

DIT638 Cyber Physical Systems and Systems of Systems

Mini Project: Final Report

Group 01

Haider Ali, Stanko Jankovic, Osman Osman, Shab Pompeiano

Table of Contents

Table of Contents	2
1. Project Organization	3
Milestone Organization Chart	5
2. Conceptual Ideas and Diagrams	6
Conceptual Ideas	6
Steering Wheel Angle Over Time	6
Infrared Sensors	6
Ultrasonic Sensors	7
Removable Parts Of Camera Feed	7
Camera Feed	8
Conceptual Diagram	9
Insights from video feed analysis	10
3. Software Design	11
Introduction	11
Component Diagram	14
Deployment Diagram	16
4. Software and Integration Testing	18
Test Plan	18
Passing Criteria	19
Steering Angle Calculator	19
Cone Detection	19
Test Cases and Results	20
Steering Angle Calculator	20
Cone Detection	20
Test Conclusion	21
5. References	22
6. Retrospective	23
What did we do well?	23
Where and when did it go wrong?	23
What should we do differently next time?	24
7. Appendix	25

1. Project Organization

- We have begun this project by selecting means of communication which were crucial during these times. We have been using WhatsApp and Zoom for both formal and informal communication. The meetings on zoom would be held very frequently throughout the week, which ensured that everyone was up to date with the project and what was expected of them.
- When it comes to the organization of the project and milestone planning we have been using Trello Boards and GitLab Environment for that. When it comes to Trello Board, we have been using three different boards to keep track of different sections of the project organization. They are: **Meeting Presence Tracker**, **Assignments Trackings** and **Backlog**.

//Done by Haider Ali

- We had assigned a project manager that was taking care of organization of the team, tasks and roles assigning. Apart from that, the project manager was ensuring that the team is on track with milestones and fulfilling deadlines.
- We have decided to apply **Pair Programming** - an Agile Software Development Technique, because we believe that it is the best practice to use during these times where we are limited to online communication only. Considering that there are four team members, by applying pair programming, we have eased up communication, by creating fewer misunderstandings which resulted in better progress.
- As said we have divided into pairs of two. Stanko And Shab worked on HSV Filters and Noise Reduction, Cone Detection, Steering Wheel Angle Calculator and Static Graph Visualization. Osman and Haider worked on Image Segmentation and Frame Cropping, Software and Integration Testing, Live Graph Visualization. Tasks that haven't been listed here were done equally by all team members.
- Each team member focused on certain individual tasks more than others. **Osman's** individual role and responsibility was taking care of Separation of Concerns when it comes to components of the software. **Shab's** individual role was developing mathematical formula for Steering Wheel Angle. **Stanko's** individual role was Project Manager and he was taking care of Team Organization, Gitlab Environment And Trello Organization. **Haider's** individual role and responsibility was maintenance of CI/CD Pipeline and Runners as well as Software Integration.
- //Done by Stanko Jankovic

Meeting Presence Tracker

We had predefined Tuesday, Thursday, and Saturday or Sunday as mandatory meetings, while during the other days of the week we met in case it was needed. Meetings have been focused around the Group Assignments and Software Development. The main idea was discussing and understanding assignments to break them down into smaller tasks which were then assigned to either pairs or individuals. We have tracked the presence of each teammate, topic of the meeting, as well as the time when the meeting began. [18] Displayed in appendix. *//Done by Haider Ali*

Assignments Trackings

For individual assignments, we have created a checklist, where each team member would check his own name after submitting a certain assignment. Group assignments had one list for Uncompleted and another for Completed. The same approach has been used for both types of Bonus Assignments (Individual and Group). [17] Displayed in appendix. *//Done by Haider Ali*

Backlog

Backlog was divided into four stages: "To-Do", "In Progress", "Done" and "Discarded". Each Item was initially added to the "To-Do" list, where team members would be assigned. After that, it is moved to the "In Progress" list. After the completion of the Item, person(s) that worked on it had to write a short description and move it to the "Done" list. When a team decides not to do a certain task it is moved to the "Discarded" list. [16] Displayed in appendix. *//Done by Haider Ali*

Gitlab Environment

When it comes to Gitlab Environment we have been using different tools available for project organization, code organization and overall readability. *Project manager* was making sure that all **Issues** were created, connected to related **Milestones** and that they were **Labeled** properly. [19] [21] Commits were connected to the related Issues and Milestones as well. [20] For each Issues, team member(s) that worked on it were assigned in the description, because it was not possible to assign more than one person using the Assign Gitlab tool. After successfully solving the issue, it is completed and moved to Closed Issues. Displayed in appendix.

//Done by Stanko Jankovic

Milestone Organization Chart

TASK(S)/ISSUES	WEEK(S)								
	1	2	3	4	5	6	7	8	9
Detect necessary parts of the frame									
Divide frame into left and right side									
Find and apply HSV values for yellow and blue colors									
Remove noise from the image									
Detect what's the side are yellow and blue colors at initialization									
Apply further Image Segmentation and divide both left and right frame into 10 segments									
Calculate percentage of white pixels in each segment									
Find Min and Max values for Steering Wheel Angle									
Compute Steering Wheel Angle using the percentage of white pixels									
Unit Testing (Cone Detection, Steering Calculator, Blue on Left)									
Testing the performance and accuracy of the system									
Plot Steering Wheel Angle using csv file (static graph)									
Plot Graph Live using UDP message between Automatic Steering and Graph Visualizer									

MILESTONES: Cone Detection Steering Wheel Angle Calculator Software Testing Graph Visualization

- We had 4 milestones that were related to the main project and they are: **Cone Detection, Steering Wheel Angle Calculator Software Testing and Graph Visualization**. As it's presented in the chart, each Item was a Task/Issue of the Software that we have been developing and also part of different Project Milestones as shown above. Every week had certain task(s) and those tasks were divided between the team. When the task is completed, Milestone that it belongs to and week where it was completed were marked with specific color. In case a Task is reopened later during the project development, another week gets marked as displayed. Milestones and Tasks(Issues) were declared in the Gitlab Environment as it was mentioned in the previous paragraph. Displayed in appendix. [19] [20]

//Done by Stanko Jankovic

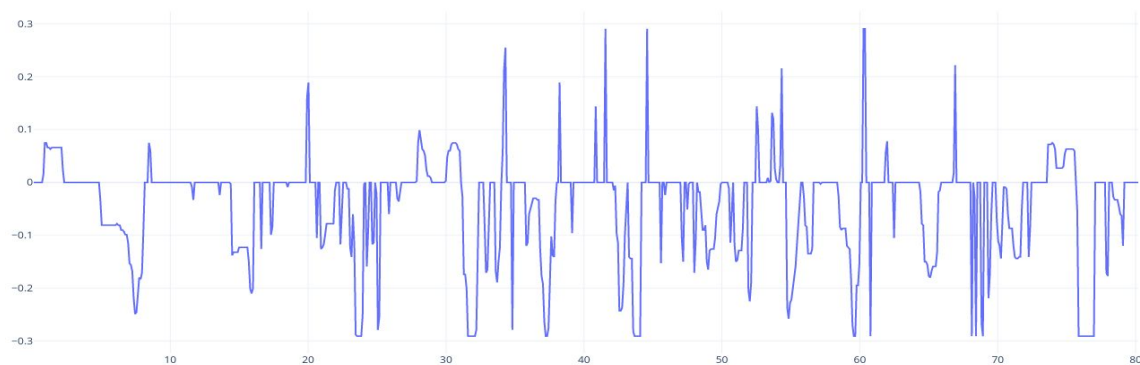
2. Conceptual Ideas and Diagrams

Conceptual Ideas

To be able to determine a steering wheel angle, conceptually there would be many different ways considering the various sensors mounted to the Kiwi Car.

During the brainstorming sessions our team came up with different ideas regarding how to make use of the provided sensor data.

