UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Development of a self-navigating AGV for use with multiple overhead cameras

Mechanical Project 478

Final Report

Sam Jeffery
20029004

Supervisor:
Dr K. Kruger
2019

Faculty of Engineering
Fakulteit Ingenieurswese

Department of Mechanical and Mechatronic Engineering
Departement Meganiese en Megatroniese Ingenieurswese

Privaat Sak / Private Bag X1, Matieland, 7602, Suid-Afrika / South Africa · Banghoek Weg / Road, Stellenbosch, 7600, Suid-Afrika / South Africa

Tel: +27 21 808 4376 · Epos / Email: mmchair@sun.ac.za

Plagiarism declaration

I have read and understand the Stellenbosch University Policy on Plagiarism and the definitions of plagiarism and self-plagiarism contained in the Policy [Plagiarism: The use of the ideas or material of others without acknowledgement, or the re-use of one's own previously evaluated or published material without acknowledgement or indication thereof (self-plagiarism or text-recycling)].

I also understand that direct translations are plagiarism, unless accompanied by an appropriate acknowledgement of the source. I also know that verbatim copy that has not been explicitly indicated as such, is plagiarism.

I know that plagiarism is a punishable offence and may be referred to the University's Central Disciplinary Committee (CDC) who has the authority to expel me for such an offence.

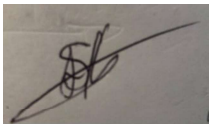
I know that plagiarism is harmful for the academic environment and that it has a negative impact on any profession.

Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully (acknowledged); further, all verbatim copies have been expressly indicated as such (e.g. through quotation marks) and the sources are cited fully.

I declare that, except where a source has been cited, the work contained in this assignment is my own work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

I declare that have not allowed, and will not allow, anyone to use my work (in paper, graphics, electronic, verbal or any other format) with the intention of passing it off as his/her own work.

I know that a mark of zero may be awarded to assignments with plagiarism and also that no opportunity be given to submit an improved assignment.

Signature: 

Name: Sam Jeffery Student no: 20029004

Date: 24 October 2019

Executive Summary

Title of Project
Development of a self-navigating AGV for use with multiple overhead cameras.
Objectives
The objectives of this project revolve around improving the inherited system. The objectives include: improving the system's robustness by implementing a position estimator to provide open loop feedback when camera feedback is inconsistent, expanding the system to be able to use multiple cameras for feedback to increase the size of the workspace, and improving the path following controller by implementing more advanced algorithms.
Which aspects of the project are new?
This project implements a self-navigating AGV using overhead cameras where all computing is done onboard the AGV. In addition, the project expands the working system to receive information from two overhead cameras and implements a position estimator to add an open loop feedback option.
What are the expected findings?
It is expected that an AGV can communicate with more than one overhead cameras in order to navigate a static workspace consistently.
What value do the results have?
The project's goal is to determine if onboard computing on an AGV with feedback from an overhead camera is a viable technique for use in a large static environment. If this design can be shown to work, this report will have shown that AGVs can be used as substitutes for conveyor belts in factories or as delivery systems in a warehouse.
What contributions have other students made?
Ms S. Brits and Mr S. Wilson worked on this system in 2016 and 2017 respectively.
What aspects of the project will carry on after completion?
Expansion of the system to work with more than two cameras, further development of the calibration system, improvement of the path planning algorithm, implementation of a Kalman filter and advancing the system to work in a dynamic environment.
What are the expected advantages of continuation?
Improving the system will allow for more successful and robust prototypes. If the product can be improved to function in a dynamic environment; it can be used in conjunction with conveyor belts to transport goods and raw materials.
What arrangements have been made to expedite continuation?
The design and processes implemented to get to this point have been clearly defined in this report so that future students can expand on the research. The AGV and cameras will be available to future students as well as the code used to run the system.

ECSA Outcomes

Outcome	Chapter
ELO 1. Problem solving: Demonstrate competence to identify, assess, formulate and solve convergent and divergent engineering problems creatively and innovatively.	1, 3, 4, 5, 6, 8
ELO 2. Application of scientific and engineering knowledge: Demonstrate competence to apply knowledge of mathematics, basic science and engineering sciences from first principles to solve engineering problems.	5, 6, 7, 8, 9, 10
ELO 3. Engineering design: Demonstrate competence to perform creative, procedural and non-procedural design and synthesis of components, systems, engineering works, products or processes.	3, 4, 5, 6, 8, Appendix A, Appendix E
ELO 5. Engineering methods, skills and tools, including information technology: Demonstrate competence to use appropriate engineering methods, skills and tools, including those based on information technology.	5, 6, 8, Developed Code
ELO 6. Professional and technical communication: Demonstrate competence to communicate effectively, both orally and in writing, with engineering audiences and the community at large.	Project Proposal; Progress Report; Progress Presentation; First Draft; Final Report; Oral Presentation; Project Poster
ELO 8. Individual, team and multidisciplinary working: Demonstrate competence to work effectively as an individual, in teams and in multi-disciplinary environments.	All aspects of the project
ELO 9. Independent learning ability: Demonstrate competence to engage in independent learning through well-developed learning skills.	5, 6, 8, References, Developed Code

Table of contents

List of figures

Page

List of tables

Page

1 Introduction

1.1 Background

In any given factory or warehouse there is a need for materials and products to be transported from one location to another. Conveyor belts are usually used to transport goods in these situations, and while they can be useful and efficient, they are inflexible and can only be used in a single configuration at a time. Automated Guided Vehicles (AGVs) offer a modern transport solution. An AGV is a self-navigating vehicle that can perform specific tasks efficiently and autonomously. The AGV system usually consists of a wheel-based transport system that is controlled via a computer. AGVs do not need regular monitoring and can increase reliability and productivity, provided they have charged batteries. Unlike a person, an AGV does not lose concentration and will always comply with standards and requirements. Furthermore, AGVs can fit into a larger system of multiple devices, that can communicate constantly enabling it to run as efficiently as possible. Figure 1.1 shows an AGV used in Amazon warehouses.

Figure 1.1 Kiva AGV used in Amazon warehouses (Heater, 2019)

AGVs use a variety of methods to navigate depending on the situation. Most commonly, where they operate within known grounds, localisation can be done using magnetic tape, laser positioning, onboard cameras or external cameras. Other methods of navigation, such as simultaneous localisation and mapping (SLAM) navigation, are being investigated (AGV Technology, 2019).

AGVs are already being used extensively in warehousing and distribution as well as in industries, such as the automotive, pharmaceutical, chemical processing, and food and beverage industries (MHI, 2012). In these applications they are used to transport raw materials, assist with assembly, transport pallets, and perform container loading and unloading (MHI, 2012).

AGVs are relatively widespread in their use. Many companies have already designed and sold working systems that can adapt and deal with a dynamic

environment. Companies like IKV Robot have catalogues that include different AGVs for different applications.

This project will use an overhead camera as a feedback system to the AGV and individual onboard computing for navigation. Using feedback from the overhead camera the AGV should be able to plan a path to avoid obstacles and successfully reach the destination, as shown in Figure 1.2.

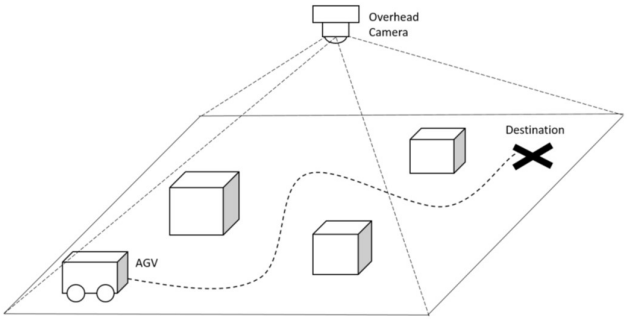


Figure 1.2 Self-navigating AGV with overhead feedback setup (Brits, 2016)

1.2 Objectives

This project is the third iteration of a project that was started in 2016 and continued in 2017. It builds on the previous projects in order to make the inherited design more reliable, and to upgrade specific components and flows in order to improve the inherited system. The project scope will be limited to a static environment, with a map of the workspace provided to the AGV. The inherited system will be reviewed in chapter 3.

The first objective of the project is to design and integrate a position estimator into the current position feedback and control system. The inherited system only updates the control system each time a new image is received by the AGV. By integrating a position estimator into the system, the AGV can estimate its position based on past locations and the inputs supplied to the stepper motors. This update reduces the system's reliance on consistent transfer from the overhead camera and thus increases the system's robustness.

The second objective is to expand the system to use more than one camera so that the size of the workspace can be increased. The project aims to design a system that works with two cameras and to provide a framework for expansion to a larger system with more than two cameras.

The third objective is to update the path-following algorithm that was implemented in the inherited system and was highlighted as a possible avenue for system improvement in past reports. Improved algorithms exist and are investigated in the literature review in chapter 2.

1.3 Motivation

This project aims to continue developing a self-navigating AGV that can be adapted for use in factories and industrial areas to, among other things, handle and transport material. Self-navigating AGVs have various advantages in the workspace and are being introduced in production, warehousing and distribution. AGVs require an initial investment and have limited costs. These costs are relatively cheaper compared with labour, which requires ongoing investment and has ongoing costs. Furthermore, AGVs follow safety and production guidelines without fail, allowing for increased productivity, accuracy and safety as well as consistent results – unlike human operators who can tire or lose concentration. The systems are also easily expandable, especially when the AGVs perform all computing onboard. In this situation, additional AGVs can be inserted into a system with ease. (Benevides, 2018)

The motivation behind this project is to show that an AGV can function successfully using an overhead camera and onboard computing. By using an overhead camera this project will take advantage of OpenCV libraries that contain advanced image processing software in order to locate the AGV. Overhead camera systems are easy to implement and are easily expandable compared with other options such as magnetic guidance, which requires pre-laid tracks. Additionally, onboard computing enables each vehicle to run autonomously, allowing for vehicles to be pulled in and out of the system without disruption. Therefore, the system can be expanded or contracted as required, increasing the system's flexibility.

Past projects have determined that processing is possible using onboard computing; however, the inherited system was not robust to communication issues causing the AGV to deviate from the planned path at times. This project aims to fix problems in the inherited design and implement new features in order to ensure that a reliable system can be delivered.

The objective to implement a position estimator in connection with the overhead camera feedback system aims to decrease the system's reliance on constant feedback from the camera system. The goal of connecting these systems would be to eliminate errors related to path following and achieve high accuracy with regards to reaching the destination. Future systems could make use of multiple AGVs, which would place a larger burden on the overhead camera system. A reliable position estimator can enable an AGV to reduce the number of requests it makes to overhead cameras, thereby lowering the burden on the camera system.

The objective to implement multiple cameras is a practical consideration. Successful use of multiple cameras would confirm that the AGV system can be expanded if required. As the size of a workspace covered by a single camera is relatively small, being able to communicate with multiple cameras is necessary for this design to be useful in a working environment.

2 Literature Review

This section will cover topics relevant to the system of interest. These topics have been researched to understand the inherited system and develop and implement methods to improve it.

2.1 Position Estimation

In order to reduce reliance on camera inputs, an AGV should be able to predict its position based on past camera feedback, as well as present inputs to the stepper motors. Path calculations for the motion of a vehicle have been studied extensively in past literature and this information can be drawn upon to create a specific estimator for this situation. The information in this section is adapted from Lucas (2006).

In order to model the movement of a vehicle, a circular path is used. While other path types can be used, this motion is the simplest and allows for smooth motion with no stopping and turning.

Firstly, as shown in Figure 2.1, basic trigonometry can be used to relate a triangle to a circle through radius (r) and position (x,y), using Equation 2.1 and Equation 2.2 below:

$$x = r \cdot \cos \emptyset \qquad \text{Equation 2.1}$$

$$y = r \cdot \sin \emptyset \qquad \text{Equation 2.2}$$

Where $\emptyset$ varies from 0 to 2π radians. If the angle is zero, the endpoint lies on the x-axis and if the angle is $\pi/2$, the endpoint lies on the y-axis. This relationship allows one to transform polar coordinates into cartesian coordinates. It also allows one to define an equation for a circle as a function of an angle.

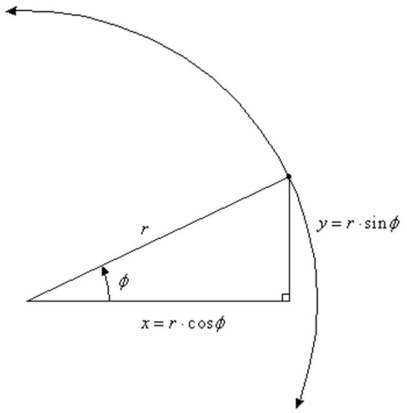


Figure 2.1 Cartesian coordinates on circle

Assuming all motion must be based on a circle (even if that circle has an infinite radius), one must develop a path based on a circular arc. The assumption is made that the robot's motion is such that it is always tangent to the centre of the turning circle at all points in its trajectory. Then, given a constant radius r_C for the turning circle and r_G for the measured length to the target, an isosceles triangle can be constructed, as shown in Figure 2.2.

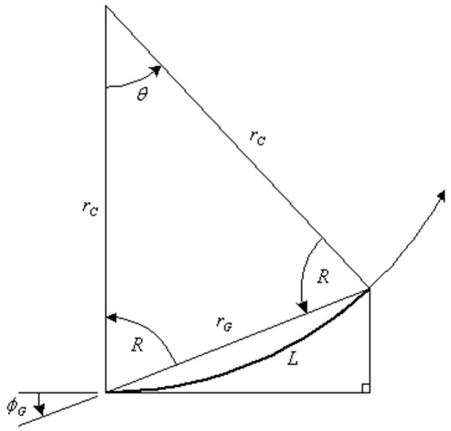


Figure 2.2: Trajectory of an AGV based on the arc of a circle

Using the law of sines, the different values in the triangle can be related in Equation 2.3.

$$\frac{\sin\theta}{r_G} = \frac{\sin R}{r_C} \qquad \text{Equation 2.3}$$

Further inspection of Figure 2.2 shows that angles R and $\emptyset_G$ form a right angle. Therefore:

$$\emptyset_G + R = \pi/2 \qquad \text{Equation 2.4}$$

$$\theta + 2R = \pi \qquad \text{Equation 2.5}$$

And

$$\theta + 2(\pi/2 - \emptyset_G) = \pi \qquad \text{Equation 2.6}$$

Which equates to:

$$\theta = 2\emptyset_G \qquad \text{Equation 2.7}$$

Importantly all these equations allow one to relate L, r_G, r_C and θ, as in Equation 2.8 and Equation 2.9:

$$r_C = \frac{r_G}{2\sin\emptyset_G} = \frac{r_G}{2\sin\theta/2} \qquad \text{Equation 2.8}$$

$$L = r_C\theta \qquad \text{Equation 2.9}$$

2.2 Concurrency in Software

A thread is a flow of execution within a program. Usually a program will only have a single thread and will only be able to complete a single task at a time. Multi-threading is a simple way of creating a separate flow of execution in a program. By making use of threads, a processor can appear to execute two or more processes at the same time. Although the processes appear to be running at the same time, the actual processing time is simply split between the two flows. Therefore, if the processor is already being used at 100%, threading will not speed up the flow of the program. Tasks that rely on and need to wait for external events are often good options for threading as the other threads can run while that part of the program is waiting. (Anderson, 2019)

To thread efficiently with threads that loop infinitely, it is important to implement a wait function to allow for other threads to receive processing time. If the loop never waits it will use unnecessary processing power looping around simple code that does not benefit the program. Using (wasting) processing time in these loops can reduce the productive power of a program.

2.3 Wireless Communication

The inherited project used Wi-Fi to facilitate communications between the AGV and the overhead camera. Wi-Fi makes use of radio waves to communicate information over a network at frequencies between 2.4 GHz and 5.0 GHz, depending on the amount of data that is sent. Typically, Wi-Fi has a range from 10 m to 35 m, but it can be boosted to reach much further (Martindale, 2019).

