Autonomous control of field robot

Individually study activity in field robotics

Authors:
Christian Liin Hansen

Instructors:
Kjeld Jensen

University:
The Technical Faculty, University of Southern Denmark

Course:
Extension of X-ROBO-U1 by individually study activity. 5 ECTS point

Project period:
1st of September 2014 - 23th of January 2015

Abstract

This rapport documents the individual study activity, which was carried out during the 3rd semester of the Master degree in Robotic System Engineering study at the University of Southern Denmark.

The goal of the individual study project is continue the work after the summer course, which was provided by the University of Southern Denmark (SDU) in 2014. The work contains implementing a FroboMind architecture, based on ROS, that makes the field robot, which is shown in the front page, able to follow a line of green tape inside RoboLab autonomously.
The robot is driven by a vision camera through line-following. At the end of a line, the robot is turning by using an IMU and follow the line back again. All the functionality is implemented in C++ code, structured by the architecture of FroboMind.

The final demonstration of the autonomously robot inside RoboLab, can be seen in the follow video on YouTube:
`https://www.youtube.com/watch?v=rYBc2Y-erCM`

This documentation can be found in the authors github repository:
`https://github.com/ChristianLiinHansen/ISA-doc`

The source code can be found on the authors github repository:
`https://github.com/ChristianLiinHansen/ISA`

Contents

1 Introduction

The University of Southern Denmark (SDU) was providing a summer course in Field Robotic in summer of 2014. This course was inspired from the annual Field Robotic Event (FRE) [1] and hence a field with growing maize was established outside the university. The goal was to design an autonomous field robot capable of making its way in the field of the university from a START position to a FINISH position. The path was similar to the field sketched in figure 1.1. The outcome of the summer course was a mechanical robot, which is shown in on the front page of this rapport. The field robot contained motors, H-bridges, FroboMind controller board and camera. The camera was mounted in the front, adjusted to make visual contact to the rows of maize in the field. A row extraction algorithm outputting the parameters distance r and angle θ was implemented by the author during the summer course. The result of this is shown in figure 1.2 and figure 1.3. Doing test inside RoboLab the algorithm was performing better, since the environment was ideal compared to the field. The result in figure 1.2 was only produced by controlling the robot using a keyboard. The result in figure 1.3 was only produced by walking with a handheld webcam in the field outside. At the moment of recording, the angle was defined between -90 and 90. In this project, the angle has been redefined to be between 0 and 180 degree.