Steering Wheel Angle Over Time



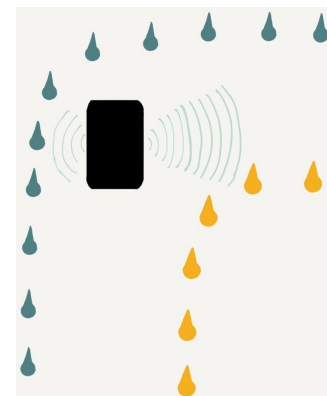
The graph above shows how the Steering Wheel Angle changes over time in one of the recordings where after analyzing the video feed we understood that a **negative value** corresponds to a turn to the **right** while a **positive value** means that the car turns to the **left**. Therefore, we made a conclusion that our algorithm should then for the same recordings plot a similar graph.

//Done by Haider Ali

Infrared Sensors

Two infrared sensors are mounted on the Kiwi Cars, one on each side.

Given their lateral position the infrared sensors could hypothetically be used to measure the distance between the cones, respectively to the left and right side of the car and the road.

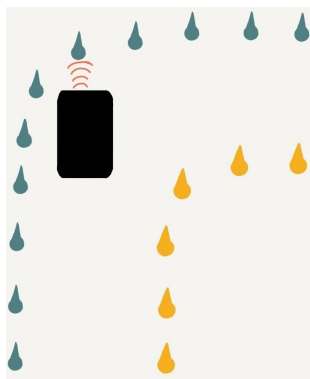


By doing so and calculating the distance, we thought it would then be possible to come up with a formula that would calculate an optimal Steering Wheel Angle.

There are two potential problems with that approach though. First of all, the readings are not so accurate, and secondly, the implementation would be very complicated, as there are empty spaces between the cones to account for. Therefore, the use of these sensors would not be feasible.

//Done by Haider Ali

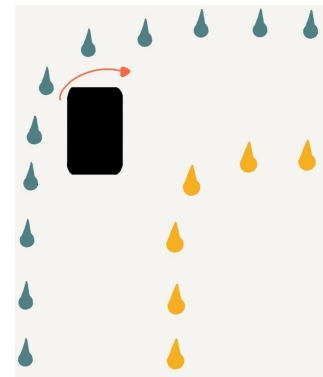
Ultrasonic Sensors



When it comes to the ultrasonic sensors, there are two of them and they are mounted to the front and the back of the Kiwi Car. We initially thought of using the readings from the ultrasonic sensor mounted on the front for making a turn during emergency situations. For example, in case a Kiwi Car gets too close to the cone (or any other object) we would maximise the steering wheel angle in the direction the car was already going. We

thoroughly analyzed the recordings and realized that there were not many situations where this happened. Therefore, we decided not to implement it.

//Done by Haider Ali



Removable Parts Of Camera Feed

We observed that some parts of the images provided by the camera feed should not be taken into consideration, i.e. the parts where no cone ever appears (top part of the image and bottom part representing the car).

Not only do these parts interfere with the cones once the HSV filter is applied as e.g. the car has a yellow wire that would be considered as a cone once the image is filtered, but also analyzing those parts in the image processing would use more computational power for no reason.

//Done by Haider Ali



Camera Feed

The feed received from the camera mounted on top of the Kiwi Car is the most useful data available to us, because it is the most reliable type of data and it helped us come up with the most feasible and at the same time effective algorithm to implement.



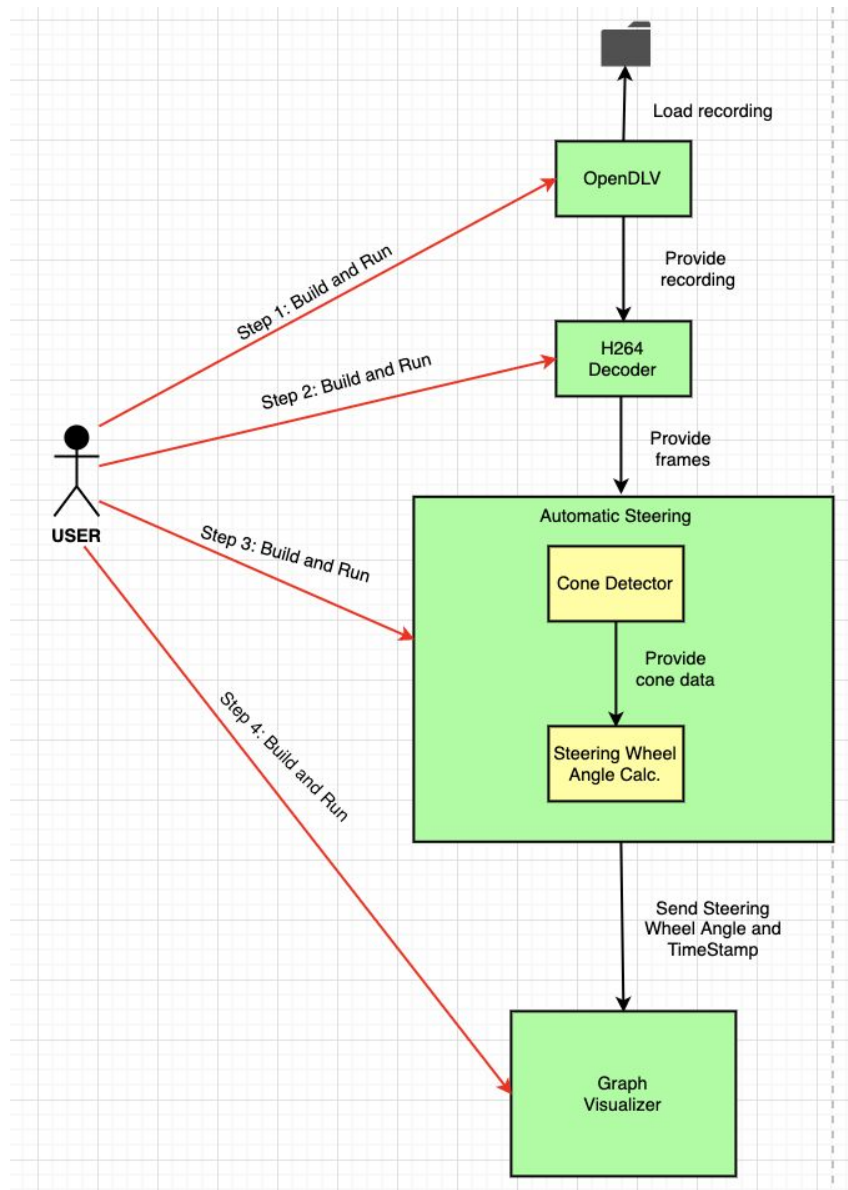
Assuming that the cones on each side are either yellow or blue (i.e. there will never be cones of the same color both on the right and left side of the car) by applying HSV filters [2] to isolate the blue and yellow colors of the cones it would be possible to then determine a steering wheel angle.

Our initial conceptual idea was to determine the steering wheel angle during a turn, by considering the size of the cones. For example, the blue cones are on the left side and a turn to the right has to be taken; the blue cones detected would be bigger on the left side of the video and becoming smaller and smaller the more we move to the right of the video because the cones to the left get closer to the camera.

Our second and final approach was to cut the useful part of the video containing the cones, apply the HSV filter, and then divide the image into many vertical segments. Once the image is divided into segments, a ratio between the quantity of segments containing blue cones and yellow cones is calculated. Then we would calculate a Steering Wheel Angle accordingly to that data.

//Done by Haider Ali

Conceptual Diagram



As it is presented in the diagram above, there are four microservices in the entire system and a User that has **4 steps** to compile the Software completely. Initially, **OpenDLV** needs to be Built and Run, where the **Recording** is loaded. Afterwards, User Builds and Runs **H264 Decoder** that provides frames to our main microservice **Automatic Steering**. Third step is Building and Running our **Automatic Steering**

microservice that uses **OpenCV** and **libclupon** libraries. Automatic Steering produces values that are written in the "csv" file that contains *Steering Angle* and *Timestamp*.

