entities, the costs should be very low to make it feasible to build a lot of them.

1.2 Neighbour detection

To make decisions based on the behaviour of the neighbours, it's important to know who the neighbours are.

1.2.1 Indirect sensing

Animals process information from for example their eyes. Also for humans, it's easy to see where another human is headed and approximately guess their speed relative to you. This process works very efficiently. We have a visual way of determining what the front or back of a human is, and based on this, we can see the orientation of someone, by processing the proportion of the body that we see.
This process would probably work very good on robots as well, however, the eye in combination with the brain is a sensor network that we can't yet create affordably.

1.2.2 Broadcasting

If all robots are connected to a single network, it becomes easy to communicate with all of the robots. When the robot is equipped with a sensor that measures its absolute location, it can broadcast this information to all the robots. The robots can then determine the neighbours by processing the raw absolute locations to a relative location.
The downside is that this solution does not scale well. When there are hundreds of robots, it will become almost impossible for a single robot to calculate its relative location, because it needs to process so much data.

1.2.3 Centralized system

To fix the problem of the heavy load used for calculations on the robots, a central hub can be inserted into the network. This can be a powerful computer which is capable of offloading this

heavy computation from the individual robots.
This way, there is a dependency created on an expensive computer, which is probably less mobile. This dependency is a single point of failure. So there is a lot of overhead in maintaining a stable connection with the central hub. When the swarm scales more, like to a million robots, the central hub won't be able to facilitate all of the robots anymore. Thus this solution won't scale perfectly as well.

1.2.4 Network division

A robot only needs to talk to its neighbours. By turning every robot into an access point, which broadcasts a network with limited range, it's possible to limit the number of neighbours it sees. The robot can talk with neighbours within its range, however, it is unable to know their location. This is not a problem, since the robot averages the information of all the neighbours. This solution does scale theoretically since all the nodes will only be talking with other nodes within a manageable range. This solution imitates nature at the closest level.

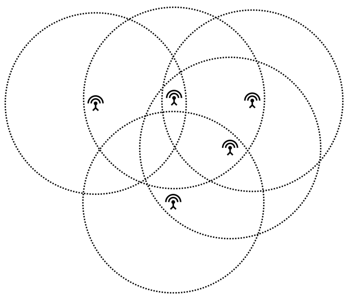
Figure 1: Five robot with their neighbours

1.2.5 Potential field

As a result of the small groups of networks. It's possible to label each node with a number, based on the number of their surroundings. As a result, a potential field will be created. This potential field is useful for the navigation of nodes in a swarm. As seen in figure 2 and figure 3, there are five nodes. When the center node will be marked as potential 0, the surrounding nodes will receive that. Once they receive a 0, they will broadcast potential 1. The last node is not in range of potential 0, but is in range of potential 1, and therefore will broadcast a potential 2.

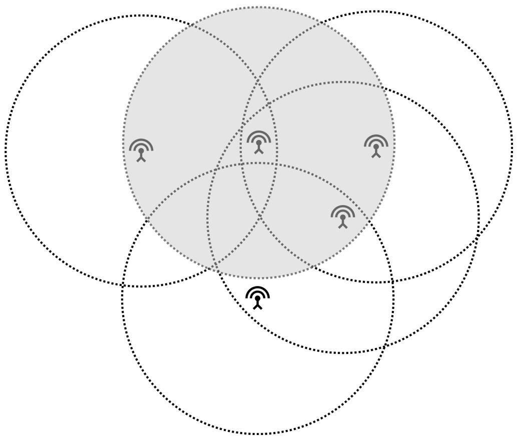
Figure 2: Potential 0 marked node

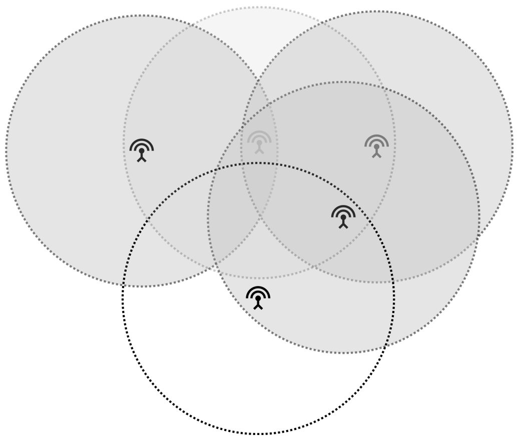
Figure 3: Potential 1 marked nodes

1.3 State of the art analyses

Swarm robotic research has been going on for quite some time now. A lot of research has been done in the communication field as well. Companies like Intel have given demos of drones flying in formations for more than five years now.[6] Very recently, a Chinese group managed to get an autonomous swarm flying in a bamboo forest. They used a mesh broadcast for communicating.[7] Protocols like BLE and ZigBee allow large mesh networks to be created.[8][9] Also, tests are performed with WiFi as communication.[10] Mesh networking is a very effective way to connect devices. These protocols have also been used in swarms already to create connectivity between the nodes.[11][12][13] The focus of most of the research is on creating effective paths and improving the QoS.[14][15] There is also a lot of research done on animal swarm behaviour.[16] There is however still an undiscovered field in translating this decentralized animal behaviour to a robot swarm.

1.4 Problem definition

There has been a lot of research done and hardware created with the purpose to create an as large as possible swarm. The problem with creating large networks is that there will also be a limit to the size of the swarm due to the processing capabilities of the robots. We try to find a solution inspired by how animal swarms in nature work. In nature, animals make decisions based on their neighbours and instinct. Detecting neighbours of a robot in the way that nature does this is complicated and very expensive because nature does a lot of processing based on vision or audio, which is too complex to implement in small robots. By using a fragmentation technique like discussed in 1.2.4 it is possible to create a forever scaling network. It is an approach that matches closer to how animals in real-life work. This design however does come with some technical challenges. There is a need for nodes that can function as access points as well as stations at the same time. Being an access point takes power, which can be a show stopper for some applications. This research has been done with WiFI, which is energy heavy. However, the same philosophy can be applied to other protocols as long as they meet the base requirements.

2 Program of Requirements

2.1 Program of Requirements for the whole project

This is the program of requirements that hold for the complete system. The goal is to present the boundaries is which the system must work.

2.2 Assumptions

There was already some hardware made available by our supervisor. From this our group could choose which micro-controller we were going to use. Also there were a few different robot chassis to choose from.

2.2.1 Mandatory requirements

Functional requirements

1. The robots must behave like a swarm.
2. The robots must be non-holonomic, meaning movement is restricted in some directions.
3. The robots must not use GPS to measure their location.
4. The swarm must work in 2D space.

Non-Functional requirements

1. The swarm must work with at least 3 robots.
2. There is a budget-cap of €150.
3. Swarm must be scalable.

2.2.2 Trade-off requirements

1. The swarm will be at least 6 robots.

2.3 Early design choices

From these requirements, and the hardware that was available, some early design choices were made. This section will discuss the hardware used in this project.

Vehicles

The "Autonomous robots" that are used, are non-holonomic, 3 wheeled vehicles. They are outfitted with an ESP-WROOM 32 development board, as well as a motor control chip. 2 of the wheels are driven by a PWM signal, generated by the ESP. Whilst the third wheel is an unpowered castor wheel.

ESP-32 WROOM

The ESP-32 WROOM development board functions as the main decision making unit of the car. It is a low-cost, low-power system on a chip micro-controller. The chip is capable of WiFi communications, which allows the car on which is outfitted to communicate with other nearby cars. Furthermore, the board is outfitted with an Inter-IC-bus (I^2C). This feature allows other integrated circuits to communicate with the micro-controller. This is necessary for the required magnetometer in the form of an IMU. The I^2C bus is also used by a distance sensor, but this is not directly of importance for this thesis.

ESP-IDF

Espressif provides a tool chain, called ESP-IDF. This tool chain is the main way to interact with and flash instructions on an ESP32 chip and can be installed with an installer provided by Espressif. ESP-IDF provides the API calls to use all the functions on the ESP32 chip. API reference can be found online, at [17].

2.4 Subgroup Program of Requirements

Our subgroup is tasked with the communication between robots within the swarm. These are the requirements.

2.4.1 Mandatory requirements

Functional requirements

1. A robot should be able to communicate with its surrounding robots.
2. A potential field should be created when the robots are on a line.
3. The system should be decentralized.

Non-Functional requirements

1. The system should be able to scale to millions of robots

2.4.2 Trade-off requirements

1. The system can create a potential field in 2 dimensions

3 Designing the neighbourhood detection

To make a swarm scale, and still have reliable communication, it is important for a node to talk with its neighbours. As it takes time to process a packet a robot receives, it is important to keep the processed packets to a minimum. But how do you determine if a packet has relevant information? In a centralized system, the hub can determine if a packet is important for a robot and forward it. But as our system needs to be decentralized, this will not work. So you want to filter the packets to the minimum needed to make the correct decisions. We have found a few ways to do this.

3.0.1 Preventing

If you can prevent a packet from arriving, it is also not possible to process it. This way there is no time spent on an irrelevant packet. This can be achieved by changing the transmission power on the ESP32. By changing the transmission power, the received signal strength index (RSSI) value will be changed. If the RSSI value of an access point is low enough, the surrounding nodes won't be able to receive packets from this access point. It is possible to change this threshold within the ESP32.