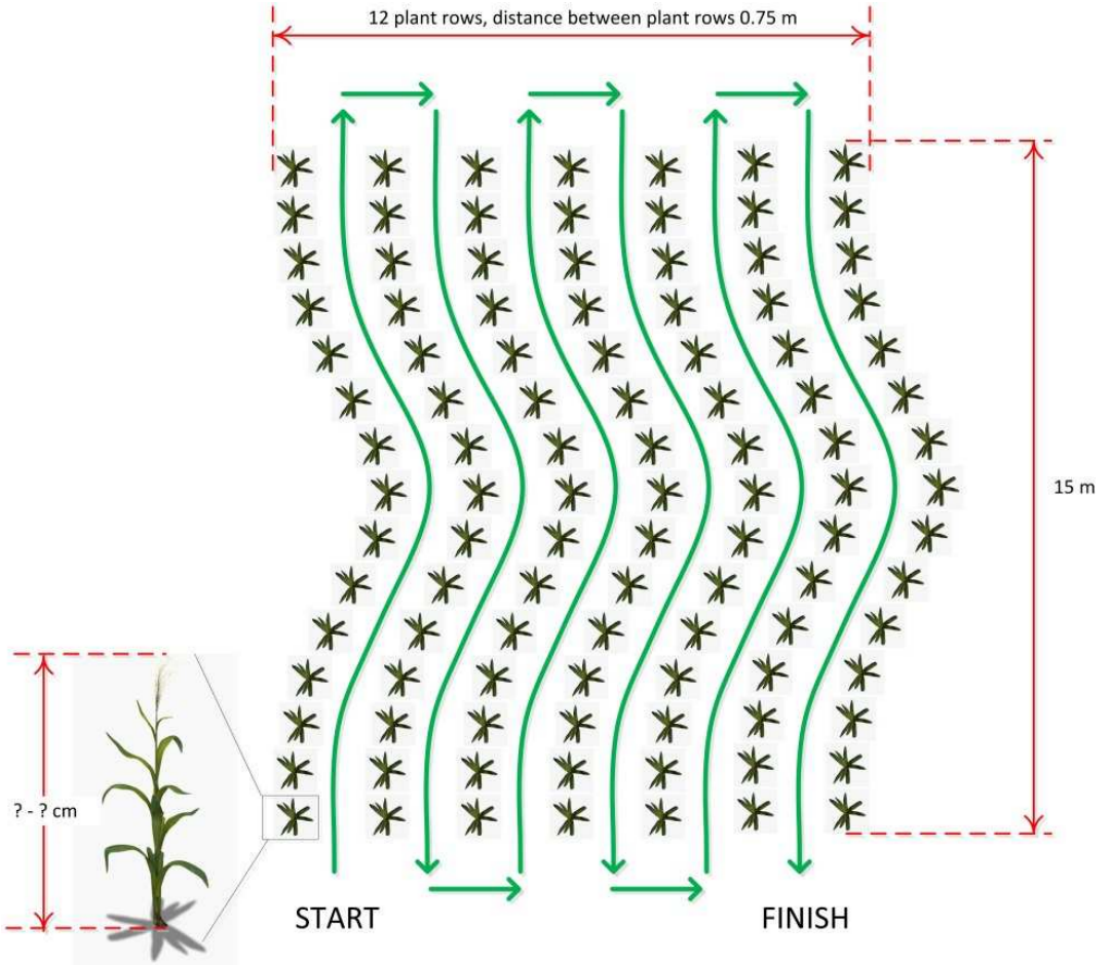

Figure 1.1: A sketch of the test field at FRE 2014

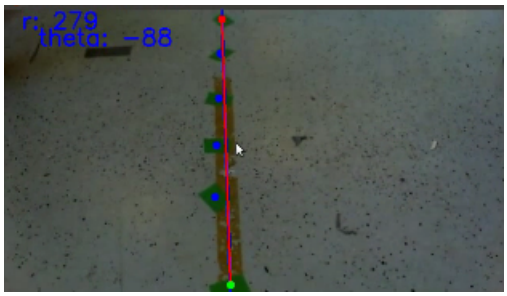

Figure 1.2: Testing algorithm inside RoboLab

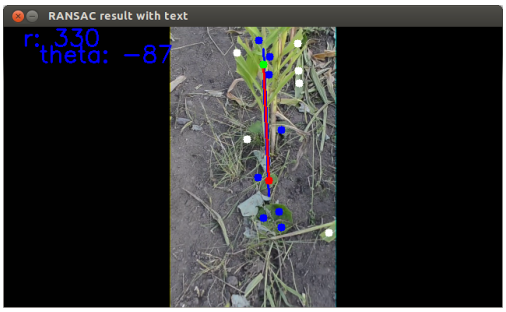

Figure 1.3: Testing algorithm outside in field

1.1 Problem description

At the end of the summer course, the mechanically robot was constructed and software for the camera was implemented to do row extraction, but nothing more was completed regarding implementation of the field robot in ROS, doing the summer course. That means that the field robot never came to the point, where it could drive autonomously through the field or even inside in RoboLab. This is the fundamental problem statement, which justify the reason for the authors individual study activity. The goal of this project is to continue the work after the summer course and be able to implement an system in order to experience a fully autonomously field robot.
The problem in this project is to build up a FroboMind architecture, bases on ROS, that make the field robot able to drive autonomously by using at least one webcam for navigation and IMU for turning. To start with the indoor environment, the robot must be able to follow the line of green tape on the floor, by having the two driving wheels on each side of the line. At the end of a line, the robot must be able to turn and follow a line back. Further extension is to have the field robot be able to follow the line of maize out in the field. This would include more optimizing work on the image processing node, due to the fact that the environment is much more complex.

1.2 Aim of project

The aim of the project aim is to implement a FroboMind [2] architecture, which is shown in figure 1.4. This figure is described below.
The outer blue *Field robot* frame, in figure 1.4 indicate that the physical robot might need optimization of the camera mounting location. The *Image processing* node needs to handle *End of line* detection and interface the parameters r and θ to the rest of the system through ROS. Optimization in general if the row detection should work outside in the field. An IMU will be connected to the system, and hence a *Gyro processing* node is needed to track the orientation. A *Decision making* node is needed to handle the inputs from the vision system and data from the IMU. This is referred to the *Mission Planning* component in figure 1.4. Inside the decision node, a state machine is implemented in order to let the robot have different *behaviours*. Regarding which behaviour the robot is in, different *executing* plans will be transferred to a FroboMind layer, which handles motor control. The *Odometry processing* component has not been implemented in this project.

1.2.1 Milestones in project

The project is decomposed into different milestones (MS). The aim is to complete all the MS. To follow a structured approach towards the aim, the MS has been prioritized. In this project the focus is to complete the chain in the project, i.e. have a field robot running

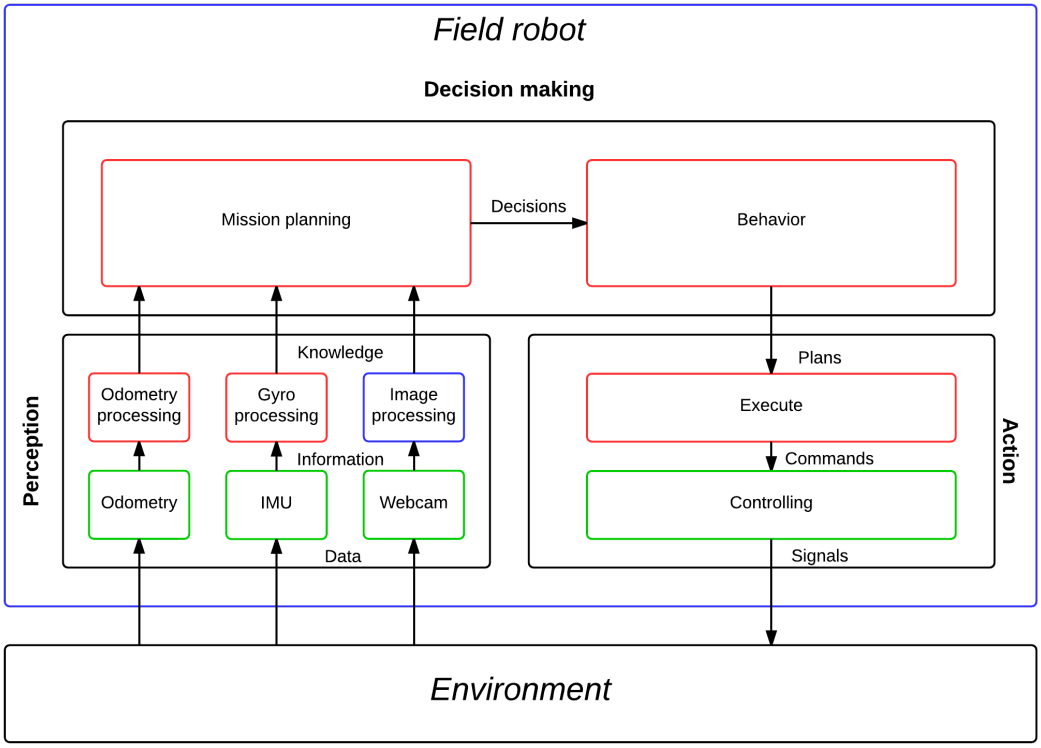

Figure 1.4: The FroboMind architecture of the field robot. Each component has been color coded. Green color indicates already implemented components, blue indicates components that needs optimization and red indicates components that is not implemented.

autonomously inside RoboLab, where all components is working in the ideal environment. Taking the project further to handle outdoor environment is an optimization process, where each component can be optimized.
The milestones is defined as followed:

- MS 1: The robot is able to sense the green tape inside RoboLab by only using the webcam. The FroboMind architecture in figure 1.4 is implemented and hence the robot can follow a line of dotted green tape autonomously.
- MS 2: The robot is able to detect *End of line* zones inside RoboLab in the vision node, which can set the robot in different behaviours, i.g turning or line-following.
- MS 3: The gyro data from the IMU can be used to perform an approximately 180 degree turning around the center of mass of the robot inside RoboLab and following the same line again.
- MS 4: With a setup of two parallel line of green tape inside Robolab, the robot is able to turn around one of the driving wheel pivot point, when entering the end of line and hence follow the new line.
- MS 5: A implemented Kalman filter will provide the sensor fusion between data from vision node and gyro node and hence further stabilizing the line-following.
- MS 6: The vision algorithm is optimized to handle the outdoor complex environment regarding to *End of line* detection. Therefore the robot is able to sense and follow a line of maize outside in the field by using the webcam, gyro and sensor fusion with a Kalman filter.