When it comes to the fourth step, User decides between two options: Static Visualization or Live Visualization. If it's Live Visualization, it requires Building and Running our second microservice, **Live Graph Visualizer**. It communicates with **Automatic Steering** and plots the values as long as it's running. Another option, **Static Graph Visualization** depends on reading values from a csv file that is provided by the **Automatic Steering** after completing execution. Final product is **Steering Wheel Angle** with the optional visualization.

//Done by Stanko Jankovic

Insights from video feed analysis

No cone detected

By analysing the video frames from the camera feed we noticed that when a sharp turn is taken, all cones that the car sees are of the same color. Sometimes the car would get so close to the cones that in some frames it does not detect any cone, which means that the car is facing the space between two cones.

In this particular case we decided to implement a clause in the algorithm that would maintain the previous steering wheel angle calculated, in case there are no cones detected in the image.

//Done by Stanko Jankovic

Detection of cones on the sides (beginning of video feed)

In order to detect what color of the cones are located on which side of the image, we have analyzed the recording and we discovered that sometimes it would be difficult to decide correctly on which side the cones are yellow and blue. The main reason would be due the fact that the video from the camera feed fades in when the car is turned on. Therefore, we decided to periodically analyze the recording during the execution for a small period of time [4].

//Done by Shab Pompeiano

Suppression of low values of steering wheel angle

By analysing the video feed we figured out that the car had a threshold for the car to turn, this means that the adjustments made to the direction of the car with respect to the cones were not fine.

This translates in our implementation to a suppression of small values of steering wheel angle.

//Done by Shab Pompeiano

3. Software Design

Introduction

Steering wheel angle

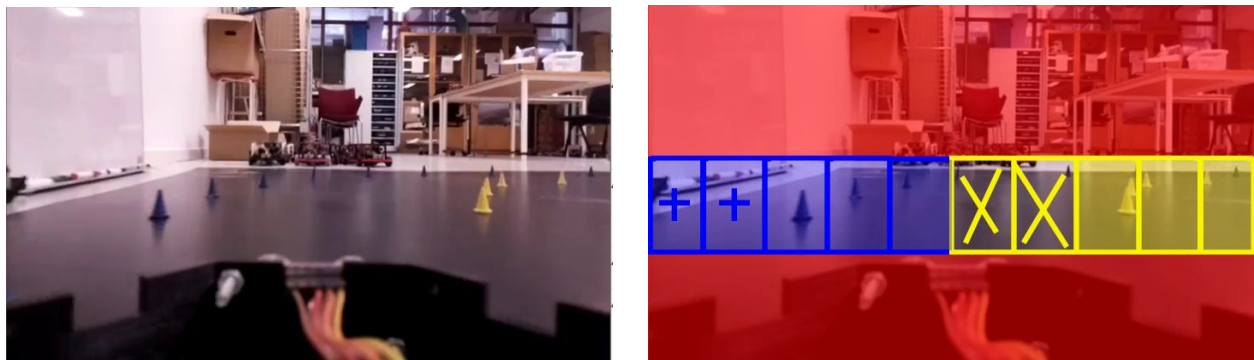
In the previous section we mentioned briefly how our algorithm works, let's dive deeper into it.

To compute the steering wheel angle we have developed two algorithmic aspects. Two of them have their own functionality, but they are connected with one another.

First of all we have been working on a Cone Detection Algorithm. We needed a way to identify cones in order to build a Steering Wheel Angle Algorithm. Initially, we determined what are the important parts of the frame that we need to analyze. Afterwards, we established what color is located on which side of the frame. By detecting where the colors are located in the frame, we had successfully detected cones, as well as their organization for the given recording.

We detect the colors using different HSV range values [2] for the two colors (blue and yellow cones) and separate them from the background. To the image filtered with HSV erosion and dilation filters are applied to the image to reduce the noise [3].

After detecting the cones, we built up an algorithm to calculate the Steering Wheel Angle. We first divided the analyzed frame into segments and looked for white pixels that represent cones in each one of them [6]. When we detected a cone in the segment closest to the middle of the frame we changed and adjusted the angle accordingly.



Namingly if the picture above is the case, the rectangular slots are the different segments that are created from the elaboration of the image. Each blue rectangle will be analyzed to determine if it contains a blue cone (e.g. if the HSV image has enough

amount of white space for the image to be considered as containing a cone [6]) and respectively each yellow rectangle will be analyzed to determine whether it contains a yellow cones.

We can notice that the last three (from left to right) of the blue slots contain blue cones but the first two slots (the ones containing a plus sign) will be also considered containing blue cones. We can then observe that the first three yellow slots (from the right to the left in this case) contain yellow cones. The yellow slots crossed out instead do not contain yellow cones and will be labeled as such.

From this point a delta Δ_{cones} will then be calculated as the difference of the quantity of slots containing cones on the right side of the image Q_{right} and the quantity of slots containing cones on the left side of the image Q_{left} . Now from all the data that we have, we only need a function that could best fit the behaviour of the car. From one of our insights we have realized that with respect to Δ_{cones} the steering wheel angle has to increase slowly for small values of Δ_{cones} and increase faster for higher values of Δ_{cones} .

We tried different equations [1], from the many x^5 and $\sin()$ and we found out that the sin function was calculating more accurate values of steering wheel angle.

By using plotting instruments we found the best fit for our case and we found the following function:

$$g(x) = \sin\left(\frac{x^5}{y^5}\right) * c$$

Where

$g(x)$ represents the steering angle to apply S

x represents the retrieved delta Δ_{cones}

c is a constant

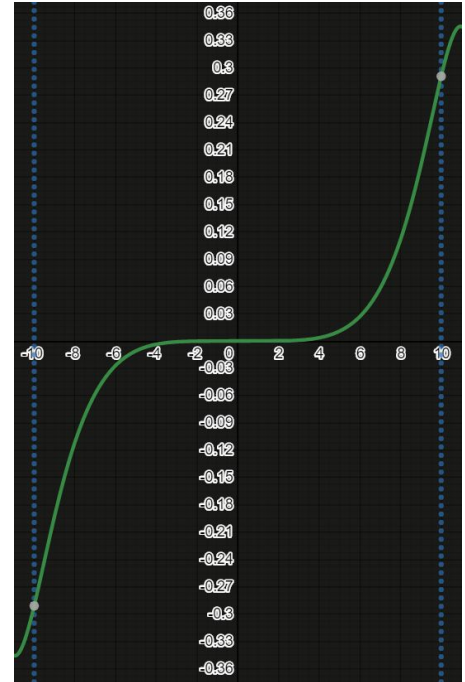
y is the number of segments per side seg

And the $\sin()$ function is calculated in radians

The formula applied to our domain becomes then:

$$S = \sin\left(\frac{\Delta_{cones}^5}{seg^5}\right) * c$$

The constant c is then calculated by feeding the above formula with the maximum possible value of delta (10 in our case which is the number of segments) so $\Delta_{cones} = 10$ and the amount of segments per side which we set as 10 so $seg = 10$ and the result of



the formula shall be the maximum steering wheel angle as at the maximum amount of delta we shall have the maximal turn so $g(x) = 0,290888$

In our case c would be calculated as follows:

$$0.290888 = \sin\left(\frac{10^5}{10^5}\right) * c$$

$$c = \frac{0.290888}{\sin(1)} \approx 0.3457$$

That leaves us with the final formula:

$$S = \sin\left(\frac{\Delta_{cones}^5}{seg^5}\right) * c$$

Or

$$steering\ to\ apply = \sin\left(\frac{(Q_{right} - Q_{left})^5}{\#segments^5}\right) * c$$

Where

$$c = \frac{max\ steering}{\sin(1)}$$

For this example an estimation of the steering would be:

$$steering\ to\ apply = \sin\left(\frac{(3-5)^5}{5^5}\right) * 0.3457 = \sin\left(\frac{-2^5}{5^5}\right) * 0.3457 \approx -0,0035$$

Which suggests to us that the car is going to turn slightly to the right.