The protocol used to communicate images is called transmission control protocol (TCP). TCP is a connection orientated protocol, so a connection is created and maintained until both ends of the connection have finished passing messages. The protocol determines how to break up data and provides highly reliable error-free data. (Rouse, 2014)

2.4 Image Processing

2.4.1 Binary Thresholding

Binary thresholding is a method for creating a two-coloured image based on a pixel intensity value. The specific pixel value, known as the threshold value, is compared to the value of each pixel in an image. Pixels above the threshold value are set to the maximum possible value while the pixels below the threshold value are set to zero, as shown in Figure 2.3. Inverted binary thresholding causes the opposite to happen. For colour pixels, the maximum value corresponds to white, while zero corresponds to black. Thresholding can be implemented using the **cv2.threshold()** function in the OpenCV library (OpenCV Development Team, 2019). Thresholding

is useful for isolating objects against a contrasting background and for defining obstacles during map generation.

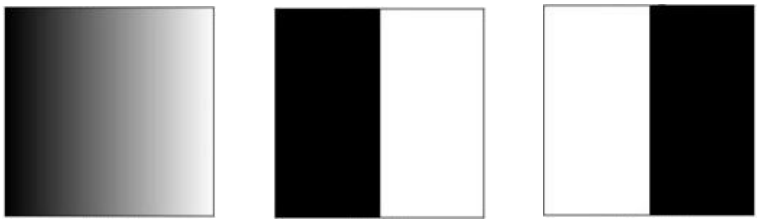


Figure 2.3: Thresholding example (a) original image
(b) normal binary thresholding (c) inverted binary thresholding

2.4.2 Erosion and Dilation

Erosion and dilation are ways of editing an image to enlarge or reduce the size of certain components. Erosion and dilation are normally performed on binary images and make use of a structuring element called a *kernel*. A kernel is a square matrix filled with ones. During erosion the kernel is compared with all the areas on an image and a pixel is only considered a one (maximum value) if all the pixels under the kernel are also one, otherwise it is eroded. Dilation is the opposite; if one pixel under a kernel is a one then all pixels become ones. (Mordvintsev, 2013)

Erosion and dilation can be performed using the functions **cv2.erode()** and **cv2.dilate()** respectively, both of which are in the OpenCV library. Erosion is used during map generation to make obstacles larger. Figure 2.4 shows examples of erosion and dilation.

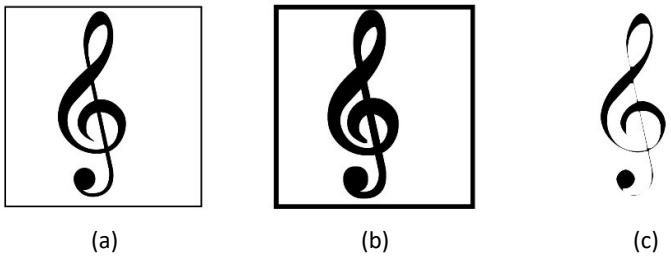


Figure 2.4 Example of erosion and dilation (a) original image (b) eroded image, (c) dilated image

2.4.3 Pattern Identification

Pattern identification is required to find a specific pattern in an image. It is necessary to identify the AGV and destination in images from the overhead camera.

Contour hierarchy is a method of identifying markers by identifying the contours within contours in a marker. A contour is simply a shape or object within an image.

For nested contours, the outer contour is referred to as the parent and the inner contour as the child. By matching a specific pattern of contours from parent to child, markers can be identified and then located within an image to provide positional feedback. (Mordvintsev, 2013) See Figure 2.5.

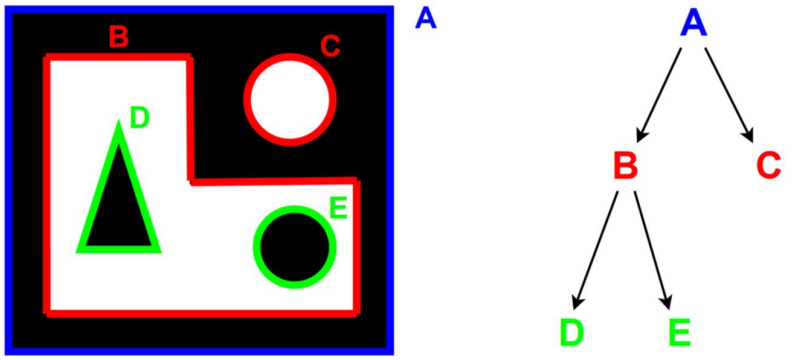


Figure 2.5 Contour hierarchy, (a) contour identification (b) hierarchy tree (Brits, 2016)

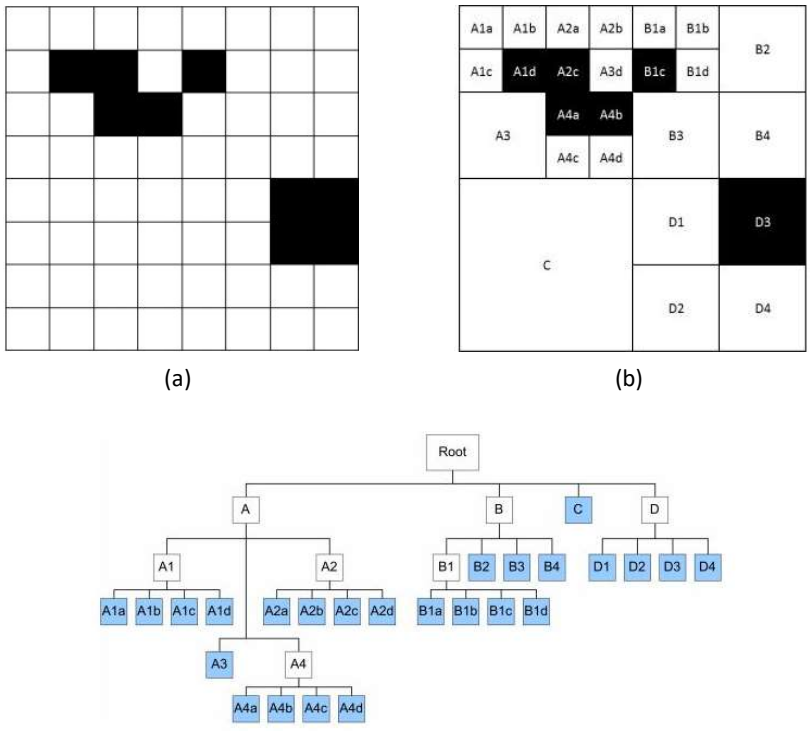
2.5 Path Planning

Path planning is the process of finding a path between two points through obstacles. To complete this process, a map is generated to represent the obstacles and the path's start and end points. A safe path is then found between the points. The inherited project implements a quadtree mapping to define the map and uses the A* algorithm to plan a path between the two points. Although these functions are not updated in this project, it is important to understand how they work in order to fit further updates into the project.

2.5.1 Map Representation

For the purpose of this project, map representations make use of binary thresholded images. Therefore, they are exclusively black and white with obstacles being represented by black pixels. The path planning algorithms require nodes to execute; therefore, maps are represented as a system of nodes. The information in this section is adapted from Yahja *et al* (2000).

Quadtree mapping is used because natural terrain is generally sparsely populated and will often have large areas that are open. The location of the open areas is unknown at the start of processing, so quadtree is a method of defining these areas through mapping. Quadtrees are created by subdividing each non-uniform square on a map into four equal-sized smaller squares. The process is repeated recursively until only uniform squares remain. The uniform squares at the end of the process with no further children are referred to as leaves (Yahja, et al., 2000). The result of the process is shown in Figure 2.6.



(a) (b)

(c)

Figure 2.6 Quadtree map representation (a) original map
(b) quadtree map leaf nodes (c) hierarchy tree (Brits, 2016)

2.5.2 Path Planning Algorithms

The method of path planning implemented by the AGV system is the modified A* algorithm. The algorithm is based on the A* algorithm and is ideal for working with the feedback given by the overhead cameras and stored in a quadtree map.

A* algorithm

The A* algorithm makes decisions based on the distance of a cell from the goal state and the length of the path from the original state. Each cell in a map is evaluated based on the value found in Equation 2.10:

$$f(v) = h(v) + g(v) \qquad \text{Equation 2.10}$$

Where $h(v)$ is the distance to the goal and $g(v)$ is the length of the path from the initial state through the selected sequence of cells. Usable cells are selected based on the quadtree map. The algorithm moves through the map from the start point and calculates the $f(v)$ value of all adjacent cells. The cell with the lowest $f(v)$ value is selected, and the process repeats for all cells adjacent to the chosen cell. The loop continues until the next cell selected is the end goal. (Duchon, et al., 2014)

Modified A* algorithm

A benefit of using the A* algorithm is that the algorithm can be updated to suit the needs of the user. The modified A* algorithm used in the inherited project updated the node calculation to consider the total number of nodes (Brits, 2016), as shown in Equation 2.11:

$$f(v) = h(v) + g(v) + y(v) \qquad \text{Equation 2.11}$$

Where $y(v)$ represents nodes used on the path to that point. By adding this factor, the system selects nodes that will lead to a path with fewer nodes, therefore making the path shorter.

2.6 Path Following

Path following was identified as one of the underdeveloped sections in the inherited project. The inherited system implemented the follow-the-carrot path following algorithm. With the help of a position estimator, more advanced path following techniques could be used to improve both the accuracy and speed of the updated system.

2.6.1 Follow-the-Carrot

The follow-the-carrot algorithm attempts to pick a point ahead of the vehicle on the path and then aim the vehicle towards that point, as depicted in Figure 2.7.

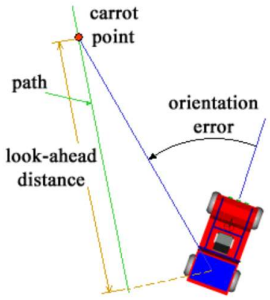


Figure 2.7 Follow-the-carrot description (Lundgren, 2003)

The algorithm draws a line perpendicular to the path, through the vehicle centre. The goal point (carrot point) is defined at a distance ahead of the point where the perpendicular from the vehicle meets the path. The goal point and the vehicle centre are joined by a line. The orientation of this line is compared to the vehicle orientation to calculate an orientation error. Using proportional control, the system attempts to minimise the orientation error by turning the vehicle towards the goal point. The magnitude of the turn φ is determined by Equation 2.12, where k_p is proportional gain and e_0 is the orientation error:

$$\varphi = k_p * e_0 \qquad \text{Equation 2.12}$$

2.6.2 Modified Follow-the-Carrot

The follow-the-carrot algorithm can be updated by predicting a future value. Reynolds (2015) proposes a five-step plan to path following that can be used with the follow-the-carrot algorithm.

1. Predict the vehicle's future position
2. Check if that future location is within an acceptable distance of the path
   If yes, then continue with the current actuation and repeat step 1
   Else:
3. Find the closest point to that future location
4. Find a lookahead point further along the path
5. Seek that lookahead point.

2.6.3 Pure Pursuit

The idea behind pure pursuit is to fit a curve between a goal point and the AGV's current position for the AGV to travel along. The vehicle continually changes its curvature as it constantly fits new arcs based on updated positions and goal points.

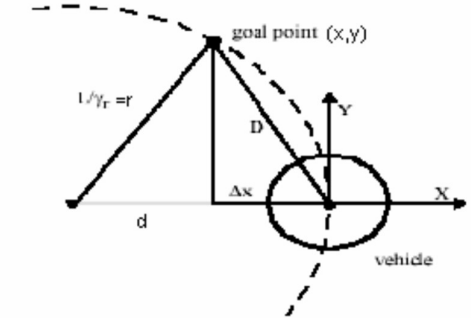


Figure 2.8 Pure pursuit representation (Lundgren, 2003)

An important part of pure pursuit is that the values of D and Δx are both defined in the vehicle coordinate system, where the y-axis is the forward direction for the vehicle and the x-axis is perpendicular to that. Therefore, the values of D and Δx must be calculated relative to the direction of the AGV.

The radius of curvature can then be calculated using the formula in Equation 2.13:

$$r = \frac{D^2}{2 \cdot \Delta x} \qquad \text{Equation 2.13}$$

This technique has shown to reduce oscillations and improve path tracking accuracy compared with follow-the-carrot (Lundgren, 2003).

2.7 Camera Sharing

2.7.1 Calibration and Global Coordinate Systems

Olsen and Hoover (2000) calibrated multiple cameras into a single global coordinate system. The calibration was executed by placing a set of domino patterns in the field of view of multiple cameras. The markings on the dominos show their own coordinate systems. Because the real-world locations of the markings on the dominos are known, by viewing them from multiple cameras, as shown in Figure 2.9, a global model can be created in which each camera covers its own area.

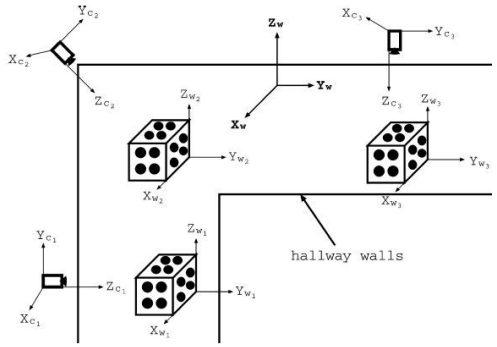


Figure 2.9 Multi-camera system with overlapping views containing calibration targets

In this calibration example, multiple domino patterns were used throughout a large area in order to provide as many calibration points as possible. The domino pattern creates a grid-like pattern which is beneficial because it creates a contrast between the markings and the open space. Furthermore, the domino pattern spans the whole image area evenly and it presents a recognisable structure.

The concept of viewing the same object/s in multiple views and using that to calibrate multiple cameras to a single world coordinate system can be applied to the inherited system in order to add more cameras.

2.7.2 Tracking

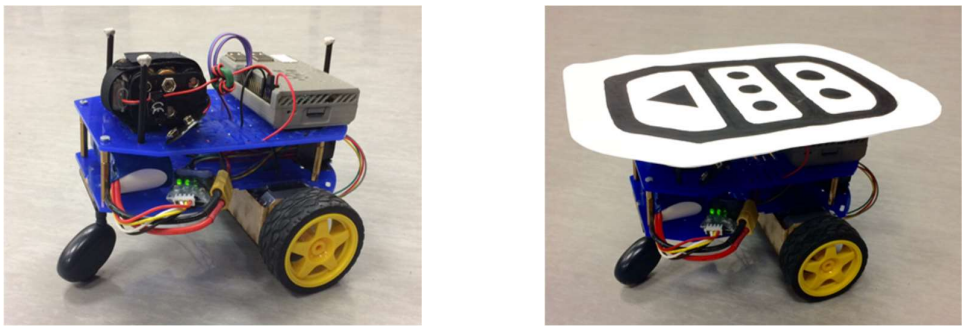
Work by Black and Ellis (2005) tracked objects in multiple views by finding objects that can be seen in all views. For a two-view system they created a list of correspondence pairs for objects that were in both views. Changes in the perceived position of each of the objects were then tracked and compared. The review used already given backgrounds to relate the movement of objects to a real system by subtracting the background and noticing only the movement of new objects.

The system works in 3D while the inherited AGV system used a 2D format for both positional and map related information. Tracking multiple objects in order to create a better picture for calibration and transformation was found to be useful.

3 System Review

3.1 Overview of the Inherited System

The mechatronics project done by Ms. Brits in 2016 is used as the primary basis for this project. The inherited project was able to design, develop and test a self-navigating AGV successfully in a static environment. The feedback system for the AGV was a single camera. The project was able to produce a functioning AGV that could navigate any solvable maze with reasonable consistency. The AGV hardware is shown in Figure 3.1.



(a) (b)

Figure 3.1 AGV hardware, (a) chassis without marker,
(b) chassis with identification marker (Brits, 2016)

The two processors used in the inherited system are Raspberry Pi 3Bs. One processor is used onboard the AGV and the other runs the camera. The two processors communicate with each other, as well as with a separate computer, over Wi-Fi. The AGV receives images taken by the camera and runs image processing on them in order to locate itself, its destination and obstacles. As the image is thresholded before processing happens, black markers are used as obstacles because they are easily noticed by image processing software. The AGV and its destination are identified with markers that can be located using contour hierarchy.

A map is generated by dividing the usable (unobstructed) floorspace into smaller areas. Using this map, path planning is undertaken using the modified A* algorithm. In order to make the path easier for the AGV to follow, the planned path undergoes smoothing to take out sharp corners. Once the path has been defined and smoothed, the system follows the path using the follow-the-carrot algorithm, which attempts to align the AGV with a point further along the path. The movement of the AGV is facilitated by two stepper motors, which drive the two active wheels. A third caster wheel balances the AGV. The use of two stepper motors allows for differential control and the exact path of the AGV is controlled by the different frequencies of the signals given to the motors.