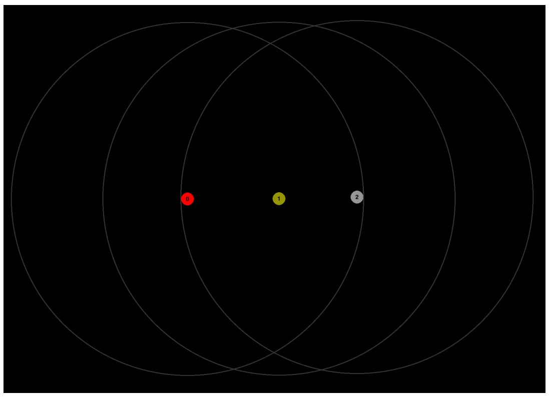

Figure 4: Linear neighbours. From left to right, node 0, node 1 and node 2 (full tx power)

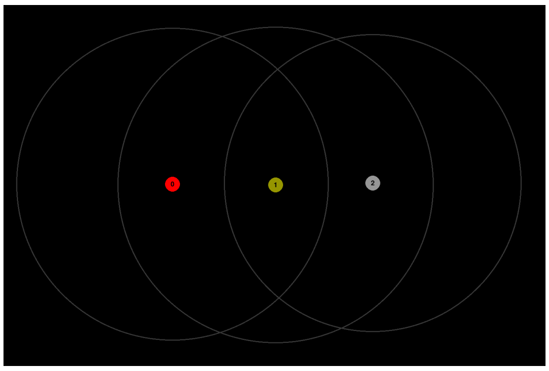

Figure 5: Linear neighbours. From left to right, node 0, node 1 and node 2 (reduced tx power)

In figure 4 there is an example. The small coloured circles are nodes, and the large circles represent their transmission power. All nodes send with the same transmission power and all have the same RSSI threshold. In this situation, node 0 does not need the information from node 2, as it is too far away to have relevant information. To prevent node 0 from receiving the information from node 2, it could up its RSSI threshold. But this would not prevent node 2 from receiving the packages sent by node 0. So both devices should lower their transmission power to keep out the noise, as seen in figure 5.

3.0.2 Filtering

Filtering is another method to prevent a packet from taking processing time. Filtering can be done on different layers. On the network layer, this is solved by the ESP-NOW protocol. This is a connectionless protocol developed by Espressif[18]. ESP-NOW can send messages to specific mac-addresses, which is basically a filter on network layer. Another possible layer is the message layer. For instance, if a timestamp is added to the messages, one could look at the timestamp to see if the message is still relevant or if it is too old. If it is the case that the message is too old, the message can be filtered. A message can also be filtered if it does not contain any new information, this prevents that the message needs to be unnecessary processed by the motor controller.

3.0.3 Skipping

Besides filtering, skipping can also be implemented. Filtering is executed at the receiving side of the channel while skipping can be done at the transmitting side. However, since the receiving node wants to know how much neighbours are transmitting to it, the transmitting node should never skip messages, because then preventing won't work anymore.

3.1 Simulation

In research done by Melle Minten en Sven Dukker into the RSSI values you get from the ESP32, they found that the RSSI values are not representative of the distance between two ESP32s. But what they also found, which is of more interest to us, is that when to ESP32's get closer than 50 centimetres to each other, the RSSI values will drop, as can be seen in figure 6. This means that if you filter only on RSSI values for your neighbours, it is possible that you filter out your closest neighbours, and get information from robots that are further away. A simulation was created to test if this is a problem.

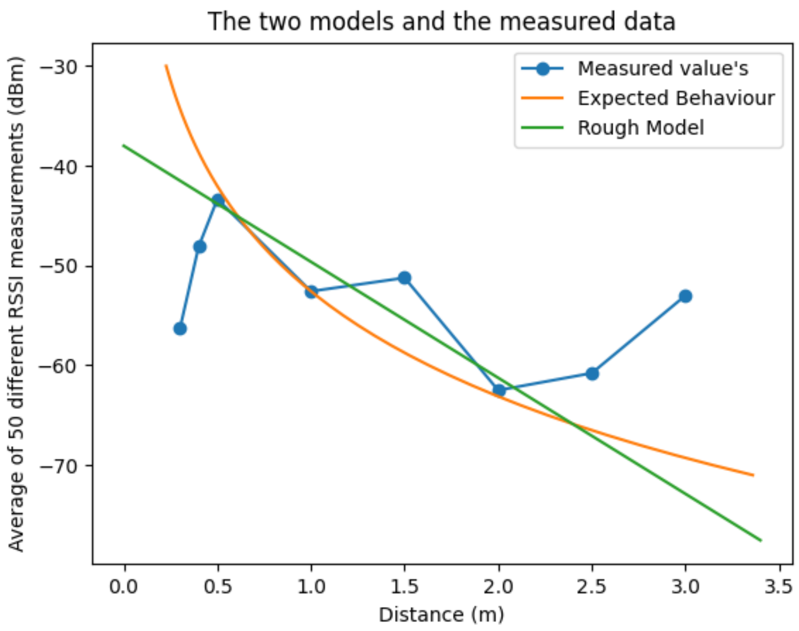

Figure 6: RSSI different models in comparison with the measured values

3.1.1 PyNBoids

To start with simulating, we found a boids simulation written in python by Nikorasu, called PyNBoids[19]. This is our starting point. Boids[20] are a way to simulate behaviour in swarms with only 3 simple rules. These rules are:

1. Avoid collision with other agents in the swarm.
2. Align headings to move in the same direction.
3. Steer towards the middle of the group

From this research, it is also known that only information of the 7 closest neighbours is needed to get bird flocking behaviour in the boids.

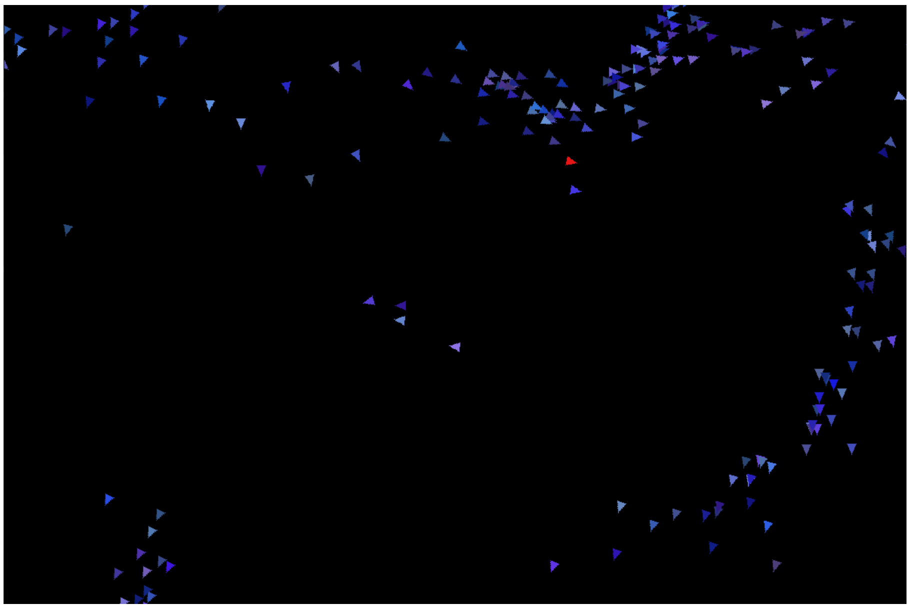

Figure 7: Screenshot of PyNBoids

3.1.2 Test cases

To test our problem, only rule 2 was implemented. The simulation was run for 10 seconds. Every second the headings from all boids are saved. With this, you can calculate an average heading and standard deviation. The standard deviation is the metric used to see if a test case is better or worse than the base test case. For all the tests, everything is the same at the start. During the tests, only the neighbour detection is changed. These are the cases that were tested:

1. This is the basis. During this test, the boids get information from the 7 nearest neighbours that are in range.
2. During this test, the 3 closest neighbours are ignored. If there are 10 boids in range, the information from those 7 boids is used. Otherwise, the boid knows less information from its neighbours.
3. In this test, in each simulation frame a random number between 0 and 7 of the nearest neighbours are ignored. But the boid still gets information from 7 boids, if they are in range.
4. In the last test, we check what happens if you have information from less than 7 boids. In this case 4 of the nearest neighbours.

3.1.3 Results

In figures 8, 9 and 10 the results is shown from the base test. The first two figures are the histograms from 0 seconds and after 10 seconds. You can clearly see that after 10 seconds the boids aligned their headings within around 80°of each other. In figure 10 you can see the standard deviation at every second a measurement is done. The lower graph is the number of groups identified during the test. For each of these groups, the standard deviation is calculated. There is a weighted sum to get the standard deviation from all groups. As you can see, the standard deviation is about 30 at the 10-second mark. In appendix A the test results from all the tests at every second can be seen.