//Done by Shab Pompeiano (from page 11 until here)

Cone side

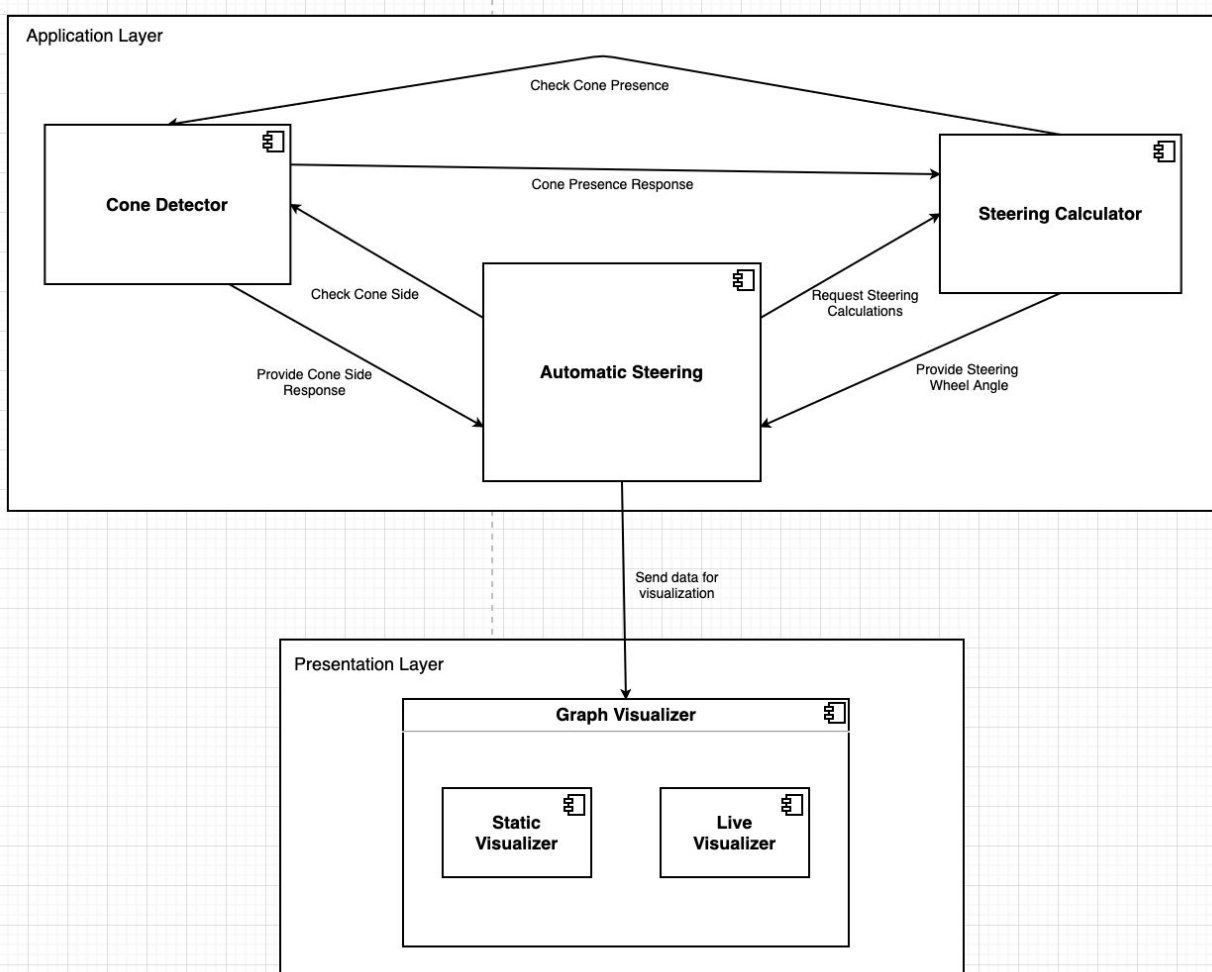
A variable called BLUE_IS_LEFT is kept updated and gives us the information about whether the cones on the left are blue. Precisely if BLUE_IS_LEFT equals to 1, the cones on the left are blue and the cones on the right are yellow, otherwise if BLUE_IS_LEFT equals to -1 the cones on the left are yellow and the cones on the right are blue.

Another algorithm periodically (every half a second) checks whether the cones on the left are yellow (and the ones on the right blue) or blue (and the ones on the right yellow). Every half of a second the algorithm applies the yellow cone filter to the image and the blue cone filter respective on the sides where the color should be the opposite, so i.e. if BLUE_IS_LEFT equals to 1 and the cones on the left are blue and the cones on the right are yellow, a yellow cone filter is applied to the left part of the image (where the cones should be blue) to detect yellow cones and consequently a blue cone filter is applied to the left part of the image (where the cones should be yellow) to detect blue cones. If these filters detect the cones the variable BLUE_IS_LEFT will be set to -1 in this case (in reality it is changed by multiplying it with -1 to invert it) and we can say that

the cones on the left are now yellow and the cones on the right are now blue, e.g. they have been swapped. //Done by Shab Pompeiano

When it comes to the Implementation View of our Software, we have decided to use Component and Deployment Diagrams to visualize our System.

Component Diagram



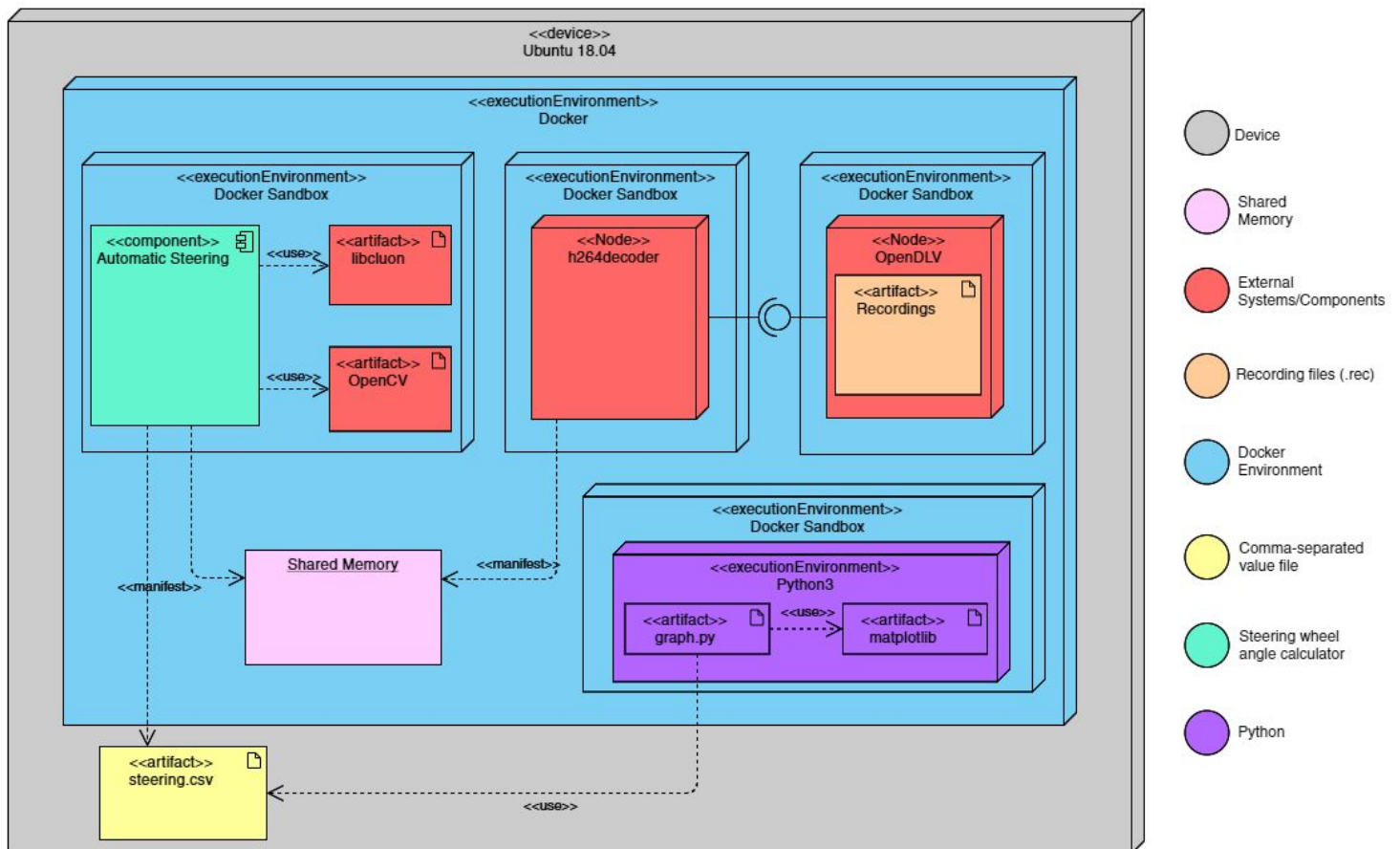
Our System has two layers, **Application** and **Presentation**. Within the Application Layer, we have our main microservice, which consists of three components. Main Component is Automatic Steering which communicates with the Steering Calculator and Cone Detector, as well as Graph Visualizer in the Presentation Layer.