The software for this project was written in python. OpenCV has extensive image processing libraries that work with python and these libraries are beneficial to development. The only feedback given to the AGV comes from the camera system. An illustration of how the system interacts with the overhead camera is shown in Figure 1.2.

3.2 Review of the Inherited System

The inherited system designed, developed and tested a self-navigating AGV. The inherited project's image transmission, object localisation and path planning systems are all at an acceptable level and do not require updating. These sub-systems will be left in their current form and any further updates will be made to work with them.

The hardware configuration of the AGV will not be updated as it is acceptable for continued use. However, the drivers and wiring have not survived storage and will need to be replaced before testing can begin.

The system is limited by the fact that it can only accommodate a single camera in a static environment. The obstacle map of the system is generated during pre-processing (before the AGV moves) and is not updated during the control loop, so the system is only suitable for use in a static environment. To update the system to be used in a dynamic environment would require a large system overhaul. With regards to the single camera feedback system, the system appears to be expandable as the grid that is generated has space for expansion. Given the current software implemented on the inherited system, the logical next step would be to expand the system to involve multiple cameras.

Accepting the limitations on the system, the largest problem with the inherited system is that the control loop can only run sequentially. In the current system flow, shown in Figure 3.2, position control can only occur after an image has been received successfully and the AGV has been located in that image. The system has no other form of feedback.

This lack of feedback options means that if there is an issue obtaining an image or locating the AGV, the execution flow will get stuck. If the flow cannot continue, the position control step will not execute, so the system will not update. If it does not update, the AGV can lose course; causing issues as shown in Figure 3.3. It is difficult for the system to recover from a lengthened break in feedback and the AGV tends to oscillate around the path or, in some cases, loses track completely. This issue has been identified as an area where the inherited system can be improved upon by providing a secondary open loop feedback system that allows position control to be performed at fixed intervals without closed loop feedback from a camera.

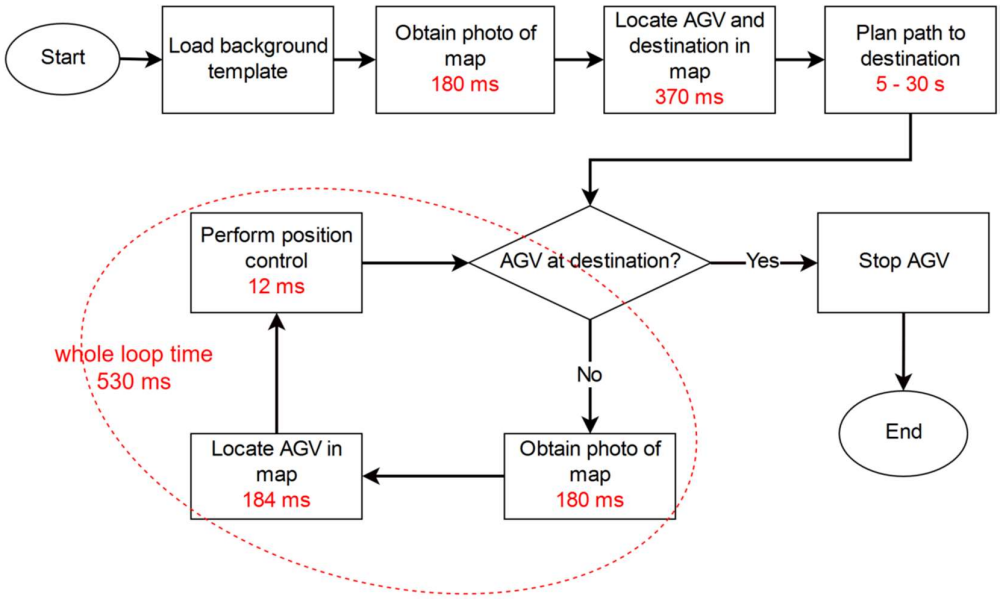


Figure 3.2: Flow diagram of inherited project with execution times (Brits, 2016)

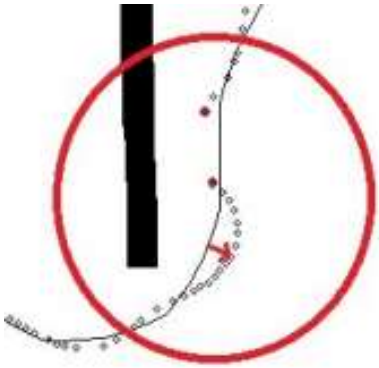


Figure 3.3 Example of loss in transmission leading to oscillation (Brits, 2016)

The path following algorithm employed by the inherited system is not very sophisticated, as the feedback issue limited the accuracy of the path following to an extent where more sophisticated algorithms would have had little benefit. Provided a solution can be found to the feedback issue, this component has been identified as a possible area for improvement.

4 System Definition

This system is a further iteration of previous projects. Since the inherited project was able to design and test a working AGV, this iteration will focus specifically on certain functions from the inherited design that were identified as possible options for improvement. Since the hardware design was not changed from the past project, this report will focus on the software updates that have been made.

4.1 Function Identification

The goal of this project is to improve the AGV system to be able to navigate in a static, multiple camera environment without relying on consistent camera feedback for positional information. The functions identified to enable the system to reach this goal are shown in Table 4.1.

Table 4.1: Identified functions (adapted from Brits (2016))

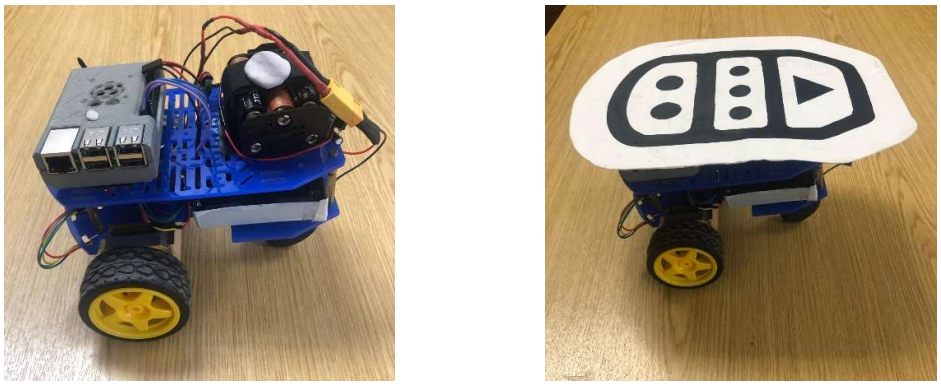
Function	Sub function
1. Capture images from all cameras	
2. Transmit image from selected overhead cameras to AGV	
3. Plot path to destination	3.1 Create a single map of obstacles in all camera views
	3.2 Find position and angle of AGV
	3.3 Find position of destination
	3.4 Plan path from AGV to destination
4. Control AGV's movement along path	4.1 Locate AGV using overhead cameras OR Locate AGV based on signals sent to actuators
	4.2 Generate control signals for path following
	4.3 Actuate AGV

4.2 Subsystem Identification

4.2.1 AGV Hardware Subsystem

No updates were made to the AGV hardware design from the inherited project as it functioned acceptably. The drivers of the stepper motors were replaced and rewired.

The AGV hardware subsystem fulfils function 4.3 in Table 4.1 and provides space for the Raspberry Pi that performs all onboard processing. The AGV hardware is shown in Figure 4.1.



(a) (b)

Figure 4.1 AGV subsystem, (a) without marker, (b) with marker

4.2.2 Camera Network Subsystem

The camera network subsystem comprises two overhead cameras placed so that their fields of view overlap with each other. The cameras connect with the AGV when it moves within range of them and once connected, they begin to capture images of the workspace below. Images are transferred to the AGV when requested by the onboard controller. The subsystem fulfils function 1 in Table 4.1. Figure 4.2 shows a common setup for the overhead cameras.

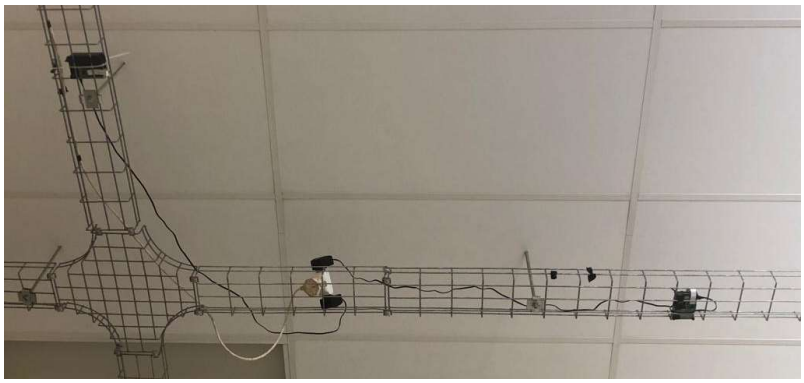


Figure 4.2 Overhead camera network setup in gantry

4.2.3 Wireless Network Subsystem

The wireless network allows the AGV processor to communicate with the two overhead cameras and a separate laptop that is used to program the systems. The wireless network is created using a mobile hotspot, which allows the user to see the IP addresses of each Raspberry Pi in use. Using the IP addresses a virtual link can

be created to allow the user to interact with each Raspberry Pi individually. The wireless network performs function 2 as defined in Table 4.1.

4.2.4 Processing and Software Subsystem

The processing subsystem functions are all performed by a Raspberry Pi 3B+. The 3B+ is a slightly newer and more powerful version of the processer used in the past, however the operating system and physical interfaces are the same as the Raspberry Pi 3B. Figure 4.3 shows the flowchart of the processing required to control the AGV in the camera network. Function 3 and its sub functions, as defined in Table 4.1, are performed during phase one in the flowchart, while the AGV is stationary and the before the control loop begins. Functions 4.1, 4.2 and 4.3, as defined in Table 4.1, are completed in phase two in the flowchart, during the control loop. The final phase of the flowchart is to shut the AGV down safely, disconnect from the overhead cameras and save any recorded results.

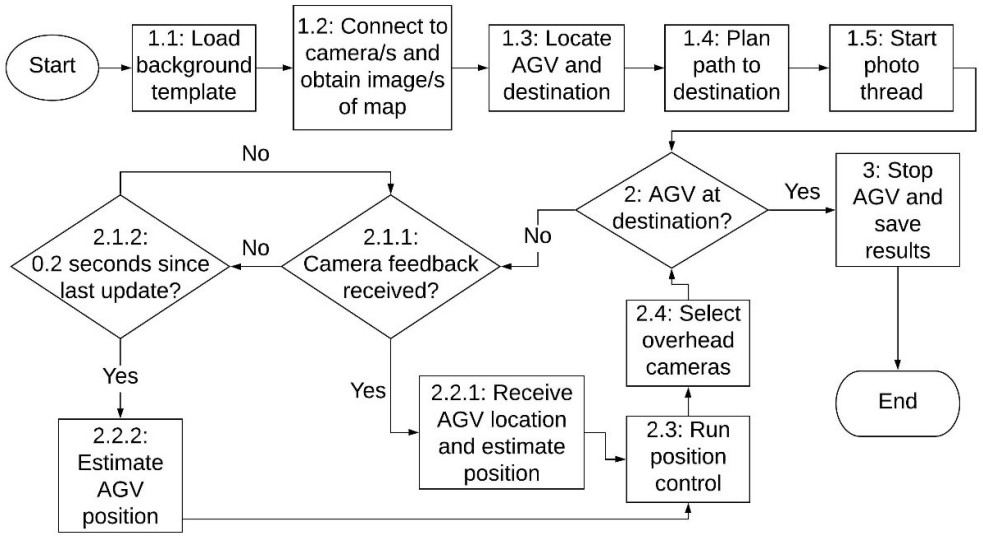


Figure 4.3: Software flow diagram

From this point forward, this report will use Figure 4.3 as the reference to show where changes have been made to the inherited project. The flowchart is adapted from the control loop in Figure 3.2.

5 Position Estimation

A position estimator was added to the system to provide open loop feedback when feedback from the overhead camera network is not available. The position estimator is used in blocks 2.2.1 and 2.2.2 of Figure 4.3.

5.1 Position Estimator Derivation

5.1.1 Assumptions

The position estimator is required to predict the motion of the AGV given specific information. For this prediction to be possible some assumptions must be made:

- The wheels grip perfectly when moving forward and slide perfectly when turning.
- All motion occurs on curves with varying radii from the width of the AGV to infinity.
- The AGV turns at constant angular velocity.
- The stepper motors change frequency instantly.

From these assumptions it can be taken that if one stepper motor is stationary and the other is not, the AGV will turn in a circle of radius equal to the width of the AGV. If each stepper motor has equal frequency the AGV will move forward in a straight line.

5.1.2 Variables

There is important information regarding the specific AGV that is being used that is relevant to the position estimator. These include the step size of the stepper motors in degrees (step), the radius of the wheels used to actuate (r), and the width of the AGV (w).

The frequencies of each stepper motor are used to calculate the distance that each wheel has travelled given a specific time period. This information is used to calculate the translation of the AGV. The output of these variables is an updated position of the AGV.

5.1.3 Mathematical Model

The motion of the AGV in a given time period, dt, can be expressed using the distance travelled by each wheel.

$$L_1 = f_1 \cdot \left(\frac{step}{360} \cdot 2\pi \cdot r\right) \cdot dt \qquad \text{Equation 5.1}$$

$$L_2 = f_2 \cdot \left(\frac{step}{360} \cdot 2\pi \cdot r\right) \cdot dt \qquad \text{Equation 5.2}$$

Given the assumption that the AGV always moves on a curve; Figure 5.1 shows that the outside wheel will travel a larger arc than the inner wheel.

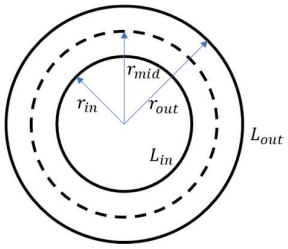


Figure 5.1 Circular path trajectory of an AGV

If $f_2 > f_1$ therefore:

$$L_2 = arc_{out} = r_{out} \cdot d\theta \qquad \text{Equation 5.3}$$

$$L_1 = arc_{in} = r_{in} \cdot d\theta \qquad \text{Equation 5.4}$$

Thus

$$\frac{r_{out}}{r_{in}} = \frac{L_2}{L_1} = \frac{f_2}{f_1} \qquad \text{Equation 5.5}$$

$$r_{in} = \frac{f_1}{f_2} \cdot r_{out} \qquad \text{Equation 5.6}$$

Because the width of the AGV is known, these equations can then be related as

$$r_{out} = r_{in} + w \qquad \text{Equation 5.7}$$

Therefore:

$$r_{in} = \frac{f_1}{f_2} \cdot (r_{in} + w) \qquad \text{Equation 5.8}$$

$$r_{in} - r_{in} \cdot \frac{f_1}{f_2} = \frac{f_1}{f_2} \cdot w \qquad \text{Equation 5.9}$$

$$r_{in} = \frac{\frac{f_1}{f_2} \cdot w}{1 - \frac{f_1}{f_2}} \qquad \text{Equation 5.10}$$

$$r_{in} = \frac{f_1 \cdot w}{f_2 - f_1} \quad \text{Equation 5.11}$$

$$r_{mid} = r_{in} + \frac{w}{2} = \frac{f_1 \cdot w}{f_2 - f_1} + \frac{w}{2} \quad \text{Equation 5.12}$$

$$L_{mid} = \frac{L_1 + L_2}{2} = r_{mid} \cdot d\theta \quad \text{Equation 5.13}$$

$$d\theta = \frac{L_1 + L_2}{2 \cdot r_{mid}} \quad \text{Equation 5.14}$$

To relate the system to Equation 2.8 and Equation 2.9 in section 2.1: $r_{mid} = r_c$ and $\theta = d\theta$. Equation 5.12 is used to find r_{mid} and Equation 5.14 is used to find $d\theta$.