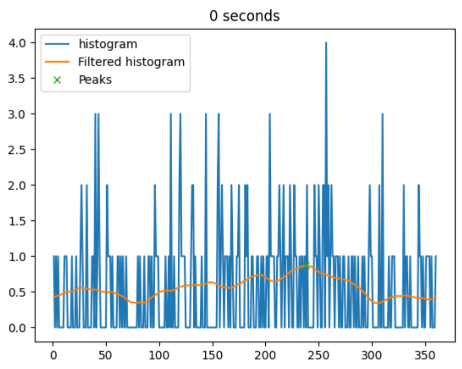

Figure 8: Histogram of the headings at 0 seconds

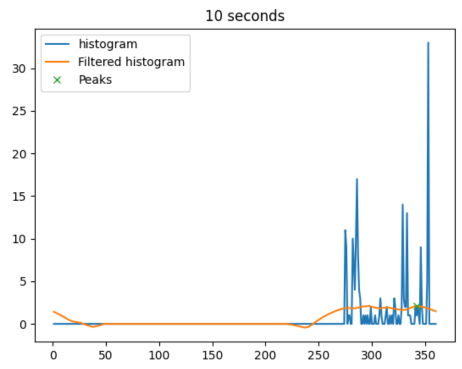

Figure 9: Histogram of the headings at 10 seconds

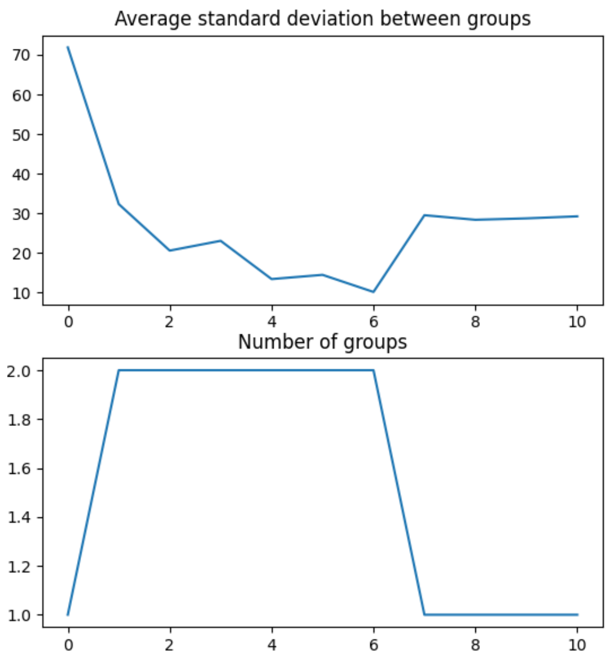

Figure 10: Standard deviation of the base test versus simulation time

In figures 11, 12 and 13 the standard deviation is seen for tests 2, 3 and 4. You can see that test 2 and 4 both end at about 30. Figure 12 shows that test 3 has a different result. Our conclusion about this test is that instead of the boids having information about 7 neighbours, they essentially have the headings of 14 boids around them. This could happen when at frame x, it gets the first 7 nearest neighbours, and at frame x + 1 it could get the information from neighbours 8 through to 14. From these results, we concluded that it does not matter if not the nearest neighbours are detected. It does matter how many neighbours are detected.

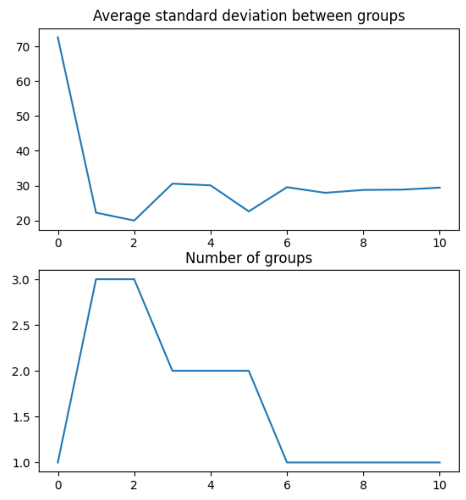

Figure 11: Standard deviation of the test case 2 versus simulation time

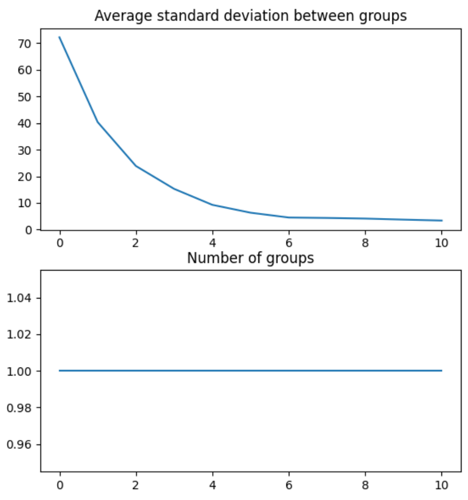

Figure 12: Standard deviation of the test case 3 versus simulation time

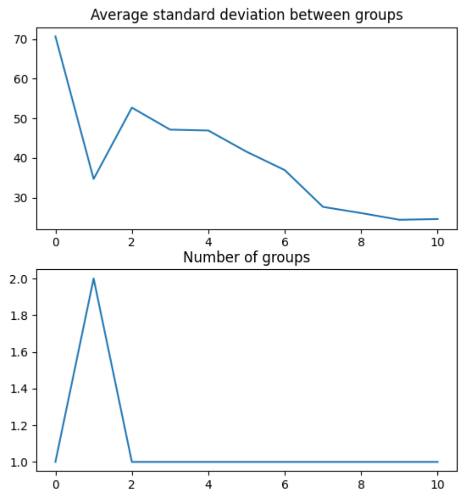

Figure 13: Standard deviation of the test case 4 versus simulation time

3.2 Implementation

3.2.1 Linear neighbours

On the ESP32, it is possible to change the transmit power. The first thing that was tried, was to lower the transmission power of the ESP32 and send a broadcast message using ESP-NOW. From the documentation of the ESP32, we learned that the transmit power can be varied between 2 dBm and 20 dBm. We found out rather quickly, that setting the transmit power to its lowest setting, the range of ESP-NOW is still more then 10 meters. This was not usable for a our purpose. So we changed tactics. Figure 14 shows how it works. Node 1 and node 2 are both ESP32s. Node 1

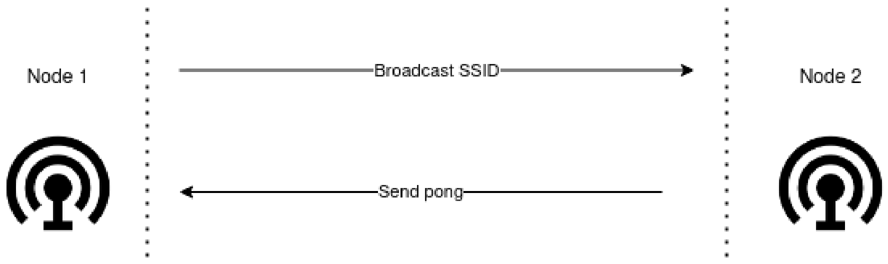

Figure 14: Neighbour pinging based on RSSI

is a wifi access point. It will send out SSID advertisement data. This can be received by node 2, and it can calculate its RSSI value of node 1. Node 2 will send back this value to node 1, using ESP-NOW. If there are more than 2 nodes in the network, node 1 can then scale its transmit power so enough neighbours can detect it. By using a software threshold for the RSSI value on the receiving end, we have verified that this method works in a straight line with three nodes and a separation of 5 meters between the outermost nodes. After we had verified this, we have done some research on the received signal strength when changing transmission power. This resulted in the graph in figure 15. In this graph, we can see that the difference between the maximum and minimum power is about 20. Also, the RSSI values are quite linear with the transmit power. With this knowledge, it is easy to come up with a simple automated system that scales its transmission power according to the number of neighbours it receives. And by varying the software threshold we set for the RSSI value on which a node will send information back, we can get to a quite small area in which nodes react to each other.

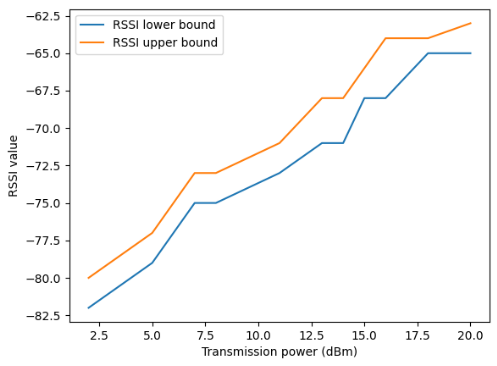

Figure 15: RSSI value versus the transmission power

3.2.2 Two dimensional neighbours

When implementing the same strategy on a two dimensional field, a new problem arises. We can't filter anymore based on the count of the ESP-NOW messages that arrive, because node can have a lot more neighbours then two as can be seen in figure 16.

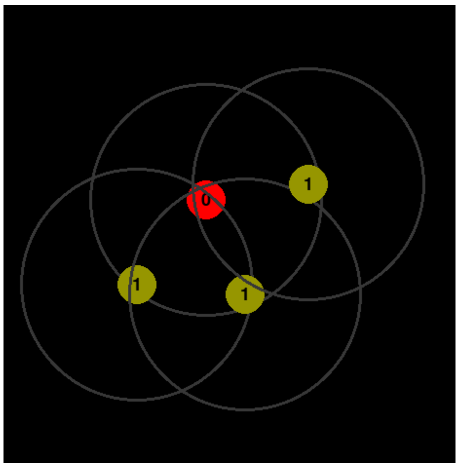

Figure 16: Node with more than two neighbours

To fix this problem, a change in strategy is implemented. The 'pong' message from the neighbours nodes will include the RSSI value the access point which they are messaging. The access point can now determine whether neighbours are in the same area of if they are further away. It the access point collects data from neighbours which are further away than other neighbours, it will lower it's transmission power to prevent receiving any data from the furthest neighbours. The new protocol is written down in figure 17

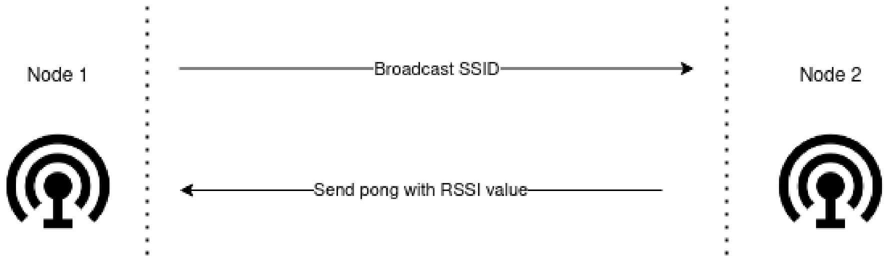

Figure 17: Neighbour pinging based on RSSI, RSSI in pong

4 Designing the potential field

4.1 Algorithm

The goal of the potential field is to create a navigational flow through a swarm. By labelling one node as 0, the adjacent nodes should be able to get a number based on their neighbours. To assign the right potential the following algorithm is used.