Initially, Automatic Steering **Checks Cone Sides** with Cone Detector. Cone Detector runs "decideSideCones" to establish which color is located on which side of the frame

(by applying methods: "applyBlueFilter", "applyYellowFilter" and "checkConePresence"). It then sends the **Cone Side Check Response**, which is the variable BLUE_IS_LEFT. Automatic Steering then **Requests Steering Calculations** from Steering Calculator. After that, Steering Calculator **Checks Cone Presence** using Cone Detector. Cone Detector applies "checkConePresence" and returns boolean as a **Cone Presence Response** to the Steering Calculator. Steering Calculator runs "calculateSteering" where it first does Image Segmentation and divides the frame into segments and then computes the Steering Wheel Angle accordingly. After successfully computing it, the Steering Calculator **Provides Steering Wheel Angle** to Automatic Steering component. When it comes to connecting to Graph Visualizer, we have two options. First one is Static Visualization, where Automatic Steering stores all the values in a **csv file** that is forwarded to the Static Visualizer (graph.py) that plots the graph. Second option is where Automatic Steering connects to another microservice called Live Visualizer (liveGraph.py) that automatically plots the data on the graph at the same time as the first microservice is being run. These two microservices communicate via UDP messages.

//Done by Stanko Jankovic

Deployment Diagram



Our **Microservice** is deployed on a **Linux** machine (the recommended edition is **Ubuntu** version 18.04) in a **Docker** environment.

Thanks to Docker it is possible to create different sandboxes to run applications in a clean, sealed environment.

In total, **three** sandboxes have to be created. One is needed for our microservice to run, while another two sandboxes are needed to run OpenDLV and the H264decoder.

The OpenDLV sandbox will run the **OpenDLV Vehicle View** which is used to select a file to play from a list of **recordings**. The files are **.rec** files containing previous recordings from a **Kiwi Car** which give access to the video feed of a camera mounted on top of the car facing the front together with other measurements from the different sensors (namely infrared sensors, ultrasonic sensors and others).

The sandbox running the **H264decoder** will get the data coming from a live playing recording from the OpenDLV and decode the images coming from the camera feed in

H.264 format to **ARGB**. The ARGB decoded image will then be saved temporarily into a **Shared Memory**.

Our microservice **Automatic Steering** will then in a loop read the data from the Shared Memory and make decisions based on that data.

To do that it uses the external libraries "**libcluon**" and "**OpenCV**". A csv file called **steering.csv** containing all the generated steering angles for each frame with timestamp is then generated by our microservice when the program is exited.

A **Python** script "graph.py" is then called to plot the data contained in the comma-separated values (csv) file with the help of the **matplotlib** library.

While the other option is simultaneous plotting of the graph with the use of UDP message between main microservice **Automatic Steering** and **Live Graph Visualizer** - "liveGraph.py".

//Done by Shab Pompeiano

4. Software and Integration Testing

Test Plan

During the development phase we have come up with a number of functions which uses the data that comes from the sensors and the car, to make the car manoeuvre between the cones. We decided to make the algorithm using the video recording from the open-dlv microservice. However, it is crucial that the software gets tested at an early stage in order to make sure that any errors we encounter get mitigated in due time.

We will mainly be testing the main functions of our main microservice, making sure that they return the expected results. Our opencv microservice is depending on the functionalities included in ConeDetection.cpp and SteeringAngleCalculation.cpp. The functions in these files will be tested to see if the video frames received from opendlv microservice will be correctly analysed as planned. Specifically to check for example if there are cones present in the video frames, or if the blue cones are on the left or right side or if the SteeringAngle is positive or negative. The reasons we specifically chose these sections to be tested was due to them being crucial and significant for the final goal of our project, and for being able to achieve both unit and integration testing in our project.

The resources used for testing, for example the image frames are directly taken or extracted from different parts of the video recordings provided in the course material.

Functions to be tested for unit-testing:

- *checkConePresence (cv::Mat image)*
This function receives the small cropped section of the original frame in HSV format, containing only the part related to the cones. It will return TRUE if there are white cones in the frame and return FALSE if the frame is totally black.
- *decideSideCones (cv::Mat img, int BLUE_IS_LEFT):*
This function is done to determine which cones should be looked out for on each side of the frame. If the blue cones are detected on the right side it should return 1 otherwise -1.

Function to be tested for integration-testing:

- *calculateSteering (cv::Mat img, int BLUE_IS_LEFT)*

This function calculates the steering wheel angle and returns it as a “double” value. Since the exact value of the steering is not predictable we will only test if the values calculated from the functions are negative (Right turns) or positive (Left turns) depending on the frame being provided to the function.

This functionality is used for integration testing because it uses *checkConePresence(cv::Mat image)* which represents another standalone functionality of the system which is being integrated and tested together to produce the final steering wheel angle.

Passing Criteria

Steering Angle Calculator

- *calculateSteering(cv::Mat img, int BLUE_IS_LEFT)*

Two different images, imported from the video feed where one shows a right turn and the second shows a left turn will be passed in the parameter along with an integer indicating whether blue cones are on the left or right side of the screen.

The thing that will be deemed as a passed test if it returns a negative number for the right turn image and a positive number for the left turn image.

Cone Detection

- *decideSideCones (cv::Mat img, int BLUE_IS_LEFT)*

To pass the testes, the method must return -1 if the image passed has blue cones on the right and vice versa.

- *checkConePresence (cv::Mat image):*

The method will be considered a passed method if it correctly determines if there are cones in the screen or not.