Therefore, from Equation 2.8, the straight-line distance travelled by the AGV is:

$$r_G = 2 \cdot r_{mid} \cdot \sin\frac{d\theta}{2} \quad \text{Equation 5.15}$$

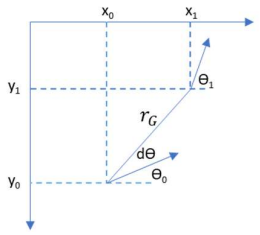


Figure 5.2 Position and orientation changes during AGV movement

Using the values calculated in Equation 5.14 and Equation 5.15 and trigonometric relationships shown in Figure 5.2; equations to describe the AGV's next position $(x_1 \quad y_1 \quad \theta_1)$ are shown below in terms of the current inputs and the initial position and orientation $(x_0 \quad y_0 \quad \theta_0)$.

$$x_1 = x_0 + r_G \cdot \cos\left(\theta_0 + \frac{d\theta}{2}\right) \quad \text{Equation 5.16}$$

$$y_1 = y_0 - r_G \cdot \sin\left(\theta_0 + \frac{d\theta}{2}\right) \quad \text{Equation 5.17}$$

$$\theta_1 = \theta_0 + d\theta \quad \text{Equation 5.18}$$

5.2 Implementation

5.2.1 Physical Properties

The physical dimensions of the AGV are required to make sense of the inputs sent to the motors. They are defined in Table 5.1. Because all processing is done using

pixel values, the dimensions are given in pixels as well as mm. The calculation to convert mm to pixels is shown in Appendix D.

Table 5.1 AGV dimensions

Component	Size	Size (pixels)
Wheel radius	32 mm	6.7 px
AGV width	146 mm	30.7 px
Step size	1.8°	N/A

5.2.2 Software Architecture

The purpose of the position estimator is to enable the system to update without getting stuck requesting an image from an overhead camera. Irrespective of the delay, the system must still request and wait for images from the overhead camera system. To solve this problem, two separate flows of functioning are required. Multi-threading is discussed in section 2.2 of the literature review and it is a suitable multi-tasking option. An additional benefit of this method is that variables are shared through all threads, therefore flags can be used to show that an image has been received, as well as to show that the AGV has reached the destination.

Block 1.5 of Figure 4.3 shows the start of the *photo thread* just before the control loop begins. The photo thread and the control loop run in parallel while the AGV follows the planned path. The photo thread loops through three main actions on repeat. The system first requests and receives an image from the overhead camera and the processor then attempts to locate the AGV in the received image. These actions are shown in blocks 1 and 2 of Figure 5.3. The functions used to perform these actions are unchanged from the inherited project and they use a TCP socket connection and pattern identification respectively. Both functions are discussed briefly in the literature review. If the processor successfully finds the AGV marker in the image, the location is saved and a flag is raised to be noticed by the control loop. Both the location and the flag are stored in object variables that can be viewed by both threads. If the AGV is not found for any reason (e.g. the camera is blocked or the AGV is not in the image) the flag is not raised and the process begins again.

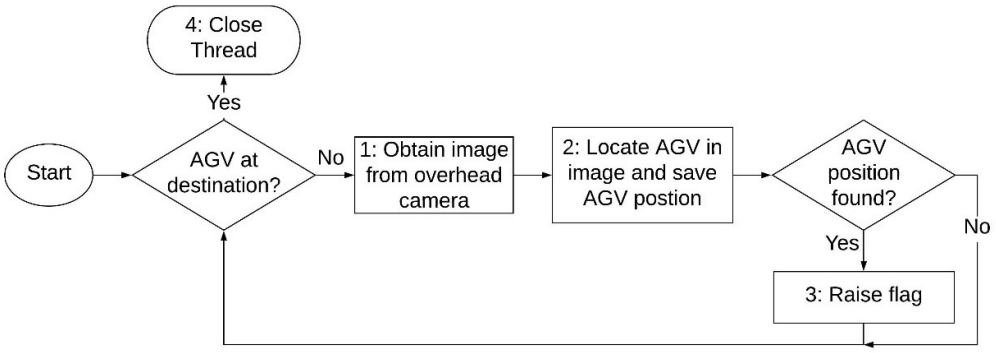


Figure 5.3 Photo thread loop

Using the update presented in this report, the program is now able to update the position of the AGV by executing blocks 2.1.2 and 2.2.2 in Figure 4.3 without successfully receiving a new image. The estimator flag is raised based on a timer that is reset each time block 3 executes and position control is run. Because camera feedback also resets the position estimator, the camera feedback overrules the position estimator feedback. The position estimator feedback is overruled because it is a prediction of position while the camera feedback is sensor information and not a prediction.

The image request action in the photo thread, block 1 in Figure 5.3, must select the correct camera to receive an image from the two-camera system. This decision process will be discussed further in chapter 8.

5.2.3 Software Implementation

The equations derived in section 5.1.3 can be translated into python code in their current form. The goal of the position estimator is to mimic the feedback given by the camera, thus the formatting must be changed to ensure that the system does not pick up a difference between the position estimator output and the camera feedback output.

The position estimation function that has been defined in the code accepts the left and right motor frequencies, the time they act for, the previous position and the previous orientation as arguments. Constants for the wheel radius and the width of the AGV are built into the system and converted from a mm value to a pixel value. The function returns a position and orientation in the same format as would be returned by the camera feedback system.

An issue that needs to be dealt with is that Equation 5.12 returns infinity if $f_1 = f_2$. Code does not deal well with dividing by zero so an if statement is used for when the frequencies are equal.

5.3 Feedback Filtering

Each image that is received by the processor is slightly delayed. This delay is due to the time taken to capture the image, transfer the image and locate the AGV. Therefore, by the time the processor receives information regarding the location of the AGV, the data is between 0.2 and 0.5 seconds old in normal cases and more when there have been issues with image transmission. In the inherited project, this delay was highlighted as a source of problems as the drive signals sent to the actuators will react slowly to changes.

5.3.1 Background

In this project a position estimator has been implemented in order to increase the accuracy of path following and allow the system to update without needing

consistent information from the overhead camera. However, when the position estimator was first implemented, the AGV's path following ability deteriorated and the deviation from the planned path increased. Initially it was assumed that the issue was due to inaccurate information from the position estimator. Further investigation revealed that the position estimator was not the problem. From there it was assumed that the inputs to the position estimator must be incorrect.

Figure 7.9 (b) shows an example of the results that were observed before the forward estimator was implemented. In the figure the small circles are the estimated positions and the larger circles represent image feedback. It is noticeable that the estimated positions do not line up with the image feedback.

It was found that the source of the issues was the delay between the time that the image is captured by an overhead camera and the time when decisions can be made based on that information. The most important information for the position estimator is the initial position as the system calculates the change in position based on the inputs to the motors. The actual translation of the AGV is based not only on the motor inputs, but also on the current position and orientation of the AGV.

For example, take a situation where the AGV is turning left but then receives feedback that tells it to turn to the right. While the software was processing the camera feedback, the AGV would have arced to the left; moving away from its position in the image that is being processed. If a position estimator were to work with the initial camera position it would estimate a position right of the initial position; however, if the true position of the AGV is used, due to changes in position and orientation during image processing, the AGV would have been to the left of the image. This situation is illustrated in Figure 5.4. As can be seen in this extreme example, movement during processing can have a major impact on the estimated position.

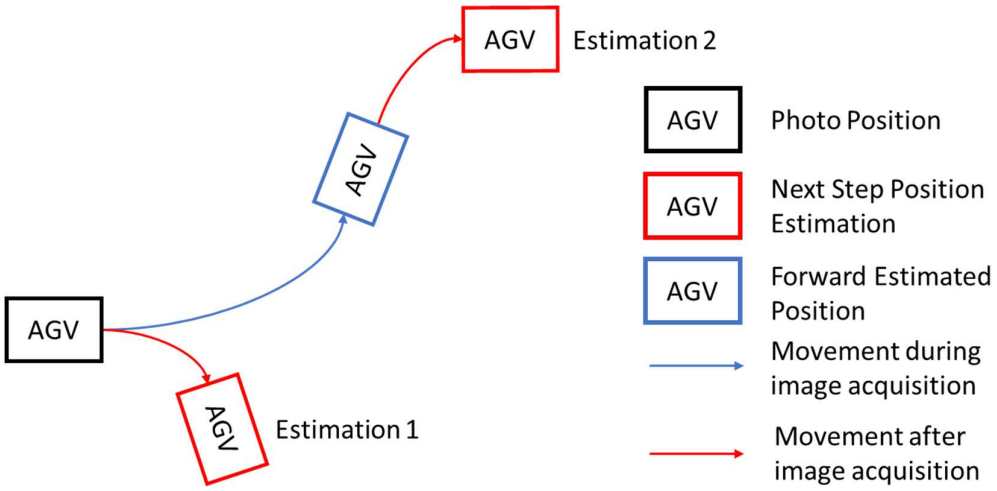


Figure 5.4 Position estimation issue explanation

Furthermore, because the position estimator loop runs every 0.2 seconds while the photo loop takes approximately 0.5 seconds to be received, movement during the delay is the most significant, making this effect even worse.

5.3.2 Solution

To improve the path following accuracy of the AGV and to make the position estimator useful, the received position of the AGV needs to be updated to make a better approximation of its current position. The position estimator is perfect for this application and is used to condition the feedback from the camera. In block 2.2.1 in Figure 4.3, the camera feedback is updated with the position estimator using the amount of time since image capture began as well as the actuation signals sent to the motors during that time. As the motor inputs update more often than images are received, the forward estimator saves the movement during each update and then adds all the updates together with the image feedback to predict the true position of the AGV. Figure 5.5 demonstrates how the system adds each component of the motion to estimate the final position accurately.

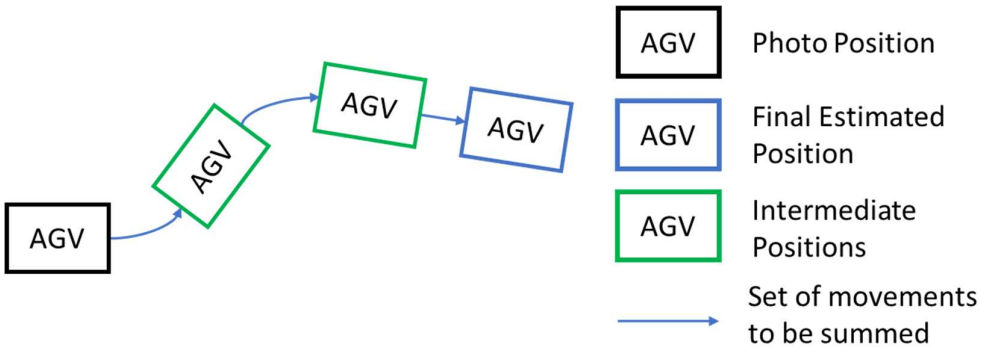


Figure 5.5 Step by step forward prediction

The problem with filtering the feedback in this way is that the position estimator is not completely accurate, so the updated position is not necessarily where the AGV really is. If the position estimator is not functioning correctly and giving non-sensical feedback, the system response will be completely inaccurate. However, the updated position is assumed to be a better guess with respect to the current position of the AGV than the unfiltered feedback, and testing shows a large improvement in the system's response when filtering is added. The overhead computation added by this additional filtering is also minimal compared to the image processing, with the entire forward estimating filter running in less than 0.1ms.

6 Path Following – Pure Pursuit

Past reports identified path following as a means to improve the response of the system. Thus, the third objective of this project is to update the path following algorithm that was implemented in the inherited system.

The path following algorithm is executed as part of block 2.3 in Figure 4.3. The inherited path following algorithm implemented a follow-the-carrot algorithm which is a basic algorithm. Follow-the-carrot is prone to extreme outputs which do not mesh well with the position estimator. It can be improved upon by using an algorithm that takes open loop feedback into account. A modified follow-the-carrot algorithm was implemented and it did improve results; however, the pure pursuit algorithm was clearly superior.

6.1 Pure Pursuit Derivation

Equation 2.13 relates a vehicle's position relative to the path to a specific radius for a curve that the vehicle should travel along. Equation 5.12 then relates that radius to the AGV and the frequencies sent to the motors.

$$\frac{D^2}{2 \cdot \Delta x} = \frac{f_1 \cdot w}{f_2 - f_1} + \frac{w}{2} \qquad \text{Equation 6.1}$$

$$\frac{f_2}{f_1} = \frac{2}{2 \cdot \frac{D^2}{2 \cdot \Delta x} - 1} + 1 \qquad \text{Equation 6.2}$$

Observation of the inherited path following system showed that the system struggled to cope when one wheel's frequency was much larger than the other. Therefore, the ratio in the pure pursuit algorithm is limited to a maximum of 3:1 during normal operation and 5:1 if the AGV starts in the wrong orientation. The larger frequency is also set to a specific value (normally 200 Hz) so that the more the AGV is required to turn the slower it will go.

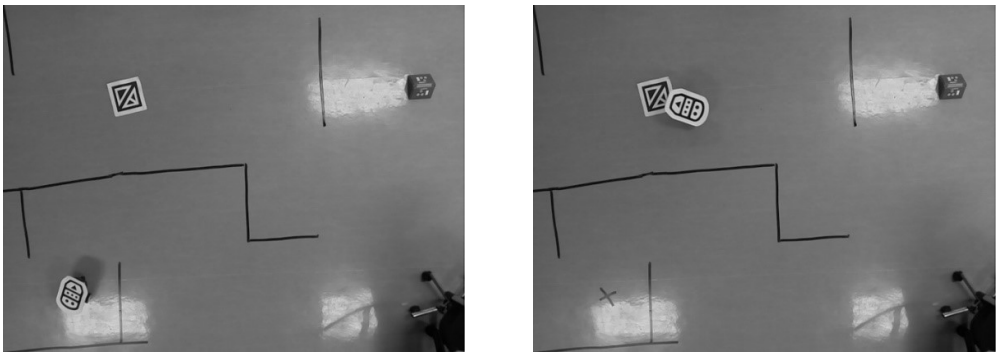
6.2 Implementation

Using Equation 6.2 the correct frequencies are calculated each time the system updates and a new curvature is created. The new path following algorithm returns frequency values in the same way as the previous system, thus minimal changes were required to fit it into the system.

7 Evaluation of Path Following Update

7.1 Test Procedure

To run a test on the system, a map of the workspace is loaded onto the Raspberry Pi on the AGV and the AGV (with its marker in place) and the destination marker are placed within the field of view of the camera. The physical setup for a test is shown in Figure 7.1. The map is an image of the workspace without any markers in view. The AGV opens a socket connection with the overhead camera, and executes the flow described in Figure 4.3. In block 3, of Figure 4.3, results of the test are saved. Results that are used in this chapter include: overhead camera feedback, position estimator feedback, path planning time, movement time, number of images taken and the maximum distance the AGV deviates from the path. Unless otherwise stated, all tests use the pure pursuit path following algorithm.



(a) (b)

Figure 7.1 Unedited image feedback from camera at (a) start and (b) end of testing

7.2 Comparison of Path Following Methods

Three path following methods were investigated in this project. The follow-the-carrot algorithm was implemented on the inherited system, a modified follow-the-carrot (FTC) algorithm was implemented and tested briefly, and a pure pursuit algorithm was implemented and tested extensively. While the modified FTC algorithm provided some improvement to the system, it was immediately obvious that the pure pursuit algorithm was superior in both accuracy and speed.