Algorithm 1 Potential marking algorithm

```
potentials ← empty list
for neighbour in neighbours do
    if neighbour.potential is defined then
        potentials.append(neighbour.potential)
    end if
end for
nodePotential ← $\min(potentials) + 1$
```

By using algorithm 1, a node will always be updated based on the lowest member it can scan. So if a node can scan potential 0, it will always become 1. If a node scans a 1 but does not see a 0, it will always become a 2, etc.

4.2 One dimensional field

To get a better insight into how the potential field would evolve from the neighbour's networks, a program is created. The goal of the program is to draw a node as potential 0 and then create the potential field based on the hops it will take to get to potential 0. The different potentials are represented as colors.

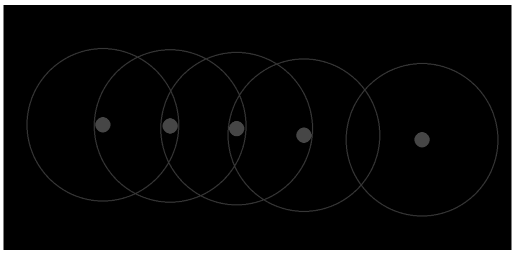
Figure 18: Linear unmarked nodes

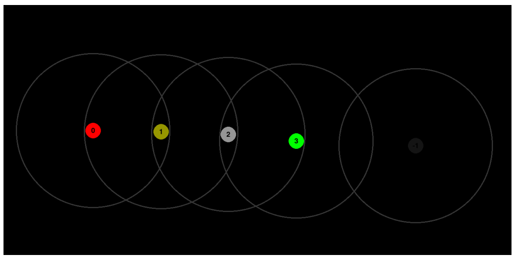

Figure 19: Linear marked nodes

As seen in figure 19, the potentials are marked based on the neighbour. Since the most outer node is out of range, it is marked as -1. When the base node is changed, the field will change accordingly. The result will be a vector field which navigates the nodes to the root node.

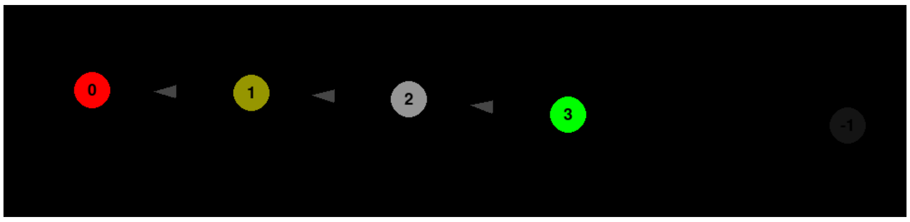

Figure 20: Linear potential field

4.3 Two dimensional field

When a swarm starts to move in two dimensions the field gets more complicated, but the theory of assigning the nodes will still work.

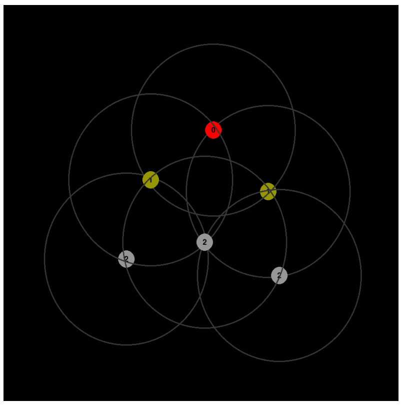

Figure 21: 2d potential nodes

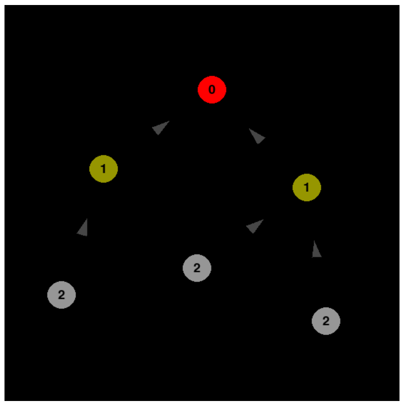

Figure 22: 2d potential field

4.3.1 Inefficient field

The goal of the potential field is to find a way to the root node. However, by using this way to mark the nodes based on the neighbours, we can't be certain that we actually are using the shortest possible path to the 0^{th} node. As can be seen in figure 24.

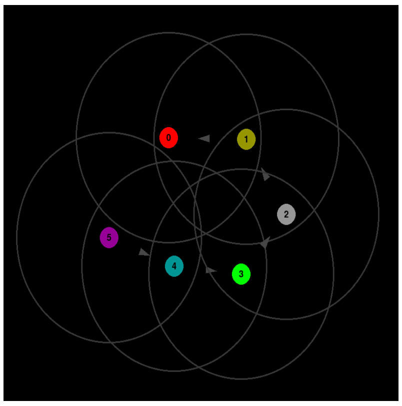

Figure 23: 2d potential nodes, inefficient

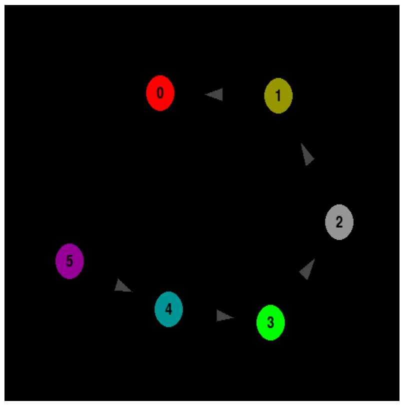

Figure 24: 2d potential field, inefficient

When a node is placed between 0 and 5, the path between 0 and 5 will be much shorter and therefore much more efficient.

4.3.2 Efficient field

It is obvious that when the swarm gets larger like in figure 25, and a lot of robots are flying around, the field will stabilize. The different potential layers are also clear to distinguish when the field gets larger.

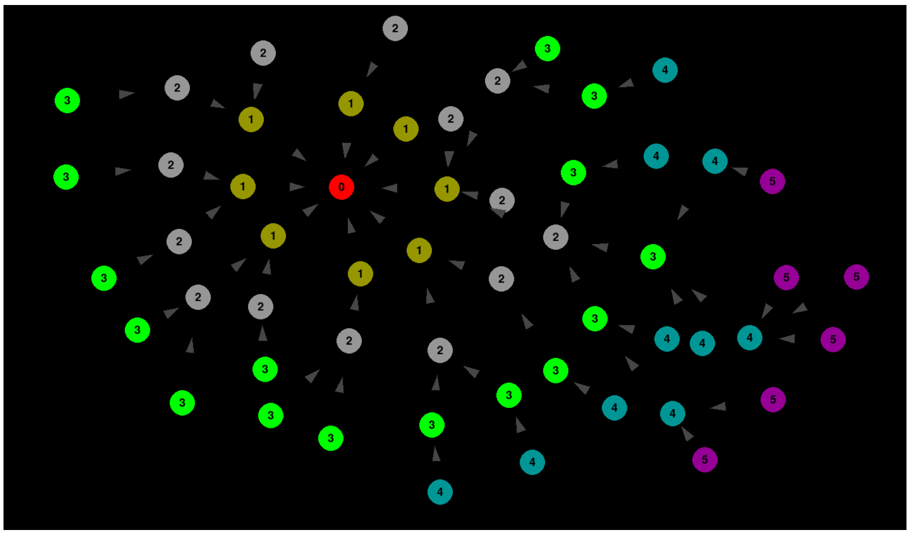

Figure 25: Large potential field

Nagekeken The field is always at optimal stability when the robots are always at the max distance of another robot. The field that will emerge is a hexagon shape in this case. This is very logical since a hexagon is the most efficient figure with respect to the ratio between the covered area and the length of the vertices.

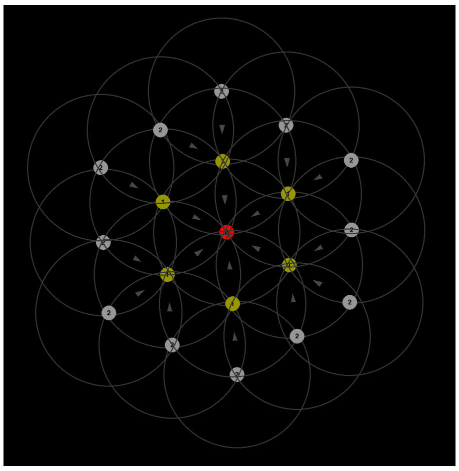

Figure 26: Hexagon potential field

4.4 Implementation

As discussed in the previous part 3 we use the ESP-NOW protocol to broadcast information to the neighbours. The purpose of this ESP-NOW message is to select the correct neighbours and simultaneously align the headings and speed across the swarm.
To create a potential field, it is necessary to know the potential of the neighbours as well. The way that we implemented it is by sending the current potential back to the node that initiated the ESP-NOW message. We only send the potential back if the node lies in the neighbour zone because it can be the case that a node is filtered by the software of another node.
For testing purposes, the zero potential will be marked by a python server which is connected to all nodes. This server can keep track of the nodes to validate whether the correct potential is assigned. The surrounding nodes will update accordingly.