//Done by Osman Osman

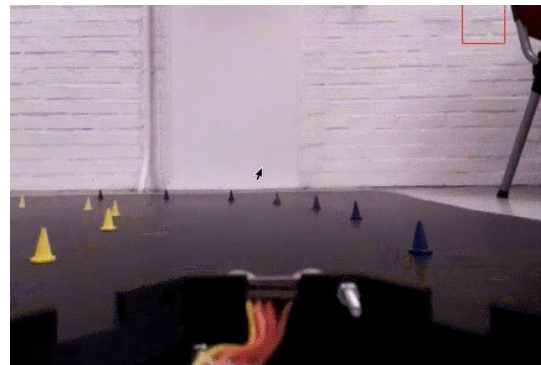
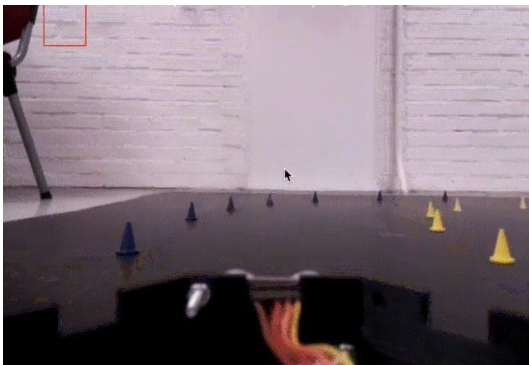
Test Cases and Results

Steering Angle Calculator

1. `calculateSteering(image,1)` and `calculateSteering(image,-1)`

For these test cases we pass these images to the function and 1 (which indicates that the blue cones are on the left side) or -1 (which indicates that the blue cones are on the right) to see whether it will return a negative number or a positive value.

Both functions pass the tests by returning the expected outputs. **[PASSED]**

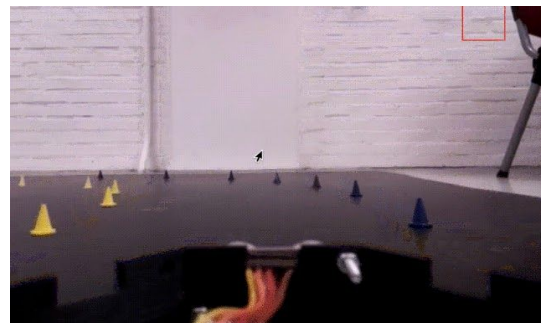


//Done by Haider Ali

Cone Detection

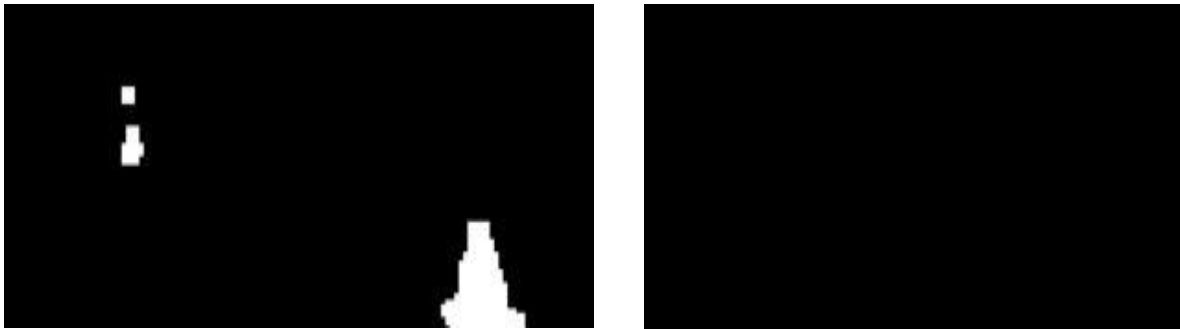
1. `decideSideCones(image, 1)` and `decideSideCones(image, -1)`

We passed both of these images to the function in order to determine which side the blue cones are on. The returned value from the right picture is 1, and for the picture on left -1 was returned. Therefore, the test cases meet our passing criteria. **[PASSED]**



2. checkConePresence(image)

For this test case we pass these images where the one on the left shows the presence of cones and the other one does not. The function passed the test cases by returning false when there was not any cone in the frame and it returned true when there was. **[PASSED]**



//Done by Osman Osman

Test Conclusion

These test cases show that the main functions of the program are able to give out the correct results based on what is being passed into them. We have shown that each individual testing module works on their own and they work and provide accurate results when they are integrated together. Therefore, we believe that we have developed our software properly and have been able to achieve the overall goal of this project.

//Done by Osman Osman

5. References

1. Plotting functions on graphs from Desmos (<https://www.desmos.com/calculator>)
2. HSV https://docs.opencv.org/3.4/da/d97/tutorial_threshold_inRange.html
3. Apply Erosion and Dilation to images for noise reduction
https://www.tutorialspoint.com/opencv/opencv_erosion.htm
4. Get current time from epoch in milliseconds
<https://stackoverflow.com/questions/19555121/how-to-get-current-timestamp-in-milliseconds-since-1970-just-the-way-java-gets>
5. Gamma correction on image
https://docs.opencv.org/3.4/d3/dc1/tutorial_basic_linear_transform.html
6. Count amount of white pixels in the image
<https://stackoverflow.com/questions/10866741/how-to-count-image-pixel>
7. Dockerize your Python Application
<https://runnable.com/docker/python/dockerize-your-python-application>
8. Append CSV <http://www.cplusplus.com/doc/tutorial/files/>
9. Plotting MatLab <https://pypi.org/project/matplotlib/>
10. Plotting Python <https://pythonprogramming.net/loading-file-data-matplotlib-tutorial/>
11. Docker Intro: Building a python 3 image
<https://medium.com/@yvescallaert/docker-intro-building-a-python-3-image-62031d0b7e39>
12. Python delete element from list
<https://stackoverflow.com/questions/4426663/how-to-remove-the-first-item-from-a-list>
13. Catch2 Testing
<https://github.com/catchorg/Catch2/blob/master/docs/tutorial.md#scaling-up>
14. Libcluon UDP https://chrberger.github.io/libcluon/classcluon_1_1UDPSender.html
15. Python : UDP Client and Server
<https://pythontic.com/modules/socket/udp-client-server-example>
16. Trello Backlog: <https://trello.com/b/hzqF9KcK/backlog>
17. Trello Assignment Tracking: <https://trello.com/b/nzblnjcX/assingments-trackings>
18. Trello Meeting Presence Tracker: <https://trello.com/b/RMSjlotg/meeting-presence-tracker>
19. Issues Gitlab:
https://git.chalmers.se/courses/dit638/students/group_01/-/issues?scope=all&utf8=%E2%9C%93&state=all
20. Milestones Gitlab:
https://git.chalmers.se/courses/dit638/students/group_01/-/milestones?sort=due_date_desc&state=all
21. Labels Gitlab: https://git.chalmers.se/courses/dit638/students/group_01/-/labels

6. Retrospective

The communication between the team members for an online based team-project like this is crucial in every aspect. The misunderstandings or miscommunications can lead to some serious problems regarding the division of work assigned to each team member and the actual work performed by each of them.

What did we do well?

We planned to do pair programming, so most of the tasks were performed while working in groups of two. We had 3 fixed group meetings per week to catch up with each other and to divide further work or tasks into smaller assignments per team pair. The flow of communication within the team was fluent and effective creating a good and friendly work environment.

After consulting with some other groups, where communication problems arose we understand how miscommunication in the teams can lead to unfinished work by some team members, which has been procrastinated and never communicated to other team members. To avoid this problem use of the Trello Boards to track each group members' individual progress for the assigned tasks. Apart from Trello Boards, with the help of GitLab's functionalities, such as Milestones and Issues, we have been keeping track of projects' progress as well.

Where and when did it go wrong?

On the other hand, no matter how good a project goes there will always be downsides and there's always room for improvements. One of the problems some team members faced was using VirtualBox to run Ubuntu 18.04 on. Due to the severe latency issues with the VirtualBox it was impossible to work in a fluent way and at the same pace as other team members. Fortunately, this problem was realised at an early stage of the project which allowed us to make the appropriate changes in our planning. Essentially, this problem was mitigated by running Ubuntu on dual boot instead of VirtualBox.

Another problem which we lost a lot of time on was with fixing gitlab-runners. We tried to debug our code and understand the errors shown during the builds. But it was merely a problem for the shared runners on gitlab which in 75% of the cases crashed the build. This problem was persistent until a few weeks ago and was mitigated with the use of Group runners instead.