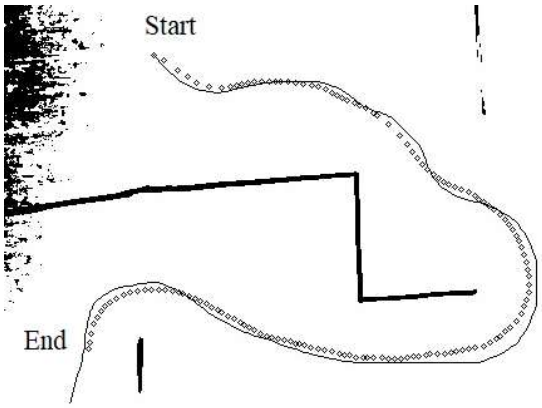


Figure 7.2 Path following with follow-the-carrot path algorithm

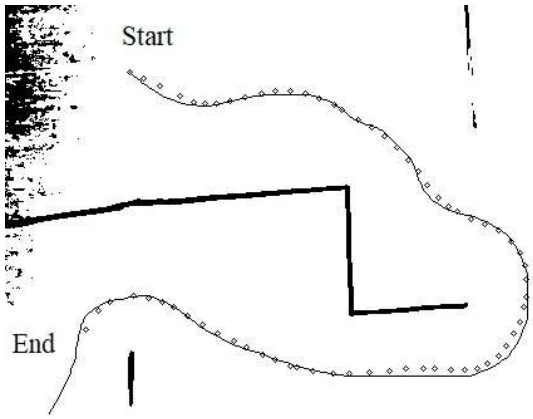


Figure 7.3 Path following with pure pursuit algorithm

Both sets of results are implemented along with both forward estimation and position estimation so they both receive high quality feedback from both the open and closed loop feedback systems. Table 7.1 compares the results of the three path following algorithms. In all categories the pure pursuit algorithm responded the best, increasing both speed and path following accuracy. For reference, the maximum deviation from the path was less than half the width of the AGV. In addition, by capping the maximum motor frequency at 200 Hz the AGV no longer overshoots the path as it was prone to do when using the follow-the-carrot algorithm implemented by the inherited system.

Table 7.1 Comparison of path following algorithms

Algorithm used	**Average time**	**Max distance from path**		**Average no. images**
Follow-the-carrot	48,95 sec	12,86 px	61.2 mm	140
Modified FTC	37,37 sec	11,7 px	55.6 mm	109
Pure pursuit	23,29 sec	9.8 px	46.6 mm	70

7.3 Position Estimator Evaluation

The position estimator is always active during testing; however, under good Wi-Fi conditions it is not a necessity as feedback is received from the overhead camera at regular intervals. Therefore, in order to test the position estimator, camera feedback must be artificially reduced.

In Figure 7.4 the camera feedback is completely shut off. The AGV only receives its initial coordinates and the coordinates of the destination. By removing the ability for the photo thread to raise a flag, block 2.1.1 in Figure 4.3 never returns yes so only the open loop feedback system is used.

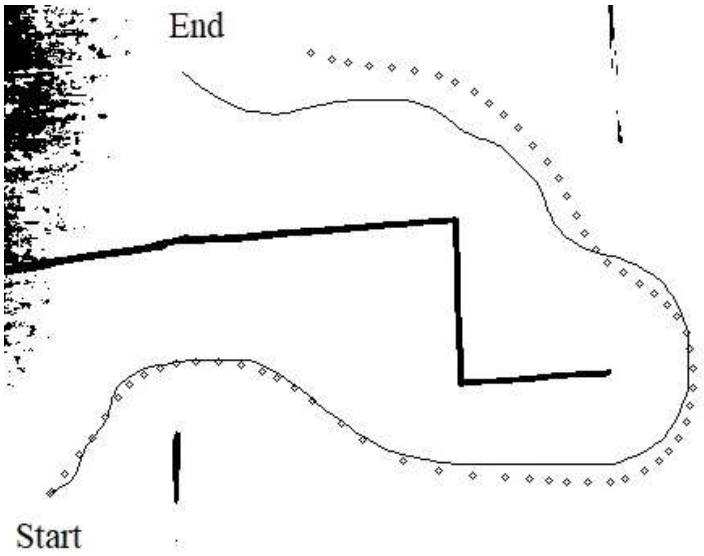


Figure 7.4 AGV path following results without any camera feedback

With only open loop feedback the AGV can track the path throughout the course however drift begins to show after the first corner and increases throughout the movement. Even after the AGV is no longer on the path it still follows a similar shape as in its own feedback system it is still on the path, so it continues to follow the same pattern.

In Figure 7.5, camera feedback is limited at stages during the AGV's movement. Specifically, in Figure 7.5, feedback from images 5 to 20 and 35 to 55 is not used to update the AGV's position.

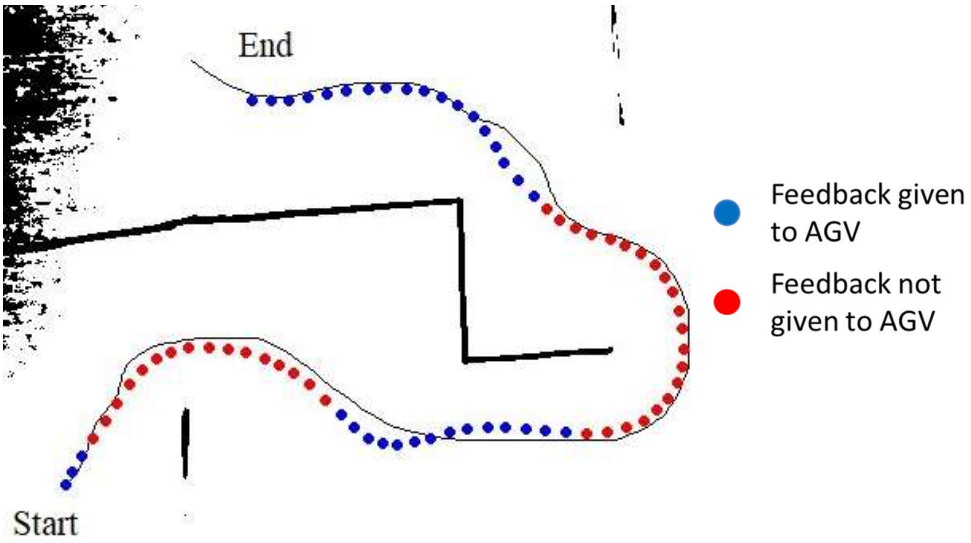


Figure 7.5 AGV path following results with intermittent camera feedback

The AGV can successfully follow the path to the destination without running into obstacles when feedback is limited in this way. During periods when the AGV's feedback is cut off, the AGV can track the path and the system is able to recover without oscillation within five to ten received images. Upon closer inspection, it can be seen that even though the AGV is close to the path at the end of both periods without feedback, it continues away from the path just after the feedback is returned. These results may show that the position estimator's guess for the orientation of the AGV is less accurate than its assumed motion. Thus, while the position is still relatively accurate, the orientation is no longer directed along the path. This result correlates with the result of the zero-feedback test, depicted in Figure 7.4, which shows how the distance travelled by the AGV is similar to the distance that would have been travelled had feedback been provided. The main reason for the errors is the position estimator's orientation assumptions.

In Figure 7.6, camera feedback is delayed by two seconds. This is done by making the photo thread sleep for two seconds after image processing is completed and before the image received flag is raised, and the image feedback is received by the control loop. This delay falls just after block 2 in Figure 5.3. By placing the delay after the image has been taken the system must be able to handle old information as each image is more than two seconds old during which time the AGV will have moved. This test simulates an extremely slow Wi-Fi connection and demonstrates the system's ability to handle non-ideal feedback.

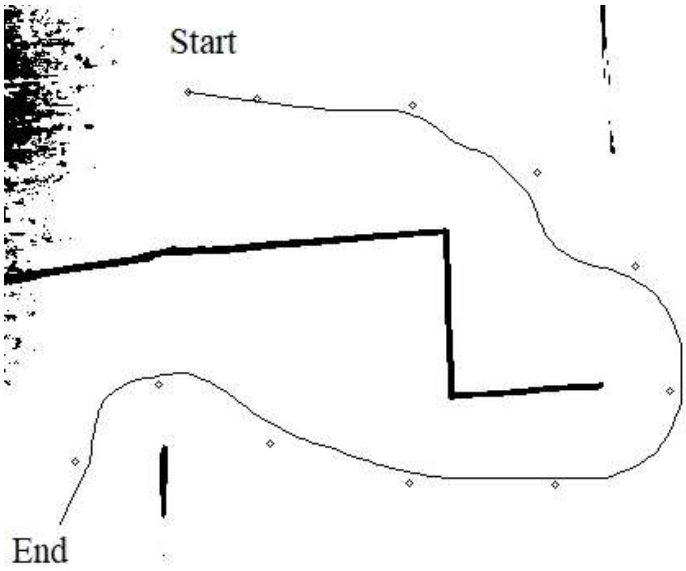


Figure 7.6 AGV path following with all feedback delayed by two seconds

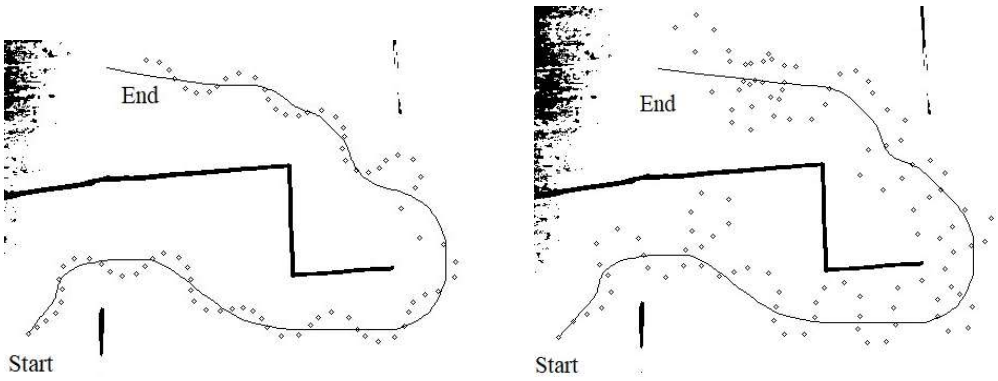
By using the position estimator for open loop control the AGV can successfully follow a given path under these non-ideal conditions. A sample of four tests was run and each time the AGV successfully reached the destination without hitting any obstacles. The results of the tests are shown in Table 7.2.

Table 7.2 Results of two second delay test

Test no.	**Movement time**	**Max distance from path**		**No. images**
1	24.6 sec	17.6 px	83.8 mm	11.0
2	22.7 sec	23.7 px	112.8 mm	10.0
3	23.7 sec	13.1 px	62.2 mm	10.0
4	23.9 sec	23.8 px	113.2 mm	11.0
Average	23.7 sec	19.6 px	93.0 mm	10.5

The results show that, under conditions where feedback is severely limited, the system does not cause the AGV to deviate from the path by more than 113.2 mm, which is less than the diameter of the AGV. This deviation is not great enough to cause the AGV to run into any obstacles. These results demonstrate how the system successfully fulfils the first objective of this project, as open loop feedback from the position estimator is used to guide the AGV when camera feedback is compromised. During testing, the AGV began to fail to reach the destination and to oscillate when the delay was set to five seconds. Running tests with a delay shows the importance of the forward estimator to the system as the received position must be updated to account for the movement while the photo thread was delayed. The forward estimator will be discussed further in section 7.4.

Figure 7.7 shows how the inherited system functions under 0.2 and 0.5 second delays on the control loop. A 0.2 second delay is comparable to testing on an overcast day, which appeared to make the Wi-Fi slower. If the delay is increased to 0.5 seconds movement becomes highly erratic and the AGV cannot be guaranteed to reach the destination.

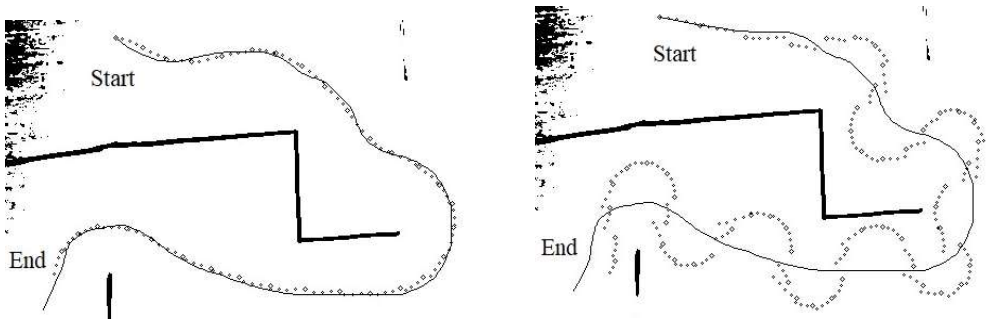


(a) (b)

Figure 7.7 Inherited system with feedback delayed by (a) 0.2 sec and (b) 0.5 sec

7.4 Evaluation of Forward Estimator

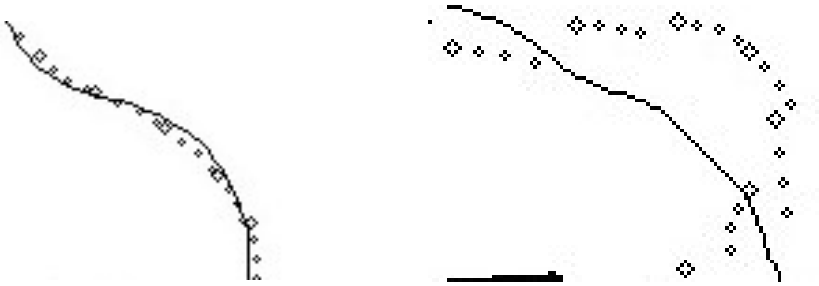
The path following ability of the system with and without forward estimation is compared in Figure 7.8. Figure 7.9 gives a closer view of results shown in Figure 7.8. In both Figure 7.8 and Figure 7.9 the large dots represent closed loop camera feedback and the small dots open loop feedback. Figure 7.8 clearly shows that the forward estimator is critical to the functioning of the system. Under normal conditions (without a delay) the AGV does not oscillate as much as in Figure 7.8 (b) but the overall system functioning is compromised, and the benefit of the forward estimator is reduced.



(a) (b)

Figure 7.8 Comparison of the system response for a 0.5 second delay (a) with forward estimation and (b) without forward estimation

As can be seen in Figure 7.9, when the forward estimator is used the open loop feedback positions line up with the AGV's trajectory along the path. Often the last estimated position overlaps the next received position, showing that the predictions are accurate, and the feedback being received is good. However, when the forward estimator is removed, the feedback no longer lines up and the new information is different from past information leading to oscillation and inaccurate path following.

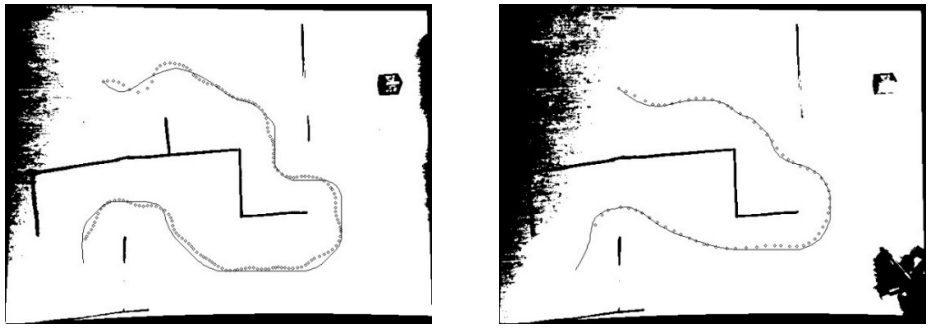


(a) (b)

Figure 7.9 Closer view of results (a) with forward estimation and (b) without forward estimation

7.5 Summary of Results

The main differences between the responses of the inherited system and the updated system are the speed of the AGV and the robustness of the system to inconsistent feedback from the overhead camera. The updated system is more than twice as fast as the inherited system, completing the same course in an average of 23.3 seconds compared with 48.4 seconds. The accuracy of the updated system is also slightly better with a maximum deviation of 9.8 px in four tests compared with 12.9 px in the inherited system. The inherited and updated system responses under normal conditions are shown in Figure 7.10. Importantly, when a 0.2 second delay was placed on the updated system the deviation did not increase, while the inherited system began to oscillate. This oscillation is shown in Figure 7.7 (a) and the maximum deviation increased to 32 px, three times that of the updated system.



(a) (b)

Figure 7.10 Comparison of systems (a) inherited system (b) updated system

8 Multiple Cameras

A single camera covers an area of approximately 3.1 m x 2.3 m. This area is not large enough to cover a room or a factory floor. Therefore, multiple cameras are required to use this system in any practical application. The issues that need to be dealt with are object tracking through multiple cameras, camera selection and connections, map generation and path planning. For the purposes of this project, it is important that the new map and multiple camera system can integrate with the inherited system so that the same path planning algorithm can be used.

This chapter will cover transformations to a shared coordinate system, two-camera map generation, camera communications and give an overview of how the concept would work in a system with more than two cameras.