- MS 7: The robot can successfully follow a line of maize in the field, turn at the end and follow the new line of maize.
- MS 8: The robot can successfully follow a line of maize in the field, perform an approximately 180 degree turning around the center of mass of the robot and following the same line of maize.
- MS 9: The robot can successfully follow a line of maize in the field,turn around one of the driving wheel pivot point, when entering the end of line and hence follow the new line of maize.
- MS 10: The robot can successfully complete multiple rows with successful line following and turning at end outside in the field.

1.3 Deliverables

The project will be reviewed by the supervisor about halfway the trough the project period which is November 1st, 2014. By this date the following will be delivered:

- The sections “Introduction” and “Materials and methods” of the project report.

The project deadline is January 1st, 2015. By this date the following will be delivered:

- A project report documenting the fulfilment of:
  - Project aims
  - Deliverables
  - Learning goals
  - Accept test results
- FroboMind software modules and documentation
  - Mission control node implementation
  - Image processing node optimization

1.4 Learning goals

At the end of this project, the student will be able to:

- Identify a specific problem domain through literature study.
- Apply engineering skills in the evaluation of existing commercial and research solutions related to the identified problem.
- Propose an optimal solution based on domain knowledge and literature review.
- Apply engineering approach to construct a prototype demonstrator based on the proposed solutions.

Furthermore the student has through the prototype demonstrated engineering skills within:

- Optimizing row detection algorithms for a web camera.
- Using FroboMind architecture to interface different sensors.
- Using a Kalman filter for sensor fusion between IMU, encoders and webcam.
- Using a PID regulator for regulate velocity commands.
- Project management.

2 Related work

From previous work [3] in the summer course 2014, ideas for implement the mission control was discussed. The image processing node output the distance r and the angle θ. Additionally an *end of line* flag parameter is added doing this project. The image processing node has the following interface:

- Input: 640x480 RGB image from webcam at 30 fps
- Output: The distance r, which is the perpendicular distance from the line extracted to the upper left corner of the images. r is between 0 and 639 pixels
- Output: The angle θ, which is the angle between the horizontal x-axis and the line extraction. θ is between 0 and 180 degrees.
- Output: The boolean flag *end of line* is true when the robot has approached an end of a line of maize. Otherwise false. angle between the horizontal x-axis and the line extraction.

The image processing algorithm contains three components. The first component is extracting the green nuance for each image and together with a threshold, contours finding and raw moments calculation, the central mass coordinates for each green segment in each images is found. The second component is applying the RANSAC [4] algorithm on the dataset that contains the (x,y) central mass coordinates. The RANSAC stands for Random Sample Consensus, which means a line is spanned by two random points and the number of inliers is counted for that specific line. An inlier is defined as a point, where the perpendicular distance from the point to the line spanned by the two random is below a given distance threshold. If a given points exceed that given threshold it is an outlier. The RANSAC algorithm repeats its procedure 1000 times for each image and the line that contains the most inliers is defined to be the best fitted line in the current image. A flowchart of the RANSAC algorithm is shown in figure 8.1 in the appendix, section 8.2. A video demonstrating the RANSAC algorithm, can be seen on the following YouTube link:
`https://www.youtube.com/watch?v=VPRr0OpJmf4`

The third and last component in the image processing algorithm is to calculate the perpendicular distance r from the best fitted line and to the origo, which is defined to be the upper left corner of the image. Additional the angle of the best fitted line in respect to the horizontal x-axis in the images is calculated. The image processing node with the RANSAC algorithm was tested both inside RoboLab and outside in the field. The performance is illustrated in figure 1.2 and figure 1.3, where the distance r and angle theta is displayed.

3 Materials And Methods

In this section the different component in the project will be presented. For better overview and more efficient process, the project has been divided into several components in a *Work-Breakdown-Structure* diagram, shown in figure 3.1. To reach the goal of the final project, the accept test must be approved. In order to pass the accept test, the line-follower and turning component must be tested together and approved and so forth.

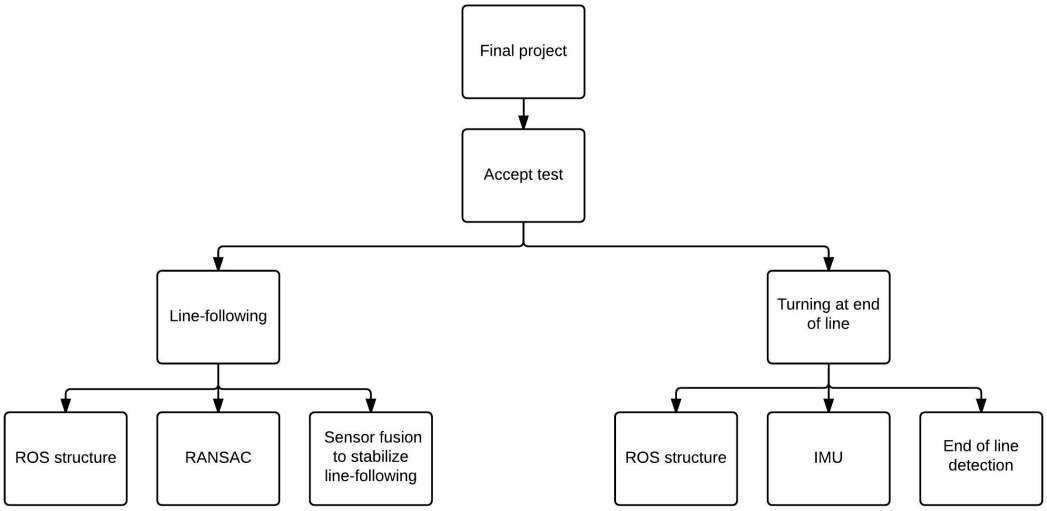

Figure 3.1: *Work-Breakdown-Structure* illustration shows how the project has been devided into several components

3.1 Line-following

The robot will be driving with a green dotted tape between the wheels inside RoboLab. The green line in figure 3.2 and figure 3.3 indicate the green tape on the floor. The green tape simulate rows of maize. The standard measurements between the rows in the field is 750 mm [**?**].The dimension of the field robot is 830 mm x 815 mm. The inner distance between the wheels is 600 mm. The robots translation and orientation offset is illustrated in these figures.