Additionally, the overall goal of the project was not clear enough until the first group meeting with the teacher. Which meant that before that, half of the time we had been focusing on some irrelevant stuff.

What should we do differently next time?

As mentioned earlier there is always room for improvement. If we were to have another extension of this project there would be a lot of things we would have done differently. We would definitely ask more questions, especially if something seems to be unclear. This is because when you do a lot of significant work based on wrong information it may put you in a dire situation where you have to backtrack and do it correctly, which can cost a lot of important time.

Furthermore, we would also meet in person rather than via Zoom if the pandemic stops. This would be more efficient as things like pair programming will mean that there will be less mistakes made. Doing a group meeting in a room creates more focus because each team member would have no other choice but to focus because of being observed by the team members. On the contrary, having a group meeting via Zoom may still be effective however it is not on the same effectiveness as the face-to-face meeting, as someone's attention may be diverted to something else which decreases the productivity of the meeting.

//Done by Osman Osman

7. Appendix

Test Cases

```
ConeDetection cd;
SteeringCalculator st;
cv::Mat image;

TEST_CASE("Blue on left or right","[test1]") {
    image = cv::imread("../MD_Material/img3.jpg", -1);
    REQUIRE(cd.decideSideCones(image, 1) == 1);

    image = cv::imread("../MD_Material/img3-M2.jpg", -1);
    REQUIRE(cd.decideSideCones(image, -1) == -1);
}

TEST_CASE("Cone presence","[test2]") {
    image = cv::imread("../MD_Material/noCones.jpg", -1);
    REQUIRE(cd.checkConePresence(image) == false);

    image = cv::imread("../MD_Material/withCones.jpg", -1);
    REQUIRE(cd.checkConePresence(image) == true);
}

cv::Mat image;

TEST_CASE("SteeringCalculator with blue cones on left","[test3]") {
    image = cv::imread("../MD_Material/img3.jpg", -1);
    REQUIRE(st.calculateSteering(image,1) <= 0);
}

TEST_CASE("SteeringCalculator with blue cones on right", "[test4]") {
    image = cv::imread("../MD_Material/img3-M2.jpg", -1);
    CHECK(st.calculateSteering(image, -1) <= 0);
}
```

Trello Board

Assignment Trackings

Assigments Trackings

Group 01 - DIT638 - 2020 Free Team Visible Join Board Bulk Actions (Authorization Needed) Butler Show Menu

Legend

- INDIVIDUAL: Individual assignemnt
- GROUP: Group assignment
- BONUS: Bonus assignment
- + Add another card

Regular Individual Assignments

- Assignment: Template 0/4
- DONE! INDIVIDUAL: provide your student CID (NOT your password!!!) to add you to Chalmers GitLab 4/4
- DONE! INDIVIDUAL: set up your development environment 4/4
- DONE! INDIVIDUAL: arrange yourself in groups 4/4
- DONE! INDIVIDUAL: Build an example project 4/4
- DONE! INDIVIDUAL: Deploy and make a video 4/4
- + Add another card

Regular Group Assignments

- Submit your final report 22 May
- Submit the URL to the Docker image of your final implementation 25 May
- Final examination 27 May
- + Add another card

Completed Regular Group Assignments

- DONE! GROUP: Test the performance of your approach 21 May
- DONE! GROUP: Book the examination slot 15 May
- DONE! GROUP: Present your progress 8 May
- DONE! GROUP: Submit your progress presentation 4 May
- DONE! GROUP: Book a presentation slot 30 Apr
- DONE! GROUP: Present your progress 29 Apr
- + Add another card

BONUS Assignments

- BONUS INDIVIDUAL: Build an example project - Bonus 16 Apr 2/4
- BONUS GROUP: Automate the performance testing for your approach in GitLab's CI/CD pipeline - BONUS 29 May
- BONUS DONE! GROUP: Set up CI/CD - Bonus 23 Apr
- + Add another card

Meeting Presence Tracker

Meeting Presence Tracker

Group 01 - DIT638 - 2020 Free Team Visible S H Invite Calendar Butler Show Menu

Planned Meetings

- Topic: Template 0/4
- + Add another card

Tuesday Meetings

- Code of conduct 4/4
- Revise Code of Conduct 8 Apr 4/4
- Revision of Concept Presentation 21 Apr 4/4
- Assignment Discussion 28 Apr 4/4
- Progress Check 12 May 3/4
- Final Report 19 May
- + Add another card

Thursday Meetings

- Set Up CI/CD 9 Apr 4/4
- Concept Presentation Preparation 16 Apr 4/4
- Presentation 23 Apr 4/4
- Software Development Discussion 30 Apr 4/4
- Final Report Discussion 14 May 4/4
- Final Report 21 May
- + Add another card

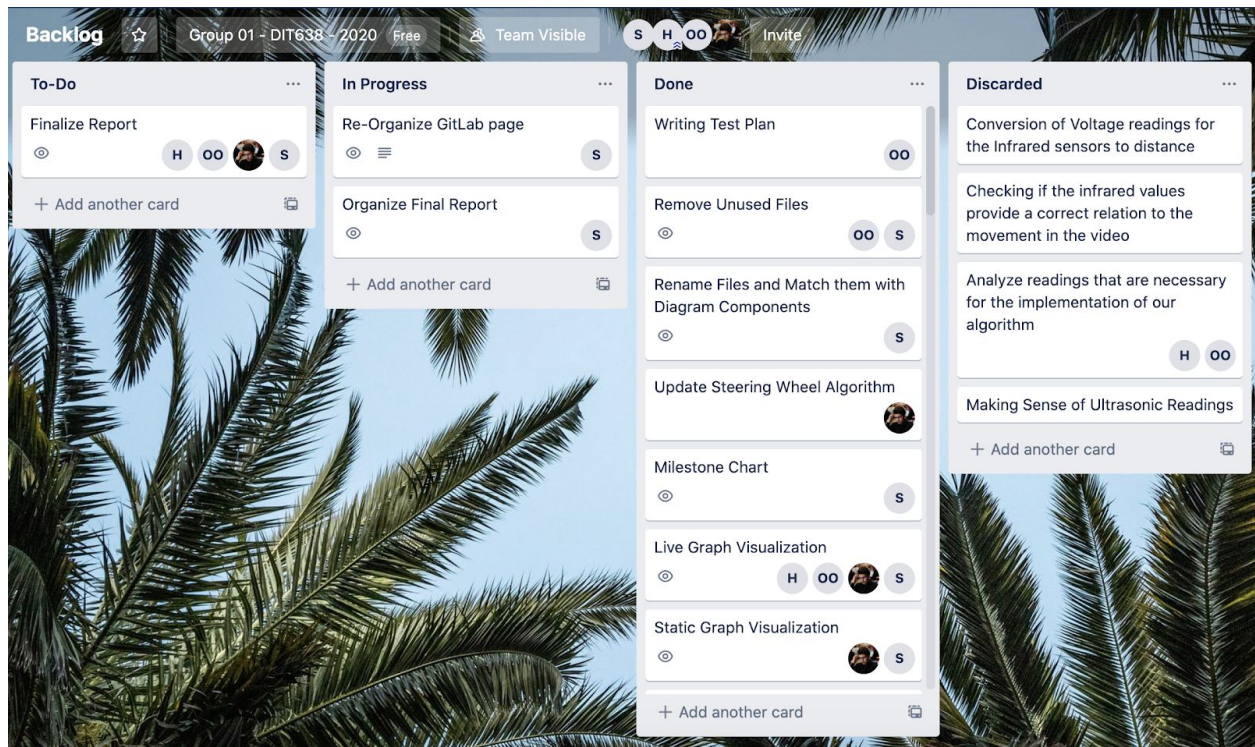
Weekend Meetings

- Assignments Planning 2/4
- Concept Planning On Zoom. 12 Apr
- Concept Presentation 18 Apr 4/4
- Concept Presentation 2 19 Apr
- Distribution of tasks 26 Apr 4/4
- Project Progress Meeting 2 May 4/4
- Progress Presentation Meeting 3 May 4/4
- Regular Meeting
- + Add another card