8.1 Transformation and Calibration

8.1.1 Background

The first issue that needs to be dealt with when converting from a single camera to a multi-camera system is the coordinate system. If the processor does not have information about how the cameras relate to each other it will not be able to undertake path planning or path following. To solve this problem a global coordinate system is created. OpenCV image processing can be used to obtain the location of an object within an image in the image's own coordinate system and those coordinates must then be transformed into the global coordinate system. Therefore, a function must be created that can take coordinates from one coordinate system and transform them into another. In the two-camera system, the first camera's coordinate system becomes the global coordinate system and the second camera's feedback is transformed. The transform must work for any arrangement and orientation, such as the layout shown in Figure 8.1. It is important to note that the images received from the camera are distorted by the lens and must be undistorted by the processor. Removing distortion leads to imperfectly shaped images which are difficult to align, so transformation is required.

To create a global coordinate system, the cameras' fields of view must overlap with one another so their relative position can be calculated. The overlap makes sense because the AGV should never be out of camera vision. By placing a marker in the area that overlaps (the blue circle in Figure 8.1), the system can use contour hierarchies to find the location and orientation of the marker in both coordinate systems. The location and orientation of the marker provides enough information to the processor to lock the coordinate systems to a specific arrangement. The required transform can then be derived to provide the coordinates of an object in camera two's field of view (e.g. the red circle in Figure 8.1) in the global coordinate system.

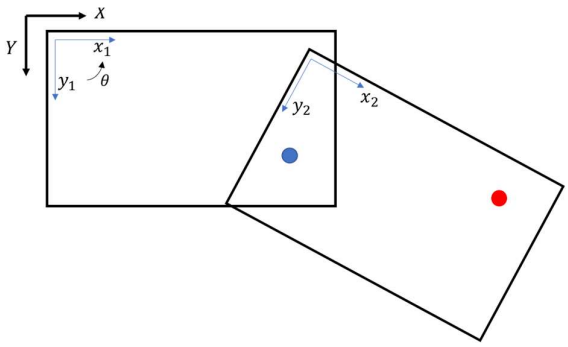


Figure 8.1 Global coordinate system

8.1.2 Transform Derivation

Let the coordinates of the blue marker be $(x_1 \quad y_1 \quad \theta_1)$ and $(x_2 \quad y_2 \quad \theta_2)$ in cameras one and two respectively. Given the red marker at coordinates $(a \quad b \quad \emptyset)$ in the second camera, this section will derive equations to find the coordinates of the red marker in the first camera's coordinate system, $(X \quad Y \quad \theta)$. Figure 8.2 defines the distances that will be calculated during the transform derivation.

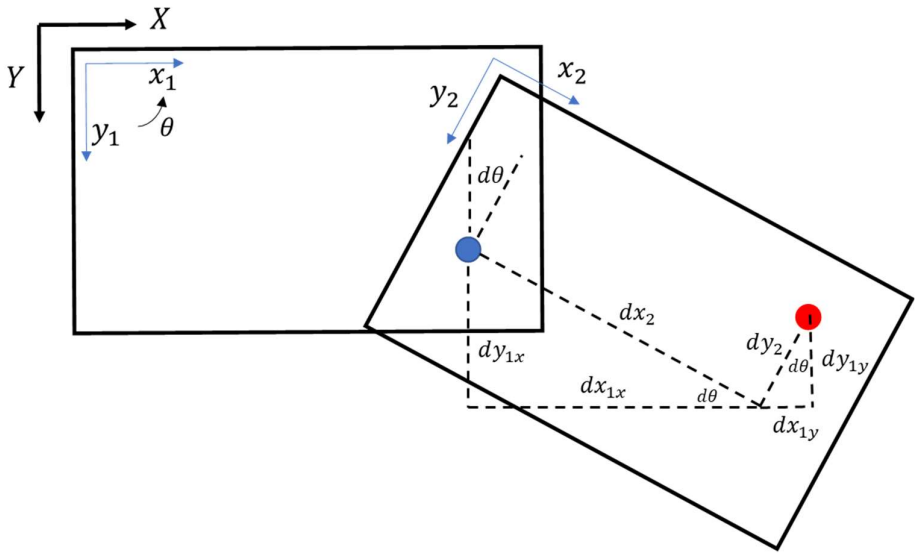


Figure 8.2: Transform derivation

$$dx_2 = a - x_2 \qquad \text{Equation 8.1}$$

$$dy_2 = b - y_2 \qquad \text{Equation 8.2}$$

$$d\theta = \theta_2 - \theta_1 \qquad \text{Equation 8.3}$$

$$dx_{1x} = dx_2 \cdot \cos d\theta \qquad \text{Equation 8.4}$$

$$dy_{1x} = dx_2 \cdot \sin d\theta \qquad \text{Equation 8.5}$$

$$dx_{1y} = -dy_2 \cdot \sin d\theta \quad \text{Equation 8.6}$$

$$dy_{1y} = dy_2 \cdot \cos d\theta \quad \text{Equation 8.7}$$

With the differences in place the transform can be read off. For a specific point $(a \quad b \quad \emptyset)$, the result in camera one's coordinates are:

$$X = x_1 + dx_{1x} + dx_{1y} = x_1 + (a - x_2) \cdot \cos d\theta - (b - y_2) \cdot \sin d\theta \quad \text{Equation 8.8}$$

$$Y = y_1 + dy_{1x} + dy_{1y} = y_1 + (a - x_2) \cdot \sin d\theta + (b - y_2) \cdot \cos d\theta \quad \text{Equation 8.9}$$

$$\theta = \emptyset + d\theta = \emptyset + (\theta_2 - \theta_1) \quad \text{Equation 8.10}$$

8.1.3 Calibration

Using Equation 8.8, Equation 8.9 and Equation 8.10, feedback from the second camera can be updated to fit the correct coordinate system. Because the operations used are simple processes (addition, multiplication and basic trigonometry), the transformation code can execute in a relatively small amount of time (less than 0.1 ms). As a result, it does not have detrimental effects on the overall system when positional information is received and subsequently transformed.

The values required to calculate the transform are measured using the same marker from the AGV. The marker is placed in the overlapping area of the two cameras and images are captured. Examples of images captured for calibration are shown in Figure 8.3. Once distortion has been removed from the images from both cameras, object localisation is executed to determine the coordinates of the marker in each coordinate system. Those coordinates are used to calculate the transform in a process that will be referred to as *two-camera calibration*.

(a) (b)

Figure 8.3 Example images from overhead cameras, (a) camera one view, (b) camera two view

8.1.4 Implementation

Figure 8.4 shows how block 5a has been added to the original flow in Figure 5.3 to expand the program to check which camera was used in block 1 and react accordingly by running block 5b to transform the result to the global coordinate system if required.

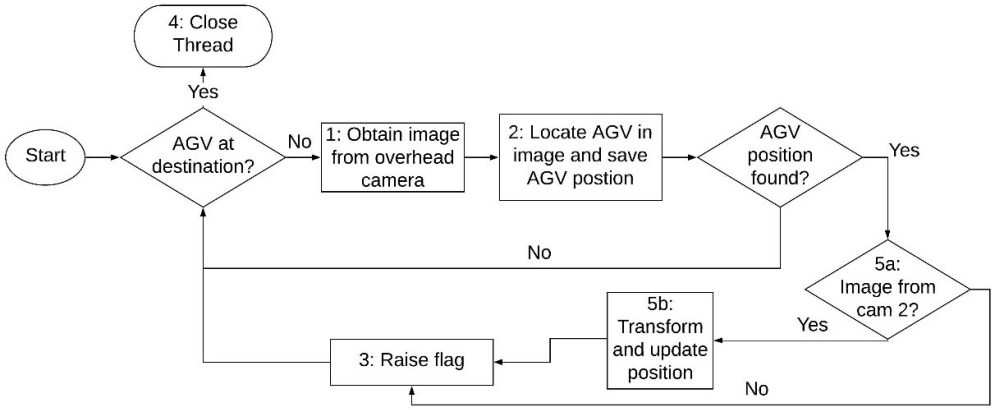


Figure 8.4: Updated photo thread

Once the transformation function has been implemented, the program is able to complete blocks 1.2 and 1.3 of Figure 4.3 by obtaining images from both cameras in the two-camera system (or by running through all cameras in a multi camera system) and by running object localisation on both images. If either the destination or the AGV is found in camera two's view, the camera feedback is transformed into the global coordinate system. In this way the flow of code is relatively unchanged from the single camera flow. As the program runs through the images sequentially, if an object is found in one view the program will cease searching for it in further views.

The updated system is designed to accommodate multiple cameras. The transform code is always a single step calculation, so required computing power does not increase with increased numbers of cameras. Further, objects in the second camera undergo the same transformation, so the relative position of the AGV and the destination is unchanged in the second coordinate system. This statement will hold for objects in any other cameras. Transform accuracy is still required because the map is generated separately to the transform. The same information is used for both transformation and map generation, but both systems need to be implemented accurately or else the AGV will misjudge obstacles.

8.2 Map Generation and Path Planning

In block 1.1 of Figure 4.3 the system is required to load the background template. Essentially, this step entails loading an image of the workspace with only the static

obstacles in place. The program then runs blocks 1.2 and 1.3, where the AGV and destination are located and their coordinates are saved. The coordinates, along with the background image, are used to plan a path for the AGV to follow. For a single camera system this image is of the camera's view without the AGV or destination in place. For the two-camera system, the image needs to be created by connecting the images from the cameras, again, without the AGV or destination in view.

8.2.1 Map Generation

From the calibration, the relationship between the two images is defined. The point of overlap and the orientation of the images relative to one another are known. As demonstrated in Figure 8.5, if the images can be placed on top of one another and then rotated about the specific point of overlap, a single, larger image can be constructed.

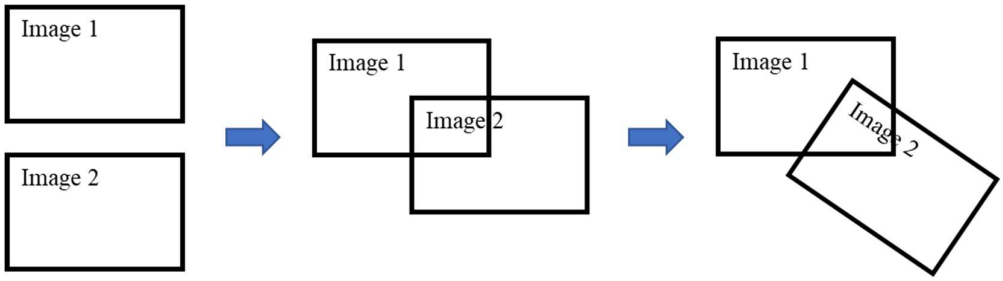


Figure 8.5 Image overlap and orientation

Unfortunately, a method of doing this simple action was not found in the OpenCV libraries. Consequently, the addition of the two images could not be completed using only python commands and it was necessary to use a separate program. The program used is called *Gimp* and the steps to add the two images together are defined in Appendix E.

For block 1.4 of Figure 4.3 to plan a path successfully, the algorithm must receive a thresholded image of the background. Therefore, the process explained above is done using undistorted images that have already been thresholded. Using thresholded images simplifies the process as detail is removed from the image, allowing for more accurate joining. Once the images have been joined, the single image is saved to the processor on the AGV where it can be accessed by the system. Explanation of this process and example images are shown in Appendix E.

This method of map generation is not ideal and a python function that automatically combines the images would be preferable. However, this action only needs to be performed once during calibration. From there the program never needs to join images again as the cameras are run separately and are related to one another using the transform derived in section 8.1.2.

8.2.2 Path Planning

The method of map generation described in section 8.2.1 allows the path finding algorithm to run without changes from the previous project. The only update is that after the image is merged it must be thresholded once more as the software used does not perfectly create a new black and white image. In initial tests the number of nodes created during mapping increased from approximately 4 000 nodes in a single camera map to approximately 36 000 nodes in a two-camera map that was less than twice the size. This increase in the number of nodes led to long processing times for path planning, if path planning was successful at all. By thresholding the final image again, the number of nodes was reduced to approximately 8 500, showing a linear growth in the number of nodes with increases in map size, which is to be expected. Due to the increase in number of nodes, the path planning time is expected to increase with a larger map.

8.3 Camera Communication

8.3.1 Background

The onboard processor needs to do two things in order to receive an image from a specific overhead camera. Firstly, the processor needs to open socket connections between itself and the camera. The implementation of the socket connection means that once the connection has been opened, the camera immediately begins to capture the workspace below it continuously. Secondly, the processor must request an image from the overhead camera. If the connection is already open, this step is trivial to implement although time consuming to complete.

For each camera, the processor needs to decide if the TCP socket connection should be open or closed. This decision must be made for each camera in the system. Multiple socket connections are required because opening a new connection takes time. Connections cannot be opened as the AGV leaves one camera's field of view because there would then be a time without image feedback. While the position estimator would allow the system to continue, this situation is not ideal. Furthermore, multiple socket connections are possible using a Raspberry Pi.

In initial iterations of the design, the onboard processor connected to both cameras constantly. While constant connection is possible for a two-camera system, this approach is not scalable to larger multi-camera networks.

The processor also needs to decide from which camera it must request images from. This decision can only have one result as the processor can only request one image at a time.

8.3.2 Boundary Definitions

In the updated system, three boundaries are defined within the field of view of each camera: a *hold-on* range, a *camera-connect* range and a *camera-disconnect* range. These three ranges are required to ensure that the AGV's system does not jump between cameras or connections. The system should make decisions and stick to them, especially with socket connections. If a single line were to be drawn and the system were to plan a path along that line (unlikely but plausible); the system would have to open and close socket connections very quickly, leading to increased overhead as well as increased chances of crashing the software or a failed connection.

These ranges also handle inaccuracies in the transform. Since the transform is not perfectly accurate, the situation shown in Figure 8.6 could occur where the first update after a camera swap would return coordinates that should be in the previous cameras range. By implementing a hold-on range, the transformed feedback (the red dot in Figure 8.6) would still be in camera two's hold on range and the connection would not swap. Examples of a correct camera pass with the hold-on range and an incorrect camera pass with a single line where the camera connection jumped between the two cameras are shown in Appendix F.

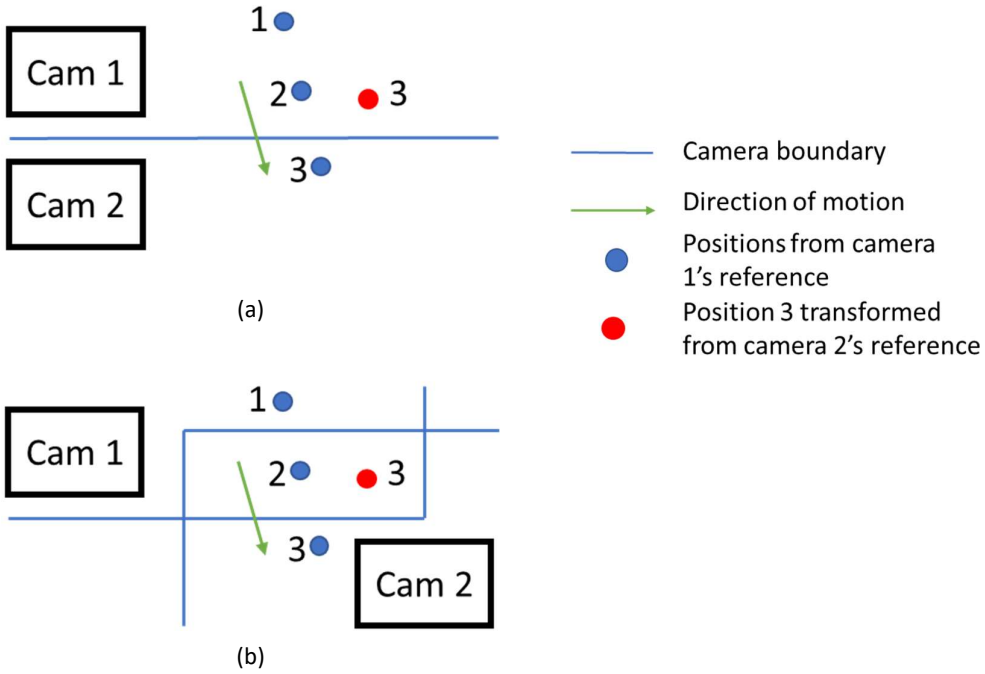


Figure 8.6 Explanation of why hold-on ranges are required
(a) single line boundary (b) hold-on range boundary

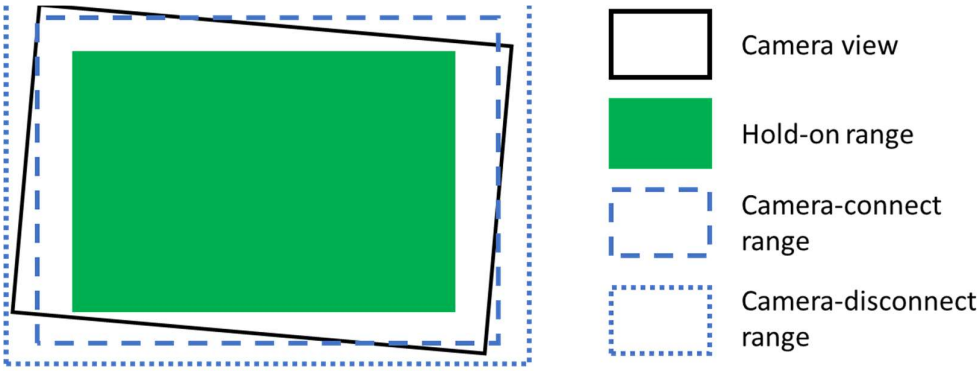


Figure 8.7 Boundary definitions

Figure 8.7 shows the relationships between the ranges defined in this section. As can be seen, the camera view entirely covers the hold-on range, but not the camera-connect and disconnect ranges. The AGV should always be visible within the hold-on range so the range must fall within the camera's field of view. The connect and disconnect-ranges are based on the hold-on range but as the AGV is not required to be visible when connecting to a camera they do not depend on the camera's view. The camera's view is assumed to be an irregular shape at any orientation. In practice the cameras should be arranged to best fit the workspace.