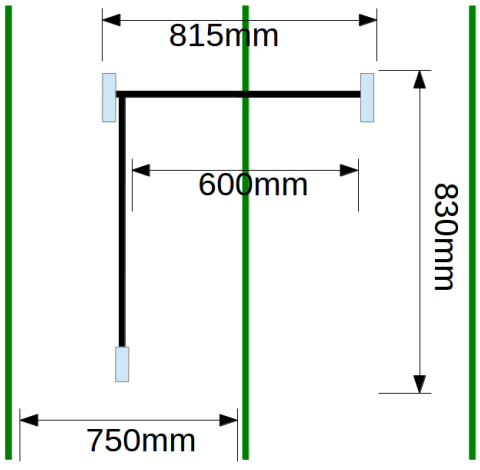

Figure 3.2: Translation offset equal to 0

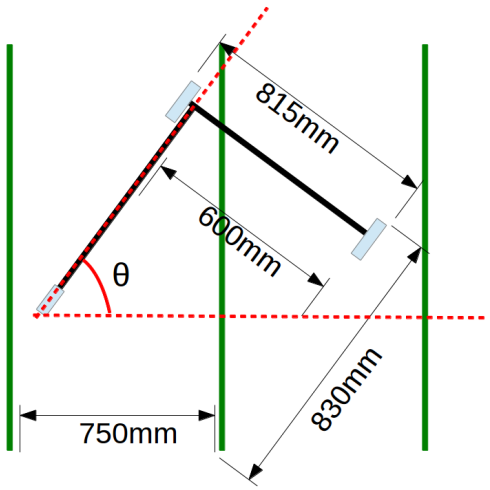

Figure 3.3: Orientation offset equal to θ

Doing test, the translation and orientation offset will change. The robot will navigate through the line by having the driving wheels between the lines.

3.1.1 ROS nodes for line-following

In order to insure line following, a communication between the vision node and the decision node was implemented. This is illustrated in figure 3.4 the rqt_graph command in ROS. The vision node was extended to send the boolean *end of line* flag within a custom messages trough the vision topic. In this way, the robot would stop, when there is less than two contours in the cameras field of view.

Figure 3.4: ROS structure with vision node that publish parameters on the */vision* topic. The decision node subscribes on the same topic and publish a linear and angular velocity on the */fmCommand/cmd_vel* topic. A dummy node listener was created to illustrate the last topic, which is not illustrated

Inside the decision node, a callback function reads the parameters on the /vision top and call the GetTwist function with r and θ, if the robot is in line-following state. The function GetTwist decides how the angular and linear velocity commands should be, depending on the parameters from the vision topic. The decision node is publishing on the topic /fmCommand/cmd_vel which is a component from FroboMind. To drive the robot forward, the following commands is e.g:

twistStamped.twist.linear.x = 0.2
twistStamped.twist.angular.z = 0.0

In order to turn the robot around is own axis the linear velocity is zero and the angular velocity is increased. If the angular velocity is positive, the robot start turning left and if the velocity is negative the robot turns to the right.
A pseudo code description is shown in algorithm 1 in appendix section 8.1.

3.2 Turning at end of line

In order to let the robot turn at the end of a line, the IMU vectornav 100 was used in this project. The reason for this, is that the IMU is a standard plug-and-play component available at the SDU library. A tutorial [5] was followed in order to receive data from the IMU. Using ROS node called *vectornav_vn100_node* available through FroboMind and hence it was easy to receive data on the topic /fmInformation/imu. This topic contains the orientation of the IMU in quaternion representation. In order to convert quaternion to Euler representation, a gyro node was implemented. The task of this node is to subscribe on the /fmInformation/imu topic and publish the orientation relative to the z-axis of the IMU. This was achieved by using the transform class *tf* in ROS [6].

3.2.1 ROS nodes for turning

Together with the vision node, the gyro node is communicating to the decision node through the topic /orientation. Using *rqt_graph* from the rqt package in ROS, the nodes and topic is illustrated in figure 3.5. Before the angle is published on the topic, the angle is converted from radian to degree. When the vision node publish the boolean

end of line flag to true, the robot goes into a turning state and reads the data from the /orientation topic. The orientation is relative, and the robot turns until a difference on 175 degrees is achieved. The data on the topic /orientation has been converted to be between 0-360 degrees.

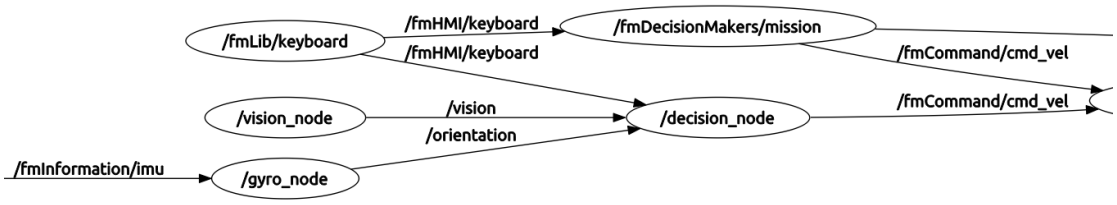

Figure 3.5: ROS structure with gyro node that publish the angle on the */orientation* topic. The decision node subscribes on the same topic and publish a linear and angular velocity on the *cmd_vel* topic.

3.3 Decision node

The software is implemented to interface the frobit_v2 component from FroboMind [7]. The frobit component contains extra functionality like way point navigation and differential_odometry. This is out of the scope of this project. Running the *rqt_graph* shows in figure 3.6 how the connection is between the vision, gyro and dececsion node and towards the Frobit node. Additional connection from the vision, gyro and decision node is shown, but should be ignored since ROS automatically subscribe on topic from a ros_back node.

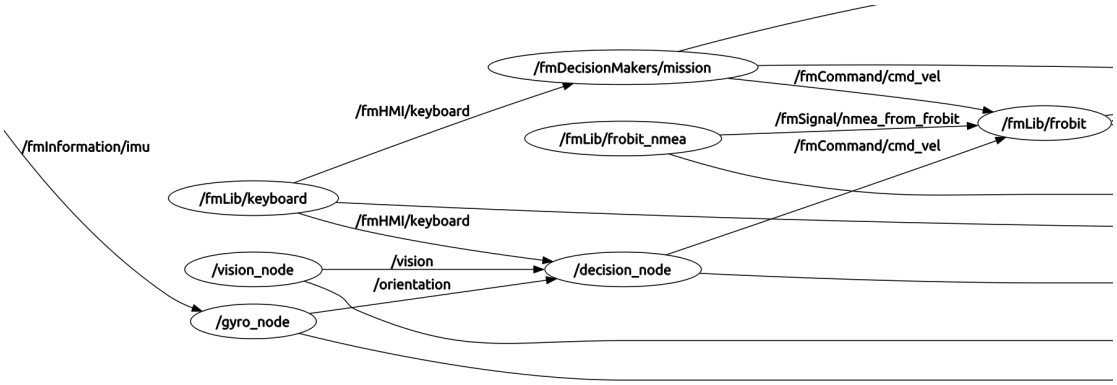

Figure 3.6: ROS structure with the frobit_v2 component running together with the vision, gyro and decision nodes