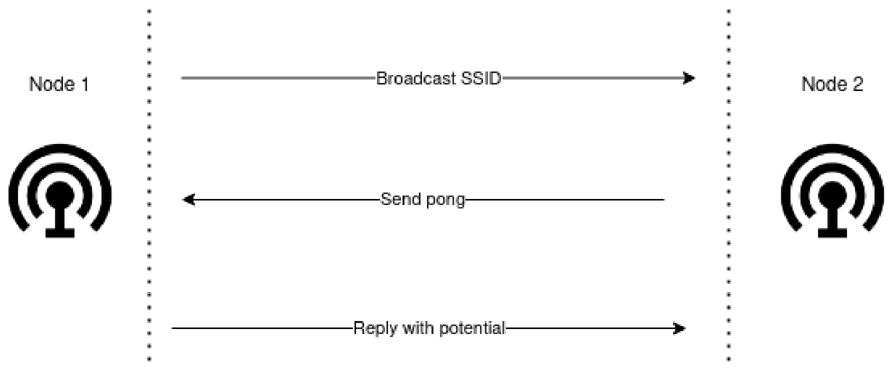

Figure 27: Neighbour pinging based on RSSI, with RSSI in pong and reply with potential

4.5 Hardware validation

To validate the working potential field, an LED light on the ESP chips will be used. The higher the potential is on a given node, the lower the blinking frequency will be. In this way, replacing two nodes, should result in an automatic interchanging of the corresponding blinking pattern. This hardware verification is limited to a distinct amount of potentials, but can be very well used to demonstrate the proof of concept.

5 Conclusion

This research has resulted in more knowledge about neighbourhood detection and how to create a potential field using real hardware. We were able to find and connect to your neighbours using the method shown in chapter 3. Using this method robots are able to relay their heading and speed to their neighbours, which is beneficial for the group project of which this thesis is a part of. From this neighbourhood detection, we were also able to show that it is possible to create a potential field. Because we create a small network per robot, this method should be able to scale to millions of robots, without the need for a central hub. So all targets from the program of requirements are met.

6 Discussion

The result of this research matches the expectation very well. The initial design philosophy is working on the hardware. The simulations gave a good insight into the possible outcome, and the hardware reflected these computations. This research can be used to further advance the natural behaviour in robot swarms. However, to make it more feasible, a lot of follow-up studies must be performed.
The simulations of the potential field were done with the assumption that every node has the same range in which it detects neighbours, however, the real-life scenario is that those will be different throughout the field. This will result in a change in the potential field. This change is little reflected when the number of robots is small, but when creating a large swarm, this change will affect a lot and needs to be further researched.
The potential field is also created on slowly moving or stationary nodes. When the nodes are moving faster (like in real swarms) the refresh rate of the neighbourhood detection and the potentials can be more pressing. There is no research done on the minimal requirements of a swarm regarding those fields. This study has also used WiFi, which is an energy-heavy communication protocol. This research should also be compatible with protocols like BLE, as long as there is a way to measure the broadcast intensity of a node as well as a way to send a connectionless message.
The potential marking works very well, however, to create a correct navigation path, it is necessary for a node to know what the direction is towards a node in a layer potential. More research should be done on the ways that this potential field can be used in an efficient way.
The potential field is lacking in providing the shortest path. By generating a secondary potential field, for example by sending an ESP-now broadcast to a wider range of their neighbours, it is very well possible to create a preferred moving direction for robots to stabilize the field. It should be researched how to create a stable field out of decentralized robots.
Nevertheless, we expect the result to have a contribution to how communication between neighbours is performed in swarming.

Bibliography

[1] Y. Liu and K. M. Passino, “Swarm intelligence: Literature overview,” *Department of electrical engineering, the Ohio State University*, 2000.

[2] G. Arnold, B. Quenet, J.-M. Cornuet, C. Masson, B. De Schepper, A. Estoup, and P. Gasqui, “Kin recognition in honeybees,” *Nature*, vol. 379, no. 6565, pp. 498–498, 1996.

[3] C. Chapman, “Field studies of hearing in teleost fish,” *Helgoländer wissenschaftliche Meeresuntersuchungen*, vol. 24, no. 1, pp. 371–390, 1973.

[4] A. Litke, N. Bezayiff, E. Chichilnisky, W. Cunningham, W. Dabrowski, A. Grillo, M. Grivich, P. Grybos, P. Hottowy, S. Kachiguine, R. Kalmar, K. Mathieson, D. Petrusca, M. Rahman, and A. Sher, “What does the eye tell the brain?: Development of a system for the large-scale recording of retinal output activity,” *IEEE Transactions on Nuclear Science*, vol. 51, no. 4, pp. 1434–1440, 2004.

[5] R. B. Cocroft and R. L. Rodríguez, “The behavioral ecology of insect vibrational communication,” *Bioscience*, vol. 55, no. 4, pp. 323–334, 2005.

[6] A. Pospischil, “Multihop routing of telemetry data in drone swarms.” International Foundation for Telemetering, 2017.

[7] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, and F. Gao, “Swarm of micro flying robots in the wild,” *Sci Robot*, vol. 7, no. 66, p. eabm5954, May 2022.

[8] R. Li and X. Li, “Directional multi-path routing algorithm based on ble mesh,” in *2019 Cross Strait Quad-Regional Radio Science and Wireless Technology Conference (CSQRWC)*, 2019, pp. 1–3.

[9] Y. Qiang and Z. Fan, “Application of wireless mesh network based on zigbee in mine safety monitoring system,” in *2021 International Conference on Information Technology and Biomedical Engineering (ICITBE)*, 2021, pp. 48–52.

[10] A. Bhuiya, A. Mukherjee, and R. K. Barai, “Development of wi-fi communication module for atmega microcontroller based mobile robot for cooperative autonomous navigation,” in *2017 IEEE Calcutta Conference (CALCON)*, 2017, pp. 168–172.

[11] H. Sharma and M. Rajesh, “Design and simulation of wsn for zigbee based communication in multi-robot system,” in *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, 2017, pp. 1353–1355.

[12] A. Aijaz, “Infrastructure-less wireless connectivity for mobile robotic systems in logistics: Why bluetooth mesh networking is important?” in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2021, pp. 1–8.

[13] ——, “Infrastructure-less wireless connectivity for mobile robotic systems in logistics: Why bluetooth mesh networking is important?” in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2021, pp. 1–8.

[14] L. Liu and J. Zhou, “Ad hoc on-demand qos routing based on bandwidth prediction(aqbp),” in *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing*, 2012, pp. 1–4.

[15] M. Li, K. Lu, H. Zhu, M. Chen, S. Mao, and B. Prabhakaran, “Robot swarm communication networks: Architectures, protocols, and applications,” in *2008 Third International Conference on Communications and Networking in China*, 2008, pp. 162–166.

[16] S. Khan, S. Momen, N. Mohammed, and N. Mansoor, “Patterns of flocking in autonomous agents,” in *2018 International Conference on Intelligent Autonomous Systems (ICoIAS)*, 2018, pp. 92–96.

[17] Api reference - esp32 - — esp-idf programming guide latest documentation. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/index.html

[18] *ESP-NOW User Guide*, ESPRESSIF.

[19] Nikorasu. Github - nikorasu/pynboids: This is a boids simulation, written in python with pygame. [Online]. Available: https://github.com/Nikorasu/PyNBoids

[20] C. W. Reynolds, “Flocks, herds and schools: A distributed behavioral model,” in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, pp. 25–34.