The hold-on range of a camera is a predefined area where the camera can locate the AGV. This range can be defined before or after the cameras have been set up. If it is defined after the cameras have been placed, the field of view can be measured and the range defined based on that information. Alternatively, the range can be defined before the camera is placed as a specific area in which the camera must be able to view the AGV. Defining the range before the camera is placed is less efficient, because the area must be smaller to guarantee vision; nevertheless, it is more practical to define a grid to cover a workspace and then place cameras to fit into that grid.

While the AGV is in the hold-on range of a camera it will request images from that camera exclusively. Camera views are required to overlap so the hold-on ranges of different cameras will also overlap, but the AGV is instructed to switch to a new camera only if it leaves the current camera's hold-on range. Because the ranges overlap, the AGV is in the next camera's hold-on range immediately. The updated system only utilises two cameras, so the algorithm to select the next camera is trivial as there is only one other possibility for camera selection. Larger systems will require a more complex decision. This decision will be discussed in section 8.4.

The camera-connect and disconnect ranges define where the system will connect to a specific camera. They are defined for each camera and the disconnect range must always be larger than the connect range and both must be larger than the hold-on range. Whether the AGV is connected to a specific camera is stored in a local variable in the program. If the camera is already connected to the AGV, the system

only checks the disconnect range. If the AGV moves outside the disconnect range, it will disconnect from the camera and the camera will go into a standby state, where it stops capturing images, but retains the same socket information. Should the camera reconnect, the socket port remains the same. The connect range works in the opposite way. If the local variable shows that the AGV is not connected to a camera and the AGV moves to within the connect range, the AGV will open a socket connection to the overhead camera and the camera will begin to stream the workspace so that images can be requested.

The interaction of a two-camera network with an AGV moving from one camera to the next is shown in Figure 8.8. Initially, the AGV is only connected to camera one and receives images from camera one. As the AGV comes into range of the second camera, it connects; however, it still receives images from the first camera. It then leaves the hold-on range of the first camera but remains connected. Finally, it disconnects from the first camera and is only connected to and receiving images from the second camera.

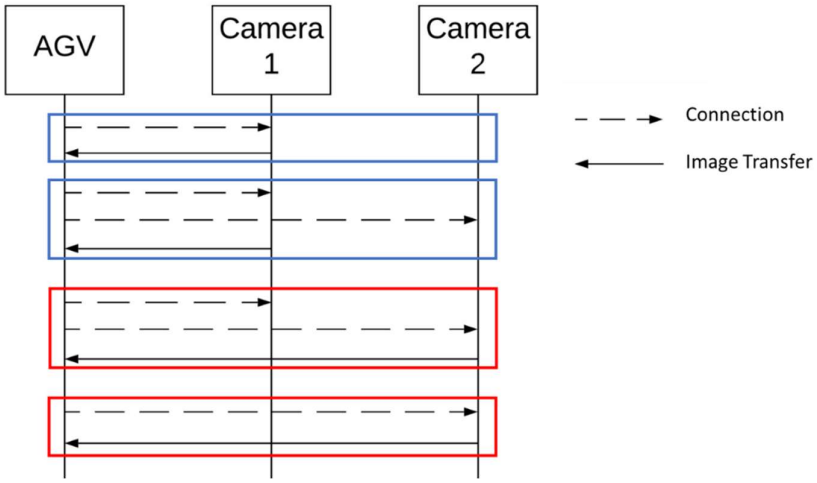


Figure 8.8 Sequence diagram for an AGV moving from camera one to camera two

8.3.3 Implementation

Camera connection and selection decisions are run in block 2.4 of the program flow defined in Figure 4.3. By defining the ranges in the global coordinate system, these decisions can be made using the AGV's positional feedback from the camera. Figure 8.9 shows an example of how the three ranges would be arranged on a sample background. Camera selection and camera connections are run as separate functions in the code implementation of the program.

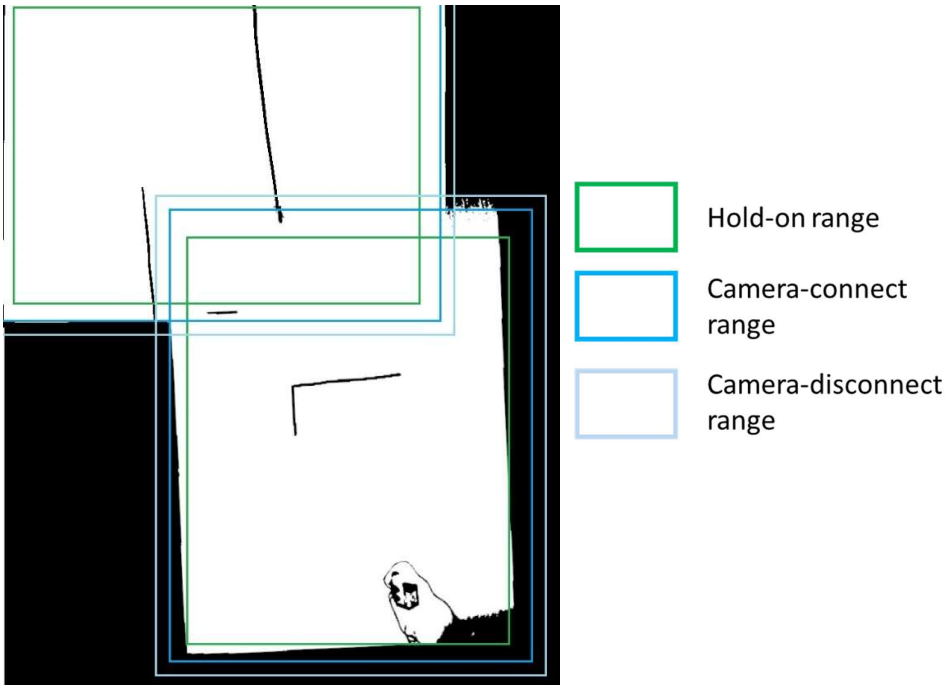


Figure 8.9 Implementation of three ranges on sample background

The camera selection function that is run as part of block 2.4 decides, based on position, which camera to request from and sets a global variable to one or two depending on the result. This decision is based on the hold-on ranges defined in section 8.3.2. At the start of the photo loop the global variable is copied into a local variable that is used in blocks 5a and 5b from Figure 8.4. Firstly, it is used to select the correct camera, and secondly, to decide whether or not to transform the camera feedback. The global variable is stored in a local variable in the photo thread so that if the camera selection function changes the global variable during the photo loop, the system continues to use the original information. This step is necessary because if the information is not stored in a local variable, a situation could arise where the incorrect transform is used because the global variable changed while an image was being received.

The camera-connect function also runs as part of block 2.4. In the two-camera implementation each camera is checked to see if it must connect or disconnect, based on its position and whether it is already connected. Each camera has a boolean variable stored on the AGV that defines the connection status. Connecting to an overhead camera is fairly taxing on the processor and the procedure can take up to 0.35 seconds; however, the average time spent dealing with cameras during a control loop is less than 1 ms and the photo thread (the main bottleneck for processing) continues parallel to connecting, so the overall system response is not compromised.

8.4 Larger Grid Expansion

The multi-camera system is setup to be expandable from the two-camera system that has been tested. By laying out a grid and placing cameras to fit, a well organised system can be created. Figure 8.10 shows how a nine-camera grid could be laid out. Markers would be placed at the overlaps between four cameras to increase efficiency. Each camera would be directly related to the global coordinate system through a single transform.

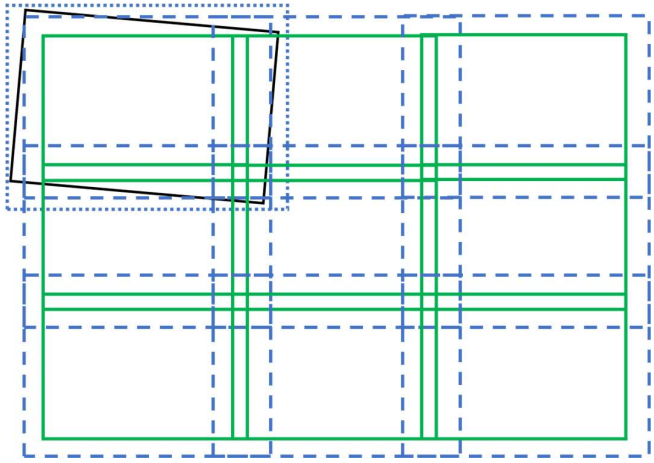


Figure 8.10 Nine camera grid pattern

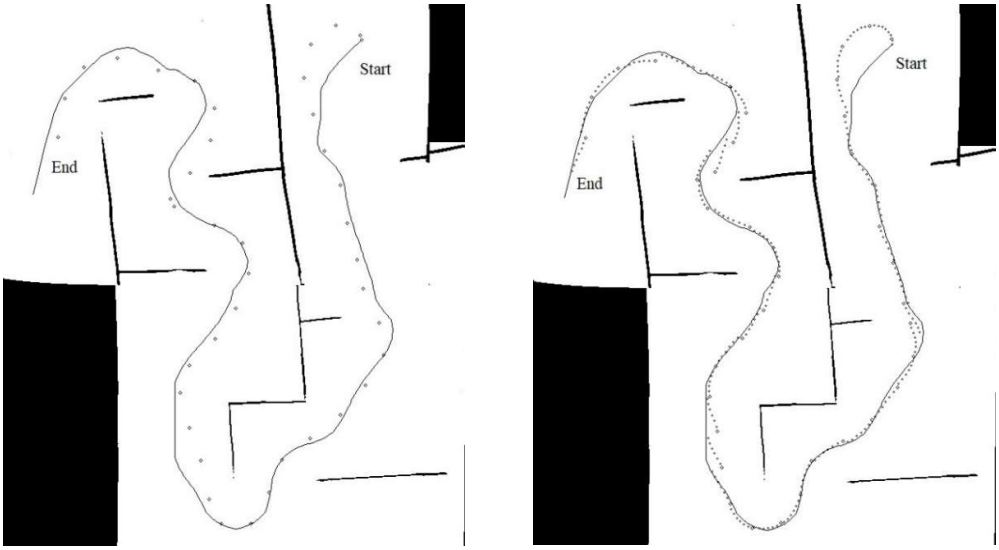
In future work it would be important to put algorithms in place that update how the AGV selects which camera to receive images from. An AGV in the top right zone does not need to check to see if it needs to open a connection with the bottom left zone and so on. Algorithms that consider the next steps of the AGV in the planned path could be even more efficient. Knowing the future trajectory of the AGV could help make intelligent decisions with regards to connections and camera selection, particularly at the corner points where the AGV could be within the hold-on range of four cameras.

9 Evaluation of Two-Camera Navigation

This section evaluates the results gathered during the testing of the updated system in a two-camera setup. The transformation for a specific arrangement of the cameras is calculated by executing two-camera calibration. Thereafter, the test procedure is similar to that detailed in section 7.1. The same results are recorded with the addition of camera connection times.

9.1 Proof of Functionality

The results shown in Figure 9.1 demonstrate the complete system that was designed in this project. The AGV can successfully navigate a two-camera setup with delayed feedback without hitting any obstacles. This test demonstrates how the updated system achieves all three objectives set out in section 1.2. The system achieves the first objective when it uses the forward and position estimators to update delayed camera feedback and turn it into estimated positional feedback that can be used to navigate successfully. The camera feedback and the estimated feedback are depicted in Figure 9.1 (a) and Figure 9.1 (b) respectively. The system completes the second objective when it locates the AGV and destination, plans a path between them and then navigates from camera one, through camera two, and back into camera one, in the process successfully connecting and disconnecting from both cameras. The third objective is met when the system uses the updated path following algorithm, pure pursuit, to follow the planned path through both cameras using transformed coordinates when in view of the second camera, despite starting off at the wrong orientation.



(a) (b)

Figure 9.1 Results of the AGV following a path in a two-camera system
(a) camera feedback (b) estimated feedback

Results of normal testing without a feedback delay and with the AGV facing in the correct direction at the start are given in Table 9.1. The AGV's maximum distance from the path is larger than the maximum distance in the single camera tests. Nevertheless, the AGV did not run into any obstacles in any of the sampled tests and the deviation is less than the width of the AGV. More images were used in these tests as the course is longer. While the number of images varied the system updated consistently in all four tests, updating the frequencies sent to the motors between 345 and 356 times.

Table 9.1 Results of two camera testing

Test No.	Movement Time	Max distance from path	No. Images	No. Updates	Max camera connection time
1	52.5 sec	19.9 px	212	352	0.35 sec
2	51.1 sec	18.5 px	201	347	0.02 sec
3	52.8 sec	15.7 px	196	356	0.31 sec
4	52.6 sec	20.8 px	185	345	0.31 sec

9.2 Connection Demonstration

The updated system can locate the AGV and the destination in a two-camera workspace, plan a path between the two and follow that path to a destination in either camera. To move throughout the workspace the AGV connects to cameras as required. Figure 9.2 shows the connections made by the AGV as it moves through the workspace.

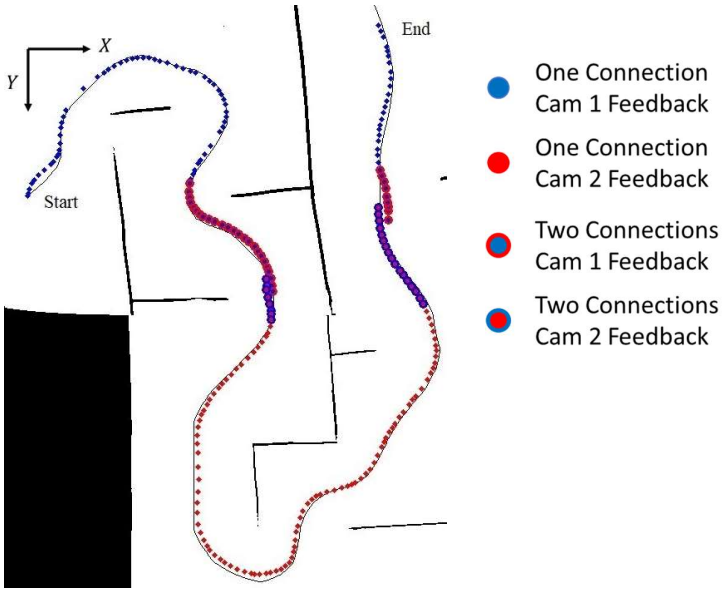


Figure 9.2 Demonstration of connections during two camera test

The influence of the hold-on range can be seen in Figure 9.2 at the changeovers between cameras. As can be seen in Figure 8.9, the top of camera two's hold-on range is higher than the bottom of camera one's hold-on range. Therefore, the first changeover happens lower on the Figure 9.2 than the second changeover. This difference is caused by AGV holding onto the first camera to the bottom of the first camera's hold-on range and then holding onto the second camera to the top of the second camera's hold-on range. The figure also shows how the system first connects to the camera and then only requests information from it at a later stage.