At start up, a check for user-input is implemented in order to let the user activate the robot. This is the reason why the decision node subscribes on the /fmHMI/keyboard topic, which is shown in figure 3.6. This is implemented to allow enough time to let the system settle. Start up happens when the user is calling the roslaunch file isa_run.launch from the terminal window on a Ubuntu 12.04 machine. In the decision node, a state machine was implemented, in order to change the behaviours of the robot. These are follow line, stop, turning and start. After the user hits the keyboard, the system is initialized to be in line following state. While the system is receiving *false* from the vision topic, the robot stays in the follow-line state. As soon as the line end, the system will send *true* and the behaviour change to end-of-line state. After a small wait, the system store the start angle value from the IMU and goes to a turning state. Within the turning state, the robot turns itself clockwise around its center of mass until the difference between the current angle and start angle is greater than or equal to 175

degree. When this occur, the system jumps to the start-of-line state, where it wait and goes to the follow-line state. The state machine is illustrated in figure 3.7.

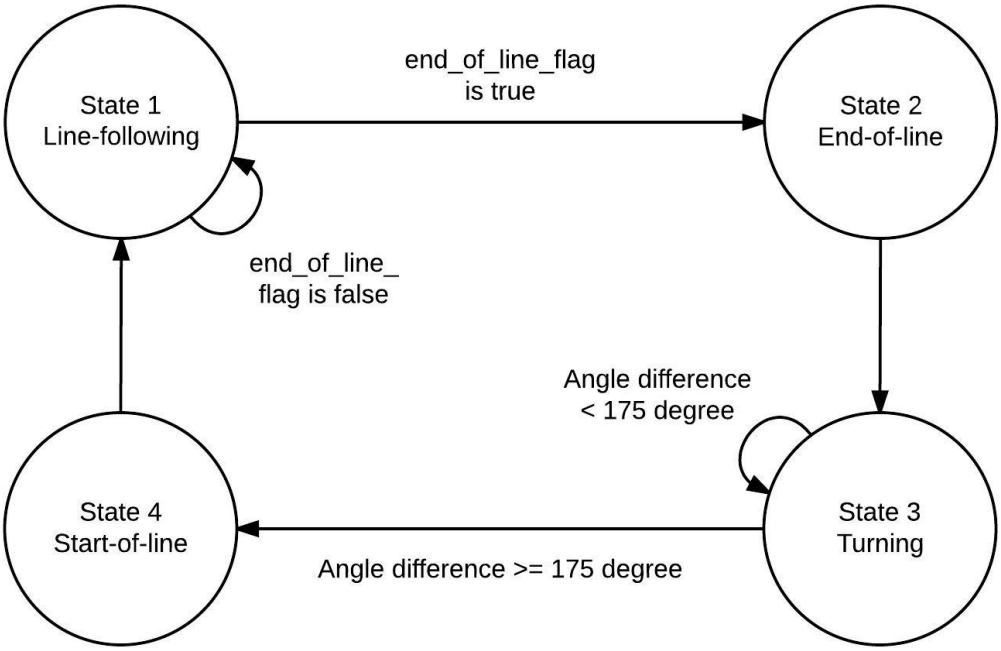

Figure 3.7: State machine in the decision node.

3.4 Simulation line-follower

To simulate the line-follower, the video input stream was based on a still images, which could be rotated an translated in any matter. This is showed in figure 3.8, 3.9, 3.10 and 3.10. The simulation component is initialized by calling the *frobit_sim_wpt_nav* component in Frobit v2 demo from FroboMind [7].

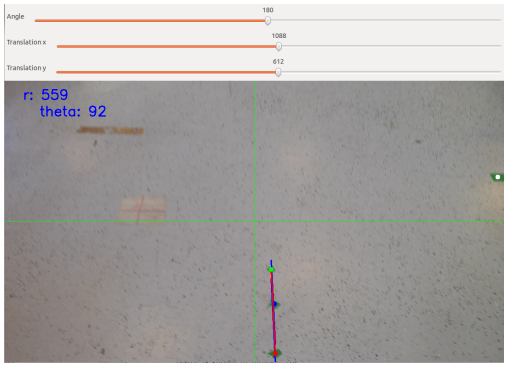

Figure 3.8: 92 degrees

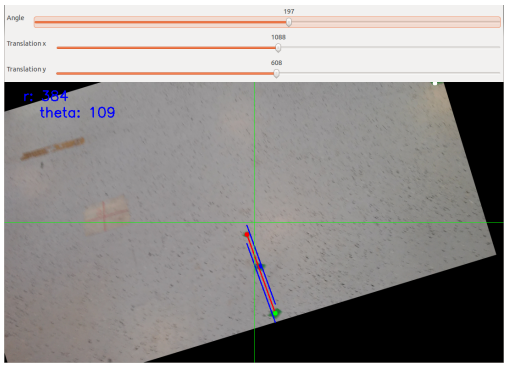

Figure 3.9: 109 degrees

The result of the simulation shows that is is possible to steer the robot. The simulation result is shown in figure 3.12. A demonstrating of the simulation is available on the following YouTube link:
`http://youtu.be/sOadsYU66x4`