Additional Meetings

- BONUS Assignemnt - set up CI/CD 22 Apr
- Project and Organizational Planning 27 Apr 4/4
- Discussion of Feedback from Progress Presentation 4 May 4/4
- Final Report 18 May
- + Add another card

Backlog



Gitlab Environment

Issues

Open 1

Closed 26

All 27

Milestones

courses > ... > Student Repositories > group_01 > Milestones

Open 4Closed 3All 7

Filter by milestone name

Due later

New milestone

Multi-platform builds

May 13, 2020–May 28, 2020

Closedcourses / DIT638 / Student Repositories / group_01

1 Issue · 1 Merge Request

100% complete

Reopen Milestone

Graph Visualisation

May 1, 2020–May 25, 2020

courses / DIT638 / Student Repositories / group_01

2 Issues · 2 Merge Requests

100% complete

Close Milestone

Testing Methods for the functions developed

Apr 1, 2020–May 25, 2020

Closedcourses / DIT638 / Student Repositories / group_01

2 Issues · 5 Merge Requests

100% complete

Reopen Milestone

Steering Wheel Angle Calculator

Apr 20, 2020–May 25, 2020

courses / DIT638 / Student Repositories / group_01

3 Issues · 6 Merge Requests

100% complete

Close Milestone

Cone Detection

Apr 20, 2020–May 25, 2020

courses / DIT638 / Student Repositories / group_01

9 Issues · 13 Merge Requests

100% complete

Close Milestone

Final Report

Mar 24, 2020–May 22, 2020

6 Issues · 2 Merge Requests

83% complete


Close Milestone

Labels

Other Labels			
assignment	Code and documentation related to assignments Issues · Merge requests	Project label ☆ ✎ ⋮	Subscribe
bug	Problem/ error with the code Issues · Merge requests	Project label ☆ ✎ ⋮	Subscribe
code readability	Code updates to make code more readable and cleaner Issues · Merge requests	Project label ☆ ✎ ⋮	Subscribe
documentation	Issues · Merge requests	Project label ☆ ✎ ⋮	Subscribe
enhancement	Additional feature Issues · Merge requests	Project label ☆ ✎ ⋮	Subscribe
features	Features and additions to every part of the system Issues · Merge requests	Project label ☆ ✎ ⋮	Subscribe
testing	Testing related issues Issues · Merge requests	Project label ☆ ✎ ⋮	Subscribe

Commits Organization Overview

Merged

Opened 1 week ago by  shabp

Edit

Report abuse

Resolve "Detect cone in segment"

Overview 0


Commits 7


Pipelines 14



Changes 1

Closes #17 (closed) Assignees @shabp and @stankoj

Edited just now by stankoj

 Request to merge 17-detect-cone-in-segment into master

 Detached merge request pipeline #12325 passed for b00ae318

 Merged by  shabp 1 week ago

Revert

Cherry-pick

The changes were merged into master with dcba8f29

The source branch has been deleted

Closed #17 (closed)

0 Assignees

Edit

None - assign yourself

Milestone

Edit

Cone Detection

Time tracking

?

No estimate or time spent

Labels

Edit



features

Lock merge request

Edit

Unlocked


2 participants

Notifications

☒

Reference: courses/dit638/stu...



Merged

Opened 21 hours ago by osma

EditReport abuse

Live graph

Overview 0Commits 2Pipelines 2Changes 2

Closes #27 (closed) Assignees @haiali and @osma

Edited just now by stankoj

Request to merge liveGraph into master

Detached merge request pipeline #13246 passed for 0bbe8738

Merged by osma 21 hours ago

RevertCherry-pick

The changes were merged into master with 487a7028

The source branch has been deleted

Mentions #27 (closed)

0 Assignees

None - assign yourself

Edit

Milestone

Graph Visualisation

Edit

Time tracking

No estimate or time spent

Labels

features

Edit

Lock merge request

Unlocked

Edit

3 participants

Notifications

☒

Reference: courses/dit638/stu...

Merged

Opened 5 days ago by stankoj

Edit

Fix Yellow Color and Steering Wheel Angle

Overview 0Commits 4Pipelines 4Changes 3

Closes #26 (closed) Closes #25 (closed)

Edited 2 days ago by stankoj

Request to merge 20-fix-yellow-color... into master

Detached merge request pipeline #12581 passed for c7cb9266

Merged by stankoj 5 days ago

RevertCherry-pick

The changes were merged into master with 81bcf4c9

You can delete the source branch now

Delete source branch

Closed #20 (closed)

Mentions #25 (closed) and #26 (closed)

Assignee

stankoj

@stankoj

Edit

Milestone

Steering Wheel Angle Calculator

Edit

Time tracking

No estimate or time spent

Labels

enhancement

Edit

Lock merge request

Unlocked

Edit

1 participant

Notifications

☒

Reference: courses/dit638/stu...

Milestones Organization Overview

courses > ... > Student Repositories > group_01 > Milestones > Steering Wheel Angle Calculator

Open

Milestone Apr 20, 2020–May 25, 2020

Edit

Promote

Close milestone

Delete

Steering Wheel Angle Calculator

Algorithm for calculating angle change needed so that the car doesn't hit the cones.

All issues for this milestone are closed. You may close this milestone now.

Issues 3

Merge Requests 6

Participants 2

Labels 3

Unstarted Issues (open and unassigned) 0

Ongoing Issues (open and assigned) 0

Completed Issues (closed) 3

Fix Steering Wheel Angle (When No Cones Visible)
#25 bug

Fix Yellow Color and Steering Wheel Angle
#20 enhancement

Determine values corresponding to the Camera Feed readings in the recordings.
#4 features

100% complete

Start date Apr 20, 2020

Due date May 25, 2020 (3 days remaining)

Issues 3 New issue
Open: 0 Closed: 3

Time tracking
No estimate or time spent

Merge requests 6
Open: 0 Closed: 1 Merged: 5

Releases
None

Reference: courses/dit638/stu...

Issue Organization Overview

courses > ... > Student Repositories > group_01 > Issues > #24

Closed

Opened 2 days ago by stankoj

Reopen issue

New issue

Test Steering Calculator

@osma and @haiali

Edited 2 days ago by stankoj

Related merge requests 2

Unit testing !77

Software testing !80

0

0

Oldest first

Show comments only

Write

Preview

Write a comment or drag your files here...

To Do

Add a To Do

0 Assignees

Milestone Testing Methods for the functions developed

Time tracking

Due date

Labels testing

Confidentiality















Lock issue

3 participants

31

CI/CD Pipeline

courses > ... > Student Repositories > group_01 > Pipelines

All 417 Pending 0 Running 0 Finished 417 Branches Tags						Run Pipeline	Clear Runner Caches	CI Lint
Status	Pipeline	Triggerer	Commit	Stages				
passed	#13260 latest		master -> d2aa69da Merge branch 'fix-deci...		00:01:24 18 hours ago			
passed	#13259 detached		I196 -> a430e4e5 The algorithm now che...		00:01:24 18 hours ago			
passed	#13258		fix-decides... -> a430e4e5 The algorithm now che...		00:01:28 18 hours ago			
passed	#13257		master -> 6ba1225e Merge branch 'liveGrap...		00:01:26 20 hours ago			
passed	#13256 detached		I195 -> b213f337 gitlab-ci.yml-update		00:01:27 20 hours ago			
passed	#13255		liveGraph -> b213f337 gitlab-ci.yml-update		00:02:08 20 hours ago			
passed	#13251		master -> cfd0d6e1 Merge branch 'liveGrap...		00:01:25 21 hours ago			