A Appendix A

A.1 Base test

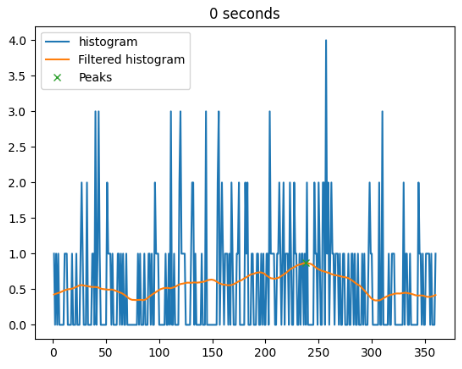

Figure 28: Histogram base test at 0 seconds

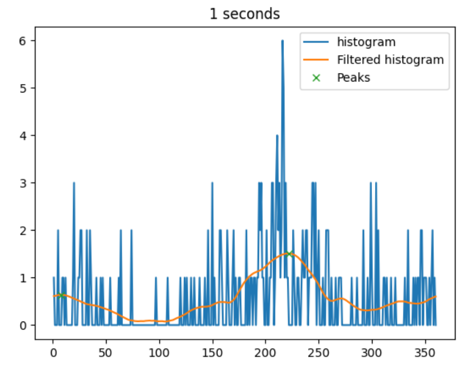

Figure 29: Histogram base test at 1 seconds

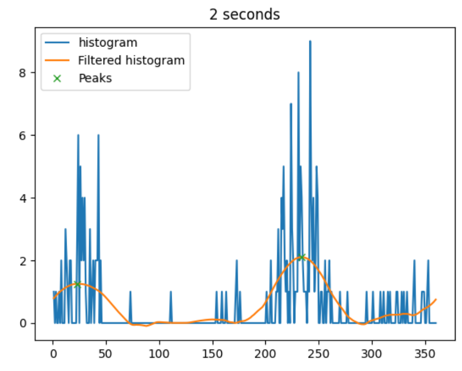

Figure 30: Histogram base test at 2 seconds

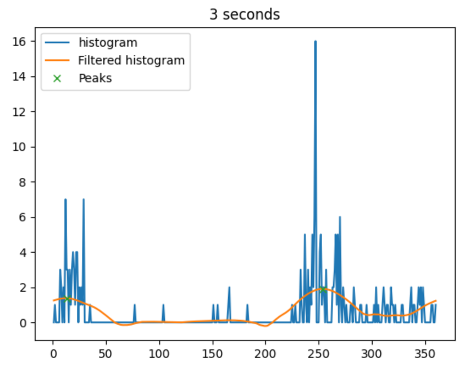

Figure 31: Histogram base test at 3 seconds

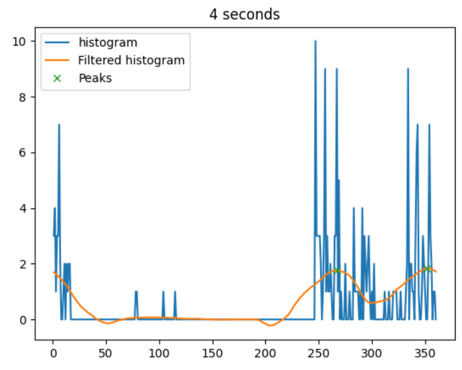

Figure 32: Histogram base test at 4 seconds

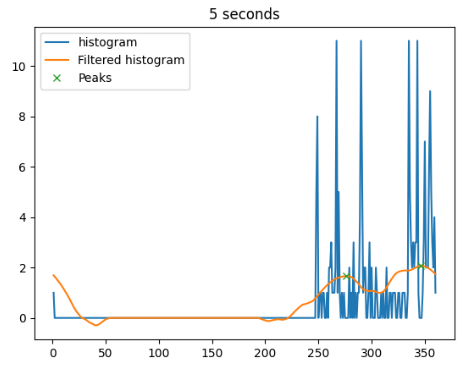

Figure 33: Histogram base test at 5 seconds

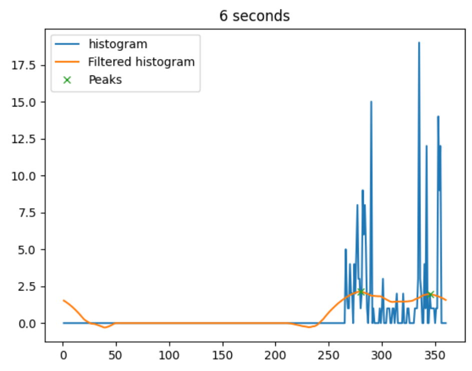

Figure 34: Histogram base test at 6 seconds

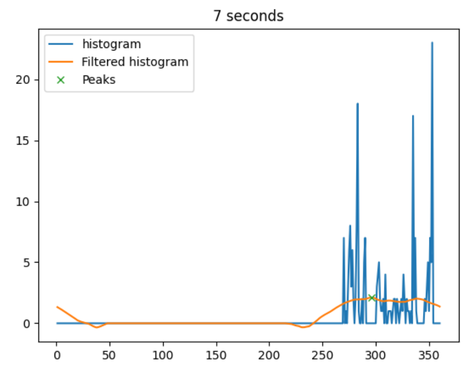

Figure 35: Histogram base test at 7 seconds

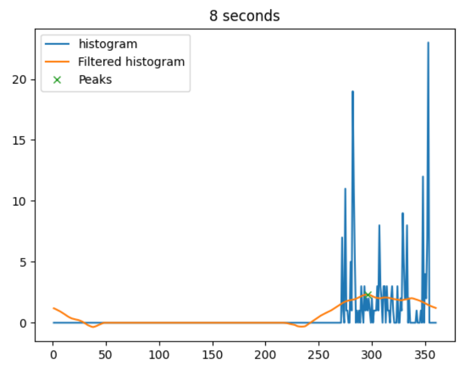

Figure 36: Histogram base test at 8 seconds

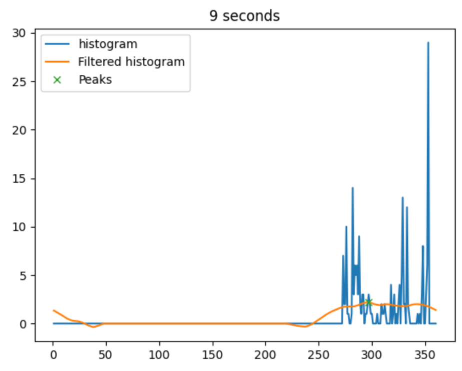

Figure 37: Histogram base test at 9 seconds

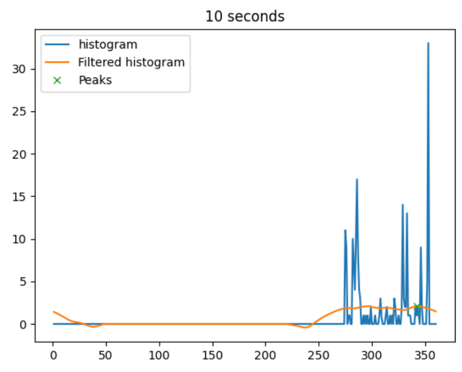

Figure 38: Histogram base test at 10 seconds

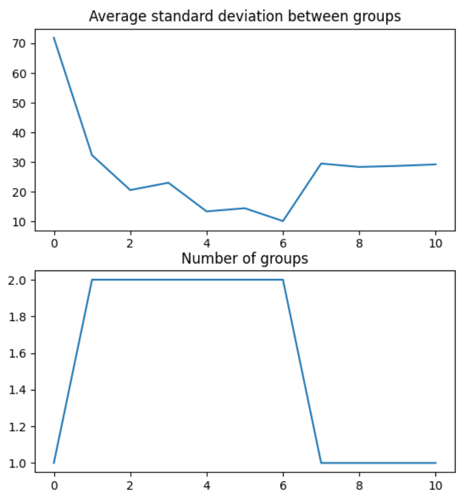

Figure 39: Standard deviation of base test

A.2 Test 2

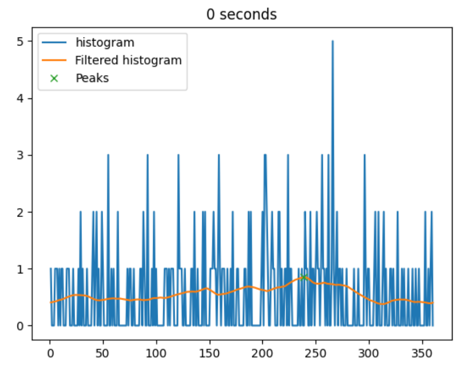

Figure 40: Histogram test 2 at 0 seconds

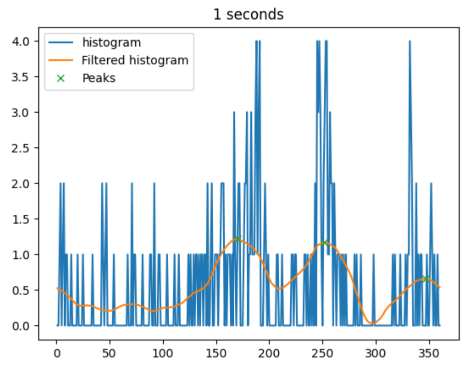

Figure 41: Histogram test 2 at 1 seconds

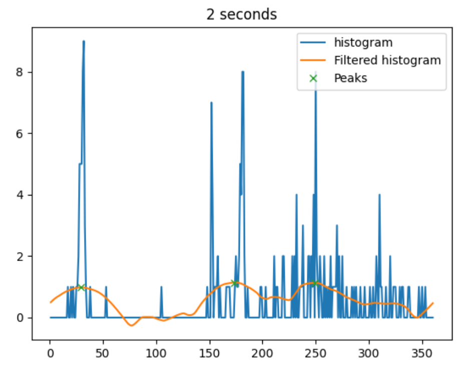

Figure 42: Histogram test 2 at 2 seconds

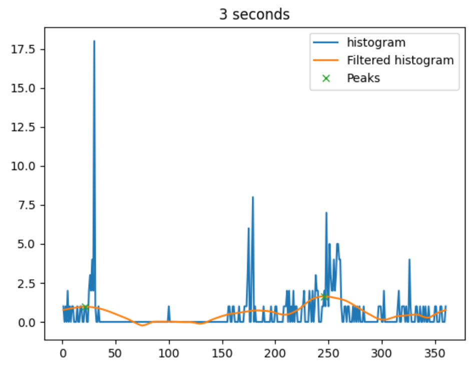

Figure 43: Histogram test 2 at 3 seconds

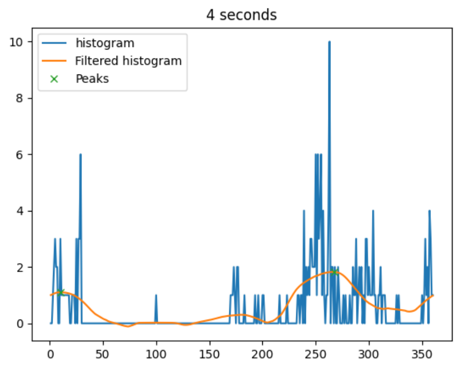