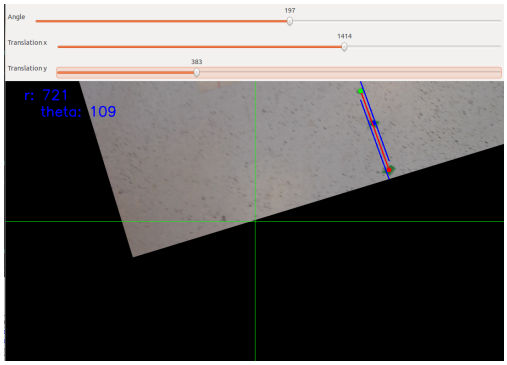

Figure 3.10: 91 degrees

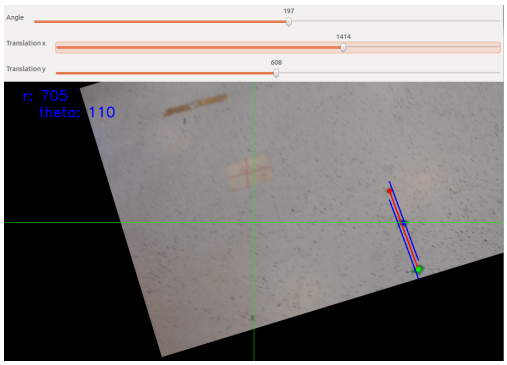

Figure 3.11: 109 degrees

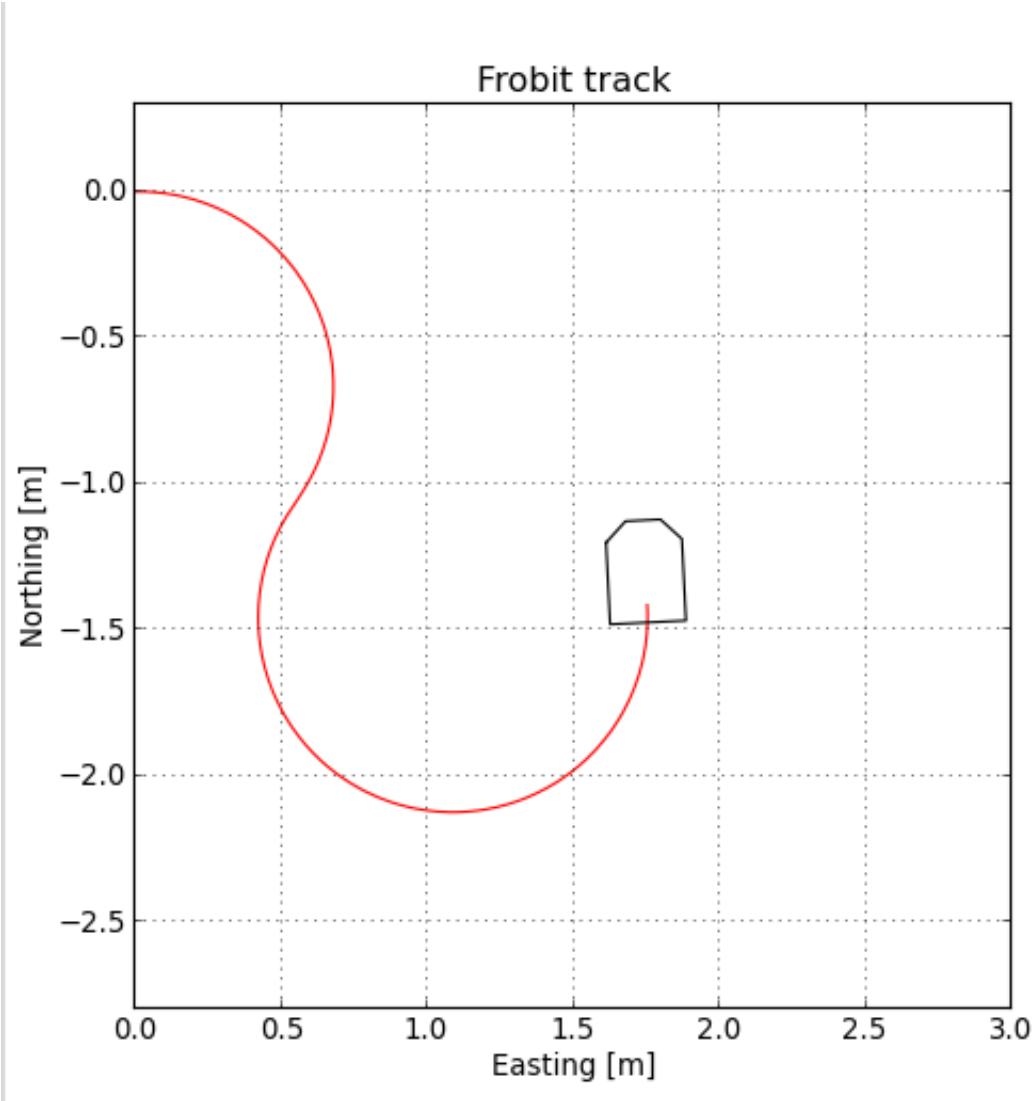

Figure 3.12: Simulating video stream with rotating still images result in change of angular velocity of robot in simulation. The linear velocity is constant.

3.5 Testing line-follower and end-of-line inside RoboLab

In order to test line-follower a line of green dotted tape was arranged inside RoboLab. To challenge the line-following additional line of green tape was placed on the floor with approximately 45 degree. The setup is showed in figure 3.13, 3.14, 3.15 and 3.16. A video is available on the following link on YouTube: `http://youtu.be/RwXT7htTvRQ`

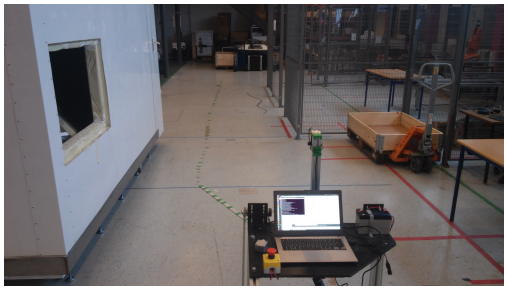

Figure 3.13: Start of line-following test seen from back

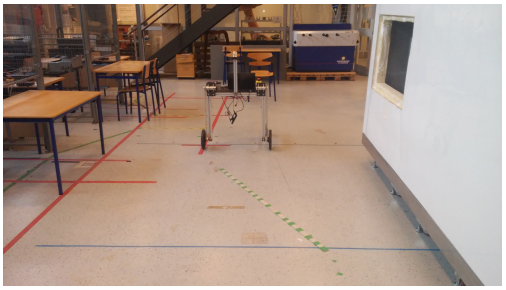

Figure 3.14: Start of line-following test seen from front

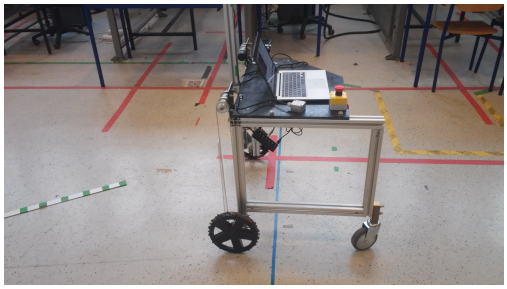

Figure 3.15: Start of line-following test seen from left

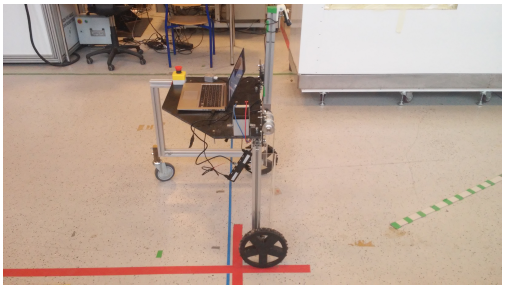

Figure 3.16: Start of line-following test seen from right

3.6 Testing IMU turning

A video of the robot turning, using the data from the IMU is available on the following YouTube link:
`https://www.youtube.com/watch?v=2ydvrYY0IEw`

3.7 Accept test

In order to complete the project, the accept test must be approved. The accept test contains the merging between line-following and turning at end of line. A demonstration of this is available in the following YouTube link:
`https://www.youtube.com/watch?v=rYBc2Y-erCM`

This video demonstration shows that the field robot is able to follow the line, turning at the end and follow the line back again multiple times. Therefore the accept test is approved.