Figure 44: Histogram test 2 at 4 seconds

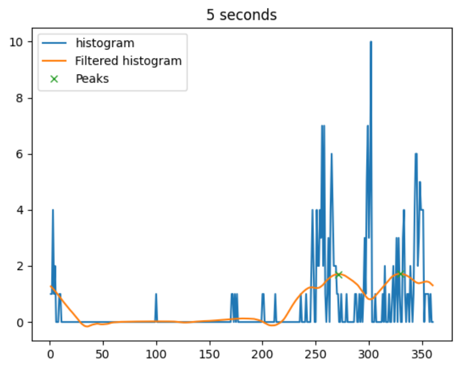

Figure 45: Histogram test 2 at 5 seconds

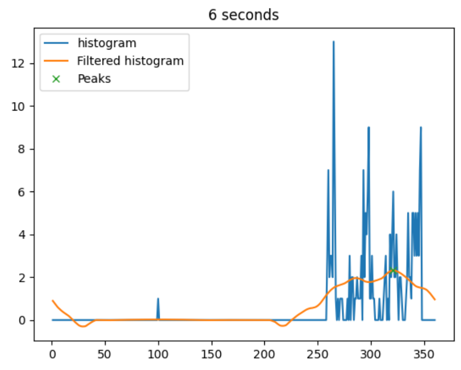

Figure 46: Histogram test 2 at 6 seconds

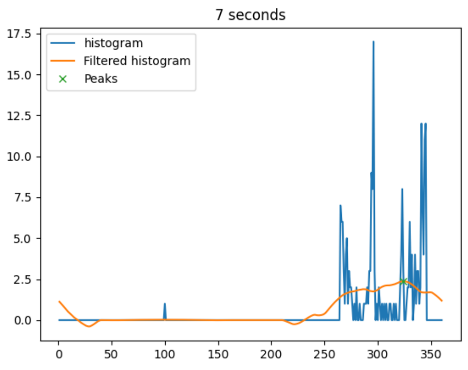

Figure 47: Histogram test 2 at 7 seconds

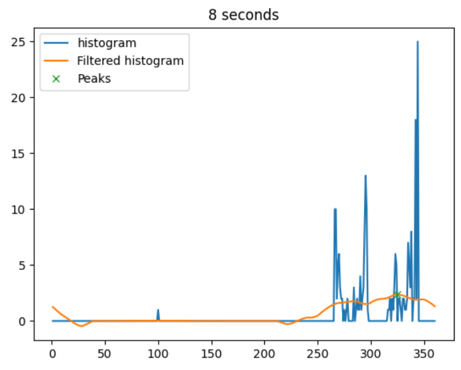

Figure 48: Histogram test 2 at 8 seconds

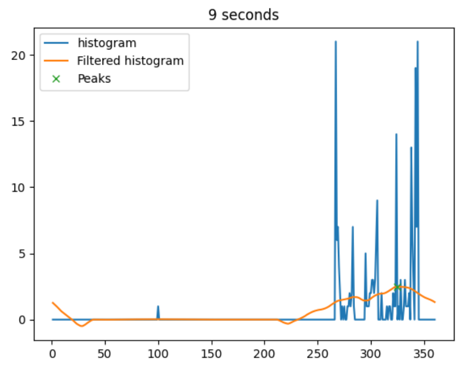

Figure 49: Histogram test 2 at 9 seconds

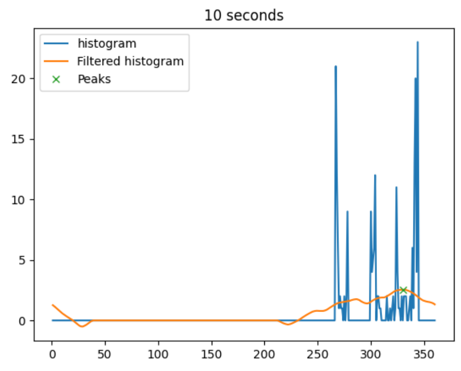

Figure 50: Histogram test 2 at 10 seconds

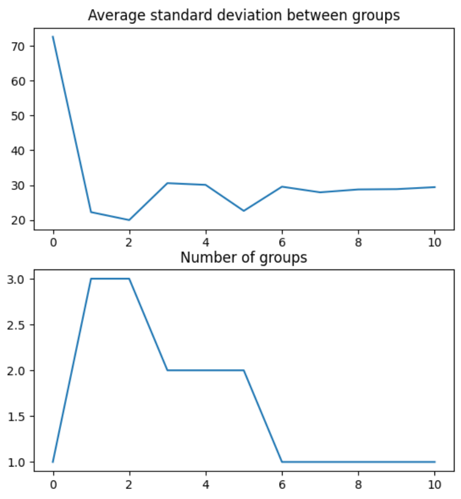

Figure 51: Standard deviation of test 2

A.3 Test 3

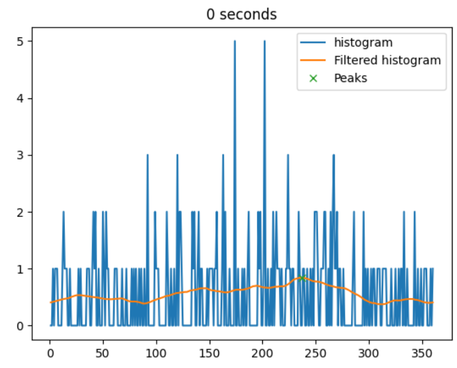

Figure 52: Histogram test 3 at 0 seconds

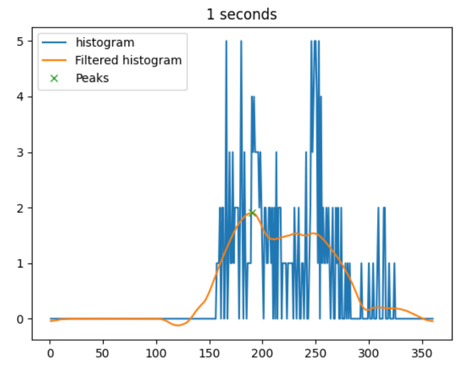

Figure 53: Histogram test 3 at 1 seconds

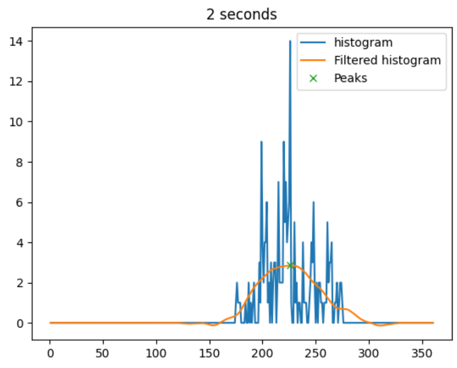

Figure 54: Histogram test 3 at 2 seconds

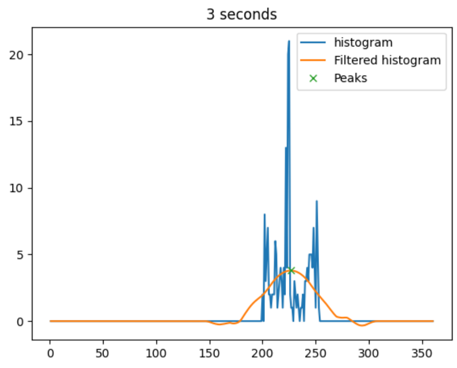

Figure 55: Histogram test 3 at 3 seconds

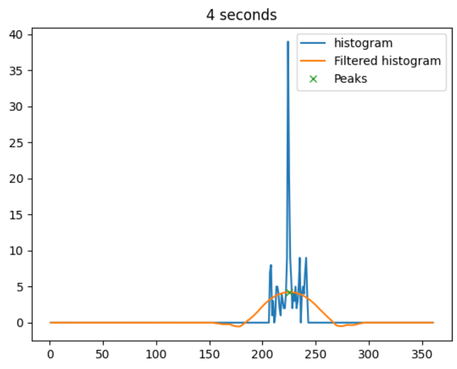

Figure 56: Histogram test 3 at 4 seconds

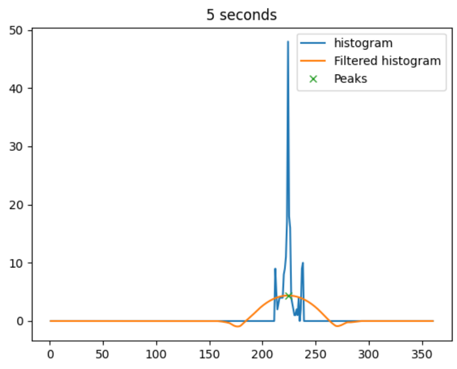

Figure 57: Histogram test 3 at 5 seconds

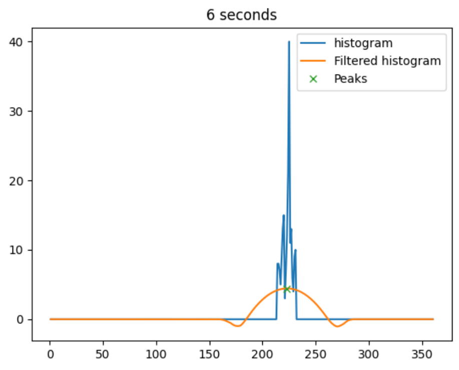

Figure 58: Histogram test 3 at 6 seconds

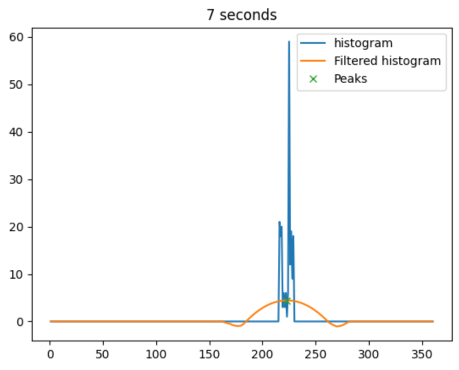

Figure 59: Histogram test 3 at 7 seconds

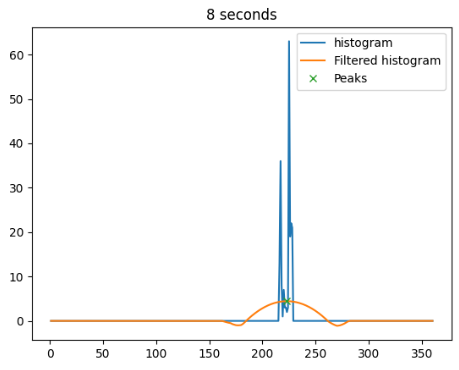

Figure 60: Histogram test 3 at 8 seconds

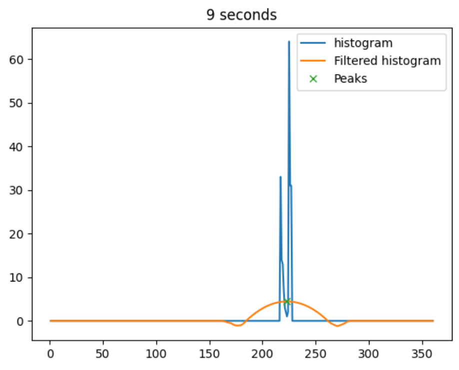