4 Discussion and future work

Looking at the milestones, which is described in section 1.2.1, the first three milestones has been completed. It was decided to rearrange the milestones in a way, so the focus was to have a field robot working autonomously inside in RoboLab, rather optimizing vision node in order to handle the outside complex environment. In that sense this closed the loop in the project, i.e. all the vital component was implemented. The next step is to examine a given component and try to optimize it, in order to text the field robot to the next level. The idea is illustrated in figure 4.1, where the vital components is indicated in green color. The arrows out to the thinking bubbles indicate ideas to optimize each component.

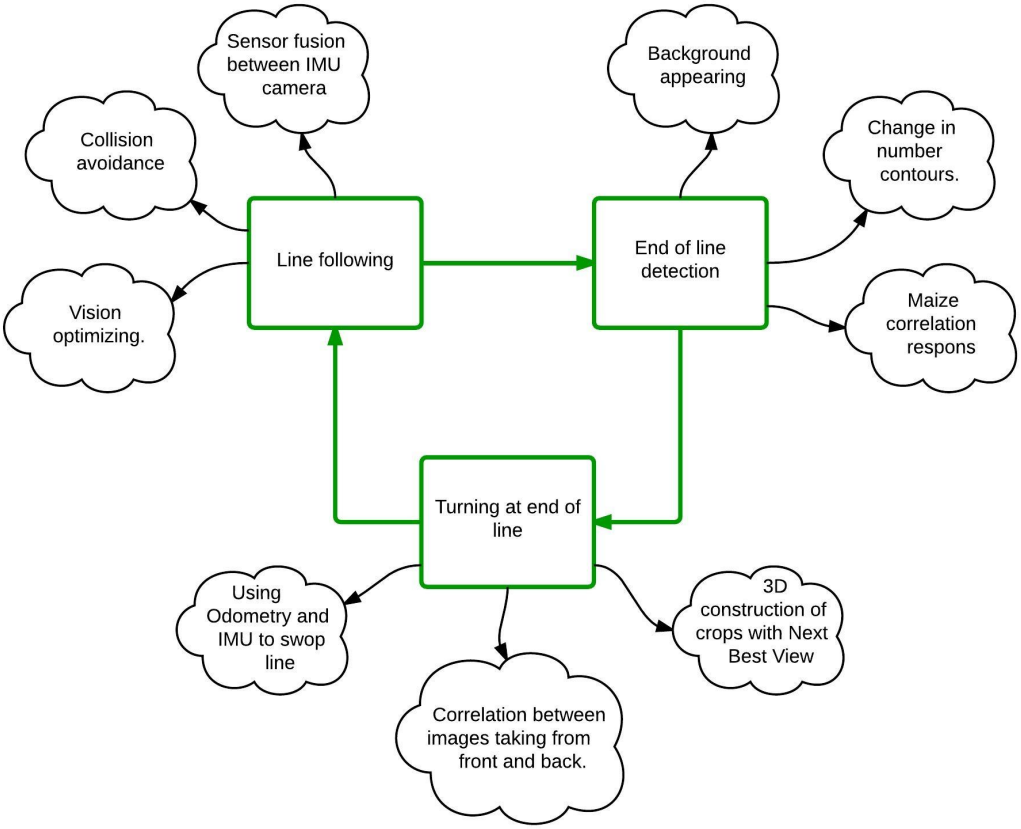

Figure 4.1: Closing the loop of the project. The loop is indicated by the green arrows that loops trough the modules, that was implemented

In order to measure how precise the robot follows the line inside RoboLob, the future work is to add a marker locater on top of the robot and use the system from the Robotic System Design coarse on SDU to monitoring the position of the robot [8].

5 Conclusion

The field robot was constructed mechanically doing the Field Robotic summer course in 2014 at SDU. However the robot lacked a software architecture, based on FroboMind in order to see the mechanically robot move autonomously. Doing this individually study activity project, the robot come to the state where it is able to run autonomously inside RoboLab in ideal environment using a webcamera and IMU. It is believed that the robot is able to drive outside in the field autonomously, however optimization of different modules is needed.

Regarding the learning goals, in section 1.4, some points has been removed in this project. Optimizing row detection algorithm for a webcam is, as described earlier not the essence of an autonomous field robot. The Kalman filter and PID regulator is important but not vital to implement when it comes to drive in the ideal environment in RoboLab. This is proved in the final video presentation, where the link is available in the abstract of this rapport. However the Kalman filter would be vital when taking the robot outside in the field. The Kalman filter should estimate the variance of the RANSAC row detection, compared to the data from the IMU.

The project management during this project has been handled in Google Drive. Here the timetable, meeting agendas etc. has been shared with the supervisor. In order to management the project together with the other courses like Master Thesis and Robotic System Design a simple overall logbook was used. This document is available in the following link: `https://docs.google.com/document/d/17jj2YI9C5GpDb2XIGMZTKp0gSwX_TrTwTy3ib3WFzTQ/edit`

6 Acknowledgement

First I would like to thank my supervisor Kjeld Jensen to approve this individual study activity. Thanks for the response and help through the project.
I would also like to thank my fellow students: Leon Bonde Larsen, Frederik Hagelskjær, Kent Stark Olsen and Mathias Neerup. Spending the time in the FroboMind office at RoboLab at SDU with these people are always appreciated. I thank them for their their great support, discussion about possible approaches and enthusiasm regarding this project.

7 Bibliography

[1] Germany University of Hohenheim.
Field robot event 2014 - task description.
https://fre2014.uni-hohenheim.de/uploads/media/Booklet_content-v3-complete_01.pdf, 2014.
Available: 7-10-2014.

[2] Kjeld Jensen, Morten Larsen, Søren H. Nielsen, Leon B. Larsen, Kent S. Olsen, and Rasmus N. Jørgensen.
Towards an open software platform for field robots in precision agriculture.
Robotics, 3:207–234, 2014.
doi:10.3390/robotics3020207.

[3] Christian Liin Hansen.
Field robot summer course 2014.
https://www.dropbox.com/s/ew82lvrkpvb3pjh/FRSC14.pdf?dl=0, 2014.
Available: 7-10-2014.

[4] Martin A. Fischler and Robert C. Bolles.
Random sample consensus - a paradigm for model fitting with apphcatlons to image analysis and automated cartography.
SRI International, 6:381–395, 1981.
doi:10.1145/358669.358692.

[5] Kjeld Jensen.
Using the vectornavvn-100 imu.
http://frobomind.org/index.php/People:Kjeld_Jensen:Using_the_VectorNav_VN-100_IMU, 2014.
Available: 7-10-2014.

[6] ROS Answer.
Convert from quaternion to euler.
http://answers.ros.org/question/36977/invalid-arguments-convert-from-qua
2014.
Available: 7-10-2014.

[7] Kjeld Jensen.
Frobolab repository on github.
https://github.com/FroboLab/FroboMind/tree/master/fmApp/frobolab/frobit_v2_demo, 2014.
Available: 7-10-2014.

[8] Henrik Skov Midtiby.
Marker locater repository.
https://github.com/henrikmidtiby/MarkerLocator, 2014.
Available: 7-10-2014.

8 Appendix

8.1 Pseudo code for GetTwist function

Input: Distance r and angle θ
Output: Twist messages which is published on the /fmCommand/cmd_dev topic

---

```
if r is more visualized in the left side and is almost vertical then
    turn left slowly
else if r is more visualized in the right side and is almost vertical then
    turn right slowly
else
    if line is in the left side and tilted to the right then
        if angle is between 85 and 70 degree then
            turn right slowly
        else if angle is between 70 and 60 degree then
            turn right medium
        else
            turn right fast
            stop linear velocity
        end
    else if line is in the left side and tilted to the left then
        if angle is between 100 and 110 degree then
            turn left slowly
        else if angle is between 110 and 120 degree then
            turn left medium
        else
            turn left fast
            stop linear velocity
        end
    else if line is in the right side and tilted to the left then
        if angle is between 100 and 110 degree then
            turn right slowly
        else if angle is between 110 and 120 degree then
            turn right medium
        else
            turn right fast
            stop linear velocity
        end
    else if line is in the right side and tilted to the right then
        if angle is between 100 and 110 degree then
            turn left slowly
        else if angle is between 110 and 120 degree then
            turn left medium
        else
            turn left fast
            stop linear velocity
        end
    else
        Drive forward
        stop angular velocity
    end
end
```

Algorithm 1: Pseudo code of the implemented GetTwist function for row navigation

8.2 Flowchart for RANSAC algorithm

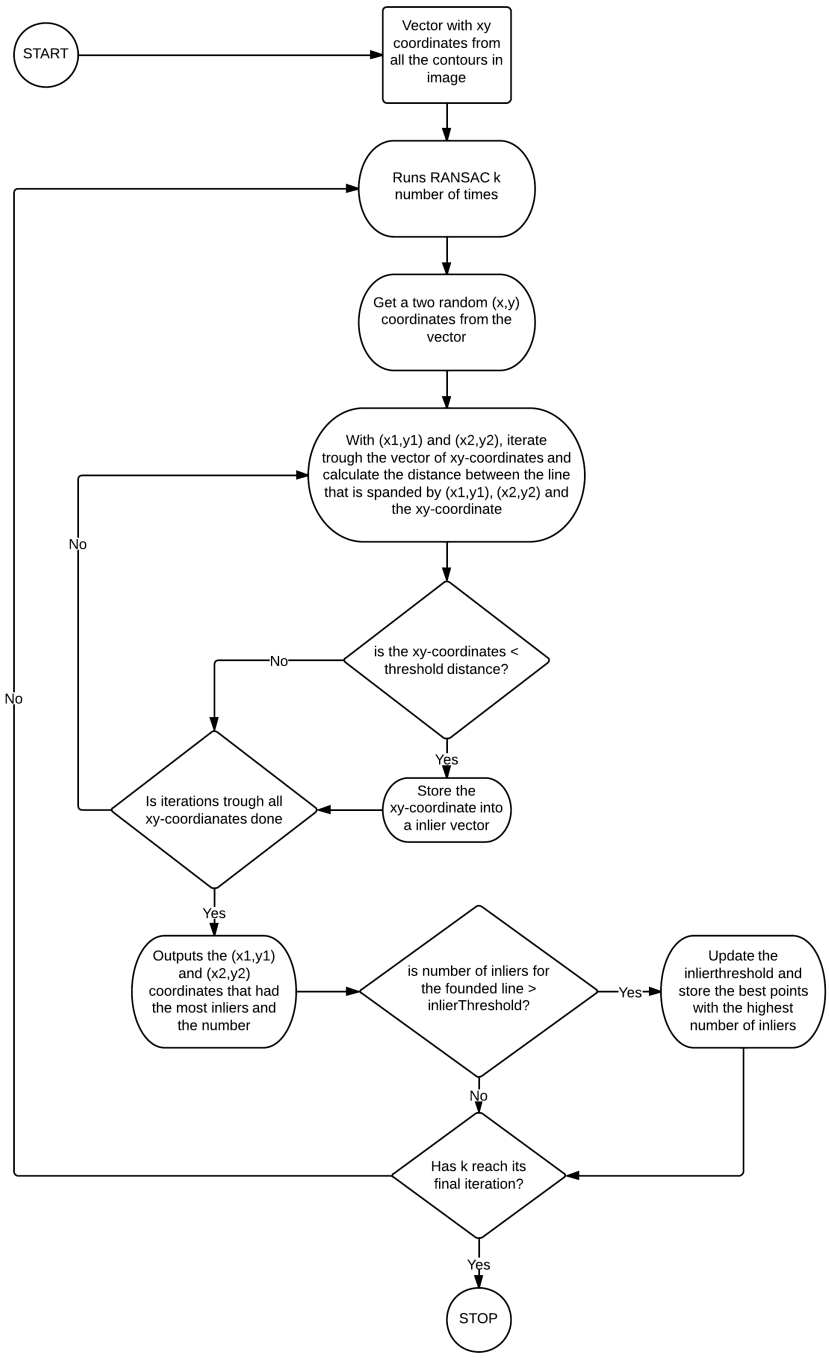

Figure 8.1: Flowchart of the implemented RANSAC algorithm