Figure 61: Histogram test 3 at 9 seconds

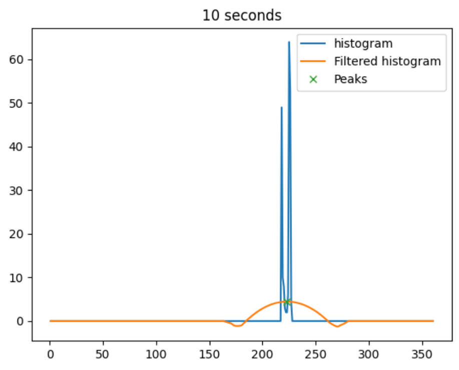

Figure 62: Histogram test 3 at 10 seconds

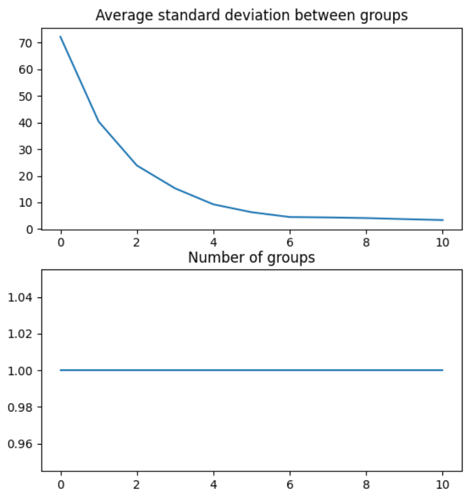

Figure 63: Standard deviation of test 3

A.4 Test 4

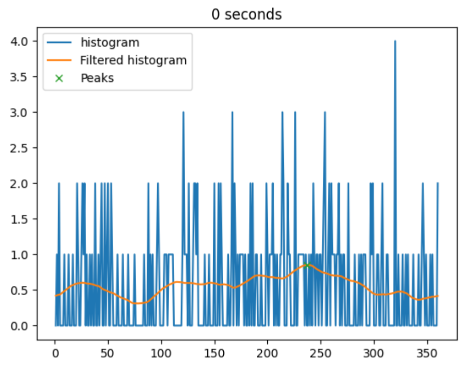

Figure 64: Histogram test 4 at 0 seconds

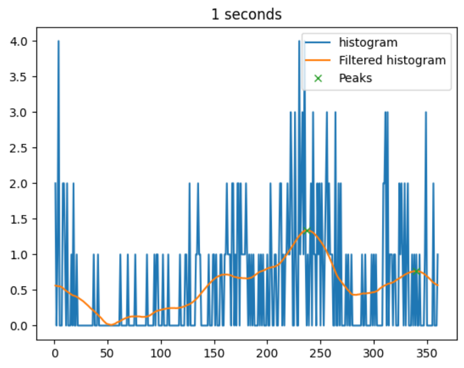

Figure 65: Histogram test 4 at 1 seconds

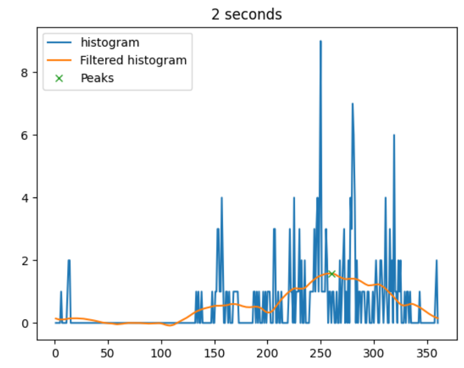

Figure 66: Histogram test 4 at 2 seconds

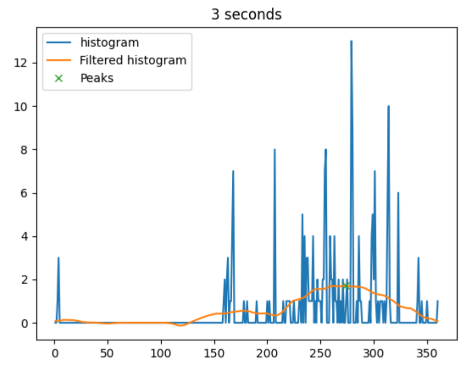

Figure 67: Histogram test 4 at 3 seconds

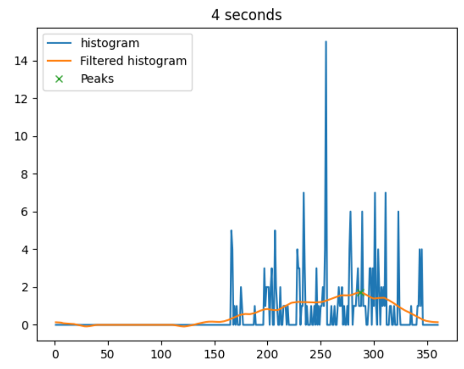

Figure 68: Histogram test 4 at 4 seconds

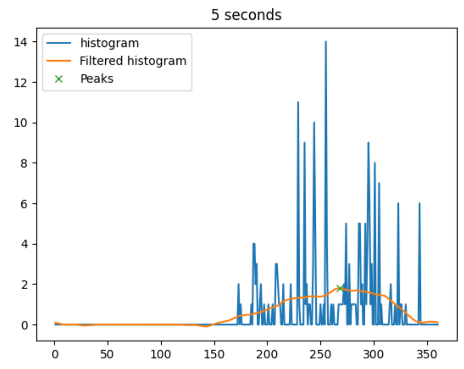

Figure 69: Histogram test 4 at 5 seconds

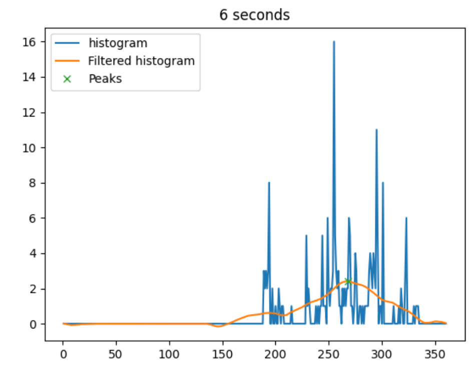

Figure 70: Histogram test 4 at 6 seconds

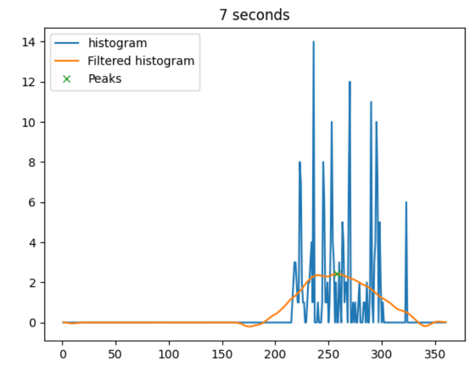

Figure 71: Histogram test 4 at 7 seconds

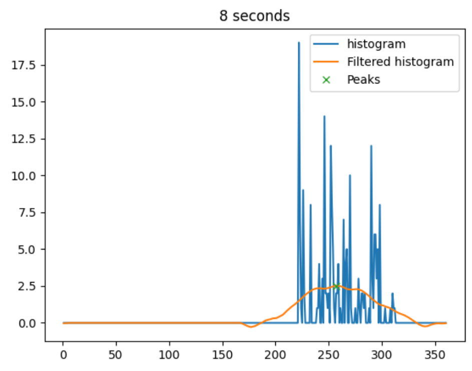

Figure 72: Histogram test 4 at 8 seconds

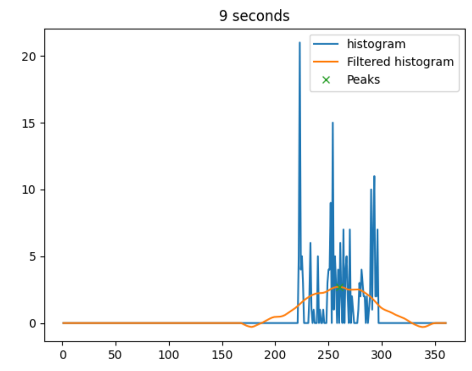

Figure 73: Histogram test 4 at 9 seconds

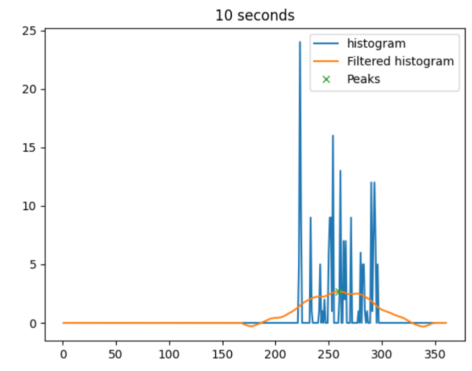

Figure 74: Histogram test 4 at 10 seconds

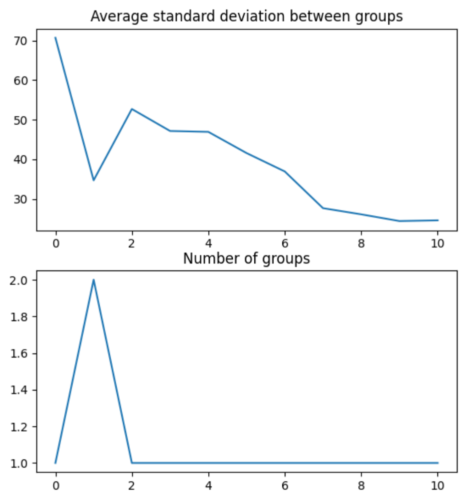

Figure 75: Standard deviation of test 4

B Appendix B

All the code will be available in a private repository controlled by Lars de Kroon and Jeroen van Uffelen. This will be provided on github, access should be requested. It is hosted at https://github.com/BAP2022.