The maximum time taken to connect to a camera for each test is shown in Table 9.1. Because the connection function is in the main flow of the program the system cannot update other functions while it is connecting so the maximum time (in all the tests) of 0.35 seconds is a significant delay. The system is still able to function because this delay only occurs at the crossover between cameras. The average time taken during normal movement is approximately 1 ms which is not an issue. The delay while dealing with cameras is unavoidable but did not cause issues during testing because they happened so rarely. However, it is important that the system does not jump between connections so that this delay does not become an issue. As can be seen in Figure 9.2 the ranges defined in section 8.3.2 function as designed and the system never connects and immediately disconnects from a camera.

9.3 Path Planning

The updated two-camera system uses the same path planning algorithm as the inherited system. An important component during map generation was to ensure that the inherited system's path planning algorithm would be able to be used unchanged. Results of testing the path planning system are shown in Appendix C.

Path planning on a map generated for two-cameras took between 10 and 85 seconds to execute, depending on the start and end points. In the inherited system (a single camera system), path planning took from 5 to 30 seconds (Brits, 2016). The two-camera system takes considerably longer to plan a path although this increase is expected. The number of nodes in a larger map increases to approximately double and therefore the path planning algorithm requires more time to execute.

9.4 Transformation

To measure the accuracy of the transform, a marker was placed in the shared view of both cameras and located by both cameras with the feedback from the second camera being transformed to camera one's coordinate system (the global coordinate system). The real position of the marker is constant so the two sets of coordinates should be the same. However, the transform is not perfectly accurate so there will be a difference between the position given in camera one's feedback and the position transformed from camera two's feedback. During testing it was found that the height of the marker in the images used for two-camera calibration influenced

the transform accuracy. For these tests, two-camera calibration was executed with the marker placed on top of the AGV. Table 9.2 shows results of tests where the marker was on the floor as well as on the AGV.

Table 9.2 Transform accuracy results

Test no.	Difference between real and transform	Distance from calibration point	Level
1	5.0 px	3.3 px	AGV Height
2	7.9 px	85.8 px	AGV Height
3	8.8 px	90.8 px	AGV Height
4	21.3 px	27.6 px	Floor
5	19.5 px	113.0 px	Floor

Results of tests one to three show that the transform is relatively accurate when the height of the objects correlates with the height at which the system was calibrated. The maximum difference in this case was 8.8 px. The difference tended to increase as the marker moved further from the original calibration point. When the marker was put on the floor, the differences increased. This increase shows the deficiencies in the method of calibration as the height at which the calibration was performed affected the results. As the location of the AGV was deemed to be most important, all other tests in this chapter were calibrated with the marker raised. The effect of this calibration can be seen in Figure 9.3. The map does not overlap perfectly and the horizontal line attached to the top right obstacle is drawn twice. The obstacles are eroded and the two lines become one during map generation.

While the transform is not completely accurate, Figure 9.3 shows that the AGV is able to find the destination in the second camera accurately and without running into any obstacles. The fact that all objects in the second camera, including the obstacles, the destination and the AGV, undergo the same transformation makes any inaccuracies irrelevant once the AGV enters the second camera's view. Issues only occur as the system switches cameras, but the system is able to correct itself without hitting obstacles; as can be seen in Figure 9.3.

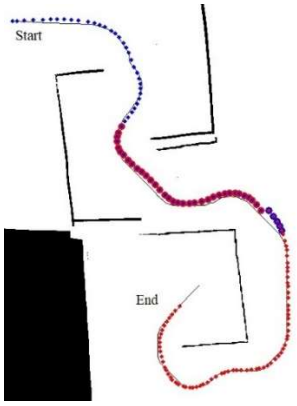


Figure 9.3 Finding a destination in camera two using the transform

10 Conclusion and Recommendations

The project's goal was to determine if onboard computing on an AGV is a viable technique for use in a large static environment, for example, as an alternative to a conveyor belt in a factory. To achieve this goal, the project involved the review and update of a working AGV system. The updates include the addition of a position estimator to provide open loop feedback, the implementation of the pure pursuit algorithm as a new path following method and the expansion of the system to work with two cameras. The updates address issues and avenues for improvement identified during past projects as well as issues that arose during the current project.

The system could only update its movement based on closed loop feedback sent to the AGV over Wi-Fi connection in the form of images from the overhead cameras. The position estimator designed in this project provides open loop positional feedback using the signals sent to the stepper motors and information regarding past locations. The impact of the movement during image processing was observed during this project and as a result, forward estimation was implemented. The forward estimator was an important step in improving the overall response of the updated system, especially when camera feedback was slow. Using the open loop feedback the AGV can update its control signals without receiving information from an overhead camera allowing it to reach the destination with inconsistent camera feedback.

The system has been expanded to use feedback from two cameras. All AGV coordinates received are immediately transformed into global coordinates so that the AGV can function normally in either camera's field of view. Using two cameras enabled the size of the workspace to be increased from approximately 7 m^2 to approximately 13 m^2, which is just less than double the original size, allowing for a small overlap between the cameras' fields of view. During the review of the inherited system, it was decided not to update the inherited system's path planning algorithm. Therefore, in order to navigate the larger two-camera map, the map generation system was designed to work with the inherited algorithm. In the implemented two camera system the AGV was able to reach a destination successfully, anywhere in the field of vision of either camera. The deviation from the path was increased compared with the single camera system due to inaccuracies with the transform; however, the AGV did not hit any obstacles in any of the tests.

The pure pursuit path following algorithm was implemented on the updated system, which resulted in an increase in both speed and accuracy. The new algorithm also caused the system to work better with the position estimator, leading to improved results.

This project has improved the inherited system by delivering an AGV that can navigate at increased speed through a static two-camera workspace with high accuracy under non-ideal feedback conditions.

Recommendations for future projects include expanding the system to work with more than two cameras, developing the calibration system further, improving the path planning algorithm, implementing a Kalman filter and enabling the system to work in dynamic environments.

The two-camera system was designed to be expandable; however, it is untested with more than two cameras. In addition, more advanced calibration techniques should be considered. Better methods of camera calibration could benefit the system by increasing the transformation accuracy, even for a two-camera system. Literature discussed in section 2.7.1 used multiple markers to calibrate a workspace to a global coordinate system leading to increased accuracy.

The increased complexity of larger maps in the two-camera system increased the time taken to plan a path. Therefore, it is reasonable to assume that path planning time will continue to increase as maps get larger. Because the path is planned before the AGV begins to move, the increased time does not impact the movement of the AGV. However, if the system is to be further expanded, it is recommended that future work look to improve the path planning algorithm.

The forward estimator is a form of filter that could be expanded to function with or as a Kalman filter. Kalman filters are used to eliminate noise could be used in this system to receive and filter camera feedback.

Warehouses and factories are not static environments. In order to maximise its use to industry, the AGV must be able to function in a dynamic environment. Therefore, future iterations of this project should aim to enable the system to function in dynamic environments.

11 List of References

AGV Technology, 2019. *Types of AGV Navigation Systems - Automated Guided Vehicles.* [Online]
Available at: http://www.agvnetwork.com/types-of-navigation-systems-automated-guided-vehicles#laser-navigation-lgv
[Accessed 21 October 2019].

Anderson, J., 2019. *An Intro to Threading in Python.* [Online]
Available at: https://realpython.com/intro-to-python-threading/
[Accessed 27 September 2019].

Benevides, C., 2018. *The Advantages and Disadvantages of Automated Guided Vehicles (AGVs).* [Online]
Available at: https://www.conveyco.com/advantages-disadvantages-automated-guided-vehicles-agvs/
[Accessed 24 October 2019].

Black, J. & Ellis, T., 2005. *Multi Camera Image Tracking,* Surrey: Elsevier.

Brits, S., 2016. *Development of a self-navigating AGV using an overhead camera,* Stellenbosch: Stellenbosch University.

Duchon, F., Babineca, A. & Kajan, M., 2014. Path planning with modified A star algorithm for a mobile robot. In: F. Trebuňa, ed. *Procedia Engineering.* s.l.:Elsevier, pp. 59-69.

Heater, B., 2019. *Amazon debuts a pair of new warehouse robots.* [Online]
Available at: https://techcrunch.com/2019/06/05/amazon-debuts-a-pair-of-new-warehouse-robots/
[Accessed 26 September 2019].

Lucas, G., 2006. *A Path Following a Circular Arc.* [Online]
Available at: http://rossum.sourceforge.net/papers/CalculationsForRobotics/CirclePath.htm
[Accessed 24 October 2019].

Lundgren, M., 2003. *Path Tracking for a Miniature Robot,* Sweden: Department of Computing Science, Umeå University.

Martindale, J., 2019. *What is Wi-Fi? Here's everything you need to know.* [Online]
Available at: https://www.digitaltrends.com/computing/what-is-wi-fi/
[Accessed 25 September 2019].

MHI, 2012. *What are AGVs used for?.* [Online]
Available at: http://www.mhi.org/downloads/industrygroups/agvs/elessons/AGVS-On-Line-Training-Module-2-FinalT.pdf
[Accessed 24 October 2019].

Mordvintsev, A., 2013. *Contours Hierarchy.* [Online]
Available at: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_contours/py_contours_hierarchy/py_contours_hierarchy.html
[Accessed 25 September 2019].

Mordvintsev, A., 2013. *Morphological Transformation.* [Online]
Available at: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html
[Accessed 25 September 2019].

Olsen, B. D. & Hoover, A., 2000. *Calibrating a camera network using a domino grid,* San Diego: Pergamon.

OpenCV Development Team, 2019. *Basic Thresholding Operations.* [Online]
Available at: https://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html
[Accessed 24 September 2019].

Reynolds, C., 2015. *Steering Behaviors: Path Following - The Nature of Code.* [Online]
Available at: https://www.youtube.com/watch?v=2qGsBClh3hE&list=PLRqwX-

V7Uu6YHt0dtyf4uiw8tKOxQLvlW&index=6
[Accessed 16 October 2019].
ROCLA, 2018. *Save Money with Automated Guided Vehicles.* [Online]
Available at: https://www.rocla-agv.com/en/agvs-save-money/save-money-automated-guided-vehicles
[Accessed 24 Oct 2019].
Rouse, M., 2014. *TCP (Transmission Control Protocol).* [Online]
Available at: https://searchnetworking.techtarget.com/definition/TCP
[Accessed 25 September 2019].
Williams, A., 2016. *AGVs find their way.* [Online]
Available at: https://www.automotivemanufacturingsolutions.com/agvs-find-their-way/35016.article
[Accessed 24 October 2019].
Wilson, S., 2017. *Development of a self-navigating AGV using multiple overhead cameras,* Stellenbosch: Stellenbosch University.
Yahja, A., Singh, S. & Stentz, A., 2000. *An Efficient On-line Path Planner for Outdoor Mobile,* Pittsburgh, PA: Robotics Institute, Carnegie Mellon University.

Appendix A Techno-Economic Analysis

This techno-economic analysis covers the project budget, time management, technical impact, return on investment and potential for commercialisation.

A.1 Budget

The proposed budget for this project was R 166 350. This amount included engineering time, running costs, use of facilities, capital costs, material costs and workshop time and labour. The amount spent on this project was R 170 129, which is R 3 779 over budget. Table A.2 and Table A.3 show the breakdown of project costs.

The total cost of the project was reduced as no time was required in the workshop or no materials were purchased. The budgeted running costs were R 1 750. These funds were allocated to fix any problems with the inherited hardware system and to purchase components for the two-camera system. The actual running costs were R2 429, which was R 679 over budget. This is due to the purchase of an extra Raspberry Pi to replace one of the inherited ones that broke during system development. Table A.1 shows expenditure on parts during the project.

Table A.1: Cost of Components

Item	Unit Cost	No.	Total Cost	Store
Jumpers Female	R79.00	1	R79.00	Micro Robotics
Heat Shrink Kit 127	R48.95	1	R48.95	Micro Robotics
Stepper Controller Up to 2A	R117.00	2	R234.00	Micro Robotics
Velcro Stick on Tape	R120.00	1	R120.00	Builders Express
Insulation Tape	R10.00	1	R10.00	Builders Express
Duracell AA Battery Pack 12	R149.00	1	R149.00	Builders Express
Raspberry Pi 3B+	R552.00	1	R552.00	Communica
Raspberry Pi 3B+	R552.00	1	R552.00	Communica
Duracell AA Battery Pack 12	R150.00	1	R150.00	Game
Second Camera Module	R184.00	1	R184.00	Micro Robotics
Raspberry Pi 3B+ Case	R110.00	1	R110.00	Communica
Duracell AA Battery Pack 12	R120.00	1	R120.00	Checkers
Duracell AA Battery Pack 12	R120.00	1	R120.00	Checkers
Total			**R2 428.95**	

Table A.2 Proposed budget allocation

Activity	Engineering Time		Running Costs	Facility Use	Capital Costs	MMW Time		MMW Material	Total
	hr	R	R	R	R	hr	R	R	R
Literature study	35	12250	100	0	0	0	0	200	12550
Past Design Review and Understanding	25	8750	100	0	0	0	0	0	8850
Position Estimator Development	50	17500	100	250	0	0	0	0	17850
Position Estimator Implementation and Testing	35	12250	100	175	0	0	0	0	12525
Forward Estimation	0	0		0	0	0	0	0	0
Progress Report	35	12250	0	0	0	0	0	0	12250
Two-camera System Development	70	24500	700	350	0	0	0	0	25550
Path Following Algorithm Update	25	8750	100	125	0	0	0	0	8975
Two-camera System Implementation	70	24500	0	350	0	0	0	0	24850
Initial Full System Tests	40	14000	400	200	0	0	0	0	14600
First Draft Report	50	17500	0	0	0	0	0	200	17700
Final Full System Tests	0	0	150	0	0	0	0	0	150
Final Report	30	10500	0	0	0	0	0	0	10500
Totals	**465**	**162750**	**1750**	**1450**	**0**	**0**	**0**	**400**	**166350**

Table A.3 Final budget allocation

Activity	Engineering Time		Running Costs	Facility Use	Capital Costs	MMW Time		MMW Material	Total
	hr	R	R	R	R	hr	R	R	R
Literature study	35	12250	0	0	0	0	0	0	12250
Past Design Review and Understanding	40	14000	492	0	0	0	0	0	14492
Position Estimator Development	35	12250	701	175	0	0	0	0	13126
Position Estimator Implementation and Testing	40	14000	0	200	0	0	0	0	14200
Forward Estimation	30	10500	0	150	0	0	0	0	10650
Progress Report	10	3500	0	0	0	0	0	0	3500
Two-camera System Development	55	19250	846	275	0	0	0	0	20371
Path Following Algorithm Update	25	8750	120	125	0	0	0	0	8995
Two-camera System Implementation	50	17500	0	250	0	0	0	0	17750
Initial Full System Tests	20	7000	150	100	0	0	0	0	7250
First Draft Report	55	19250	0	0	0	0	0	0	19250
Final Full System Tests	35	12250	120	175	0	0	0	0	12545
Final Report	45	15750	0	0	0	0	0	0	15750
Totals	**475**	**166250**	**2429**	**1450**	**0**	**0**	**0**	**0**	**170129**

A.2 Time Management and Gantt Chart

Figure A.1 shows the Gantt chart that tracks the project progress over its duration and compares it with the planned timeline in the project proposal. Activities completed during the project are shown in the chart and compared to the baseline. The review of the past design and position estimator development took longer than expected. The requirement for the development of a forward estimator also delayed the project. The two-camera system progressed faster than expected and time was recovered at that stage. All milestones were met during the project. The project was completed and the report was handed in before the deadline on 25 October 2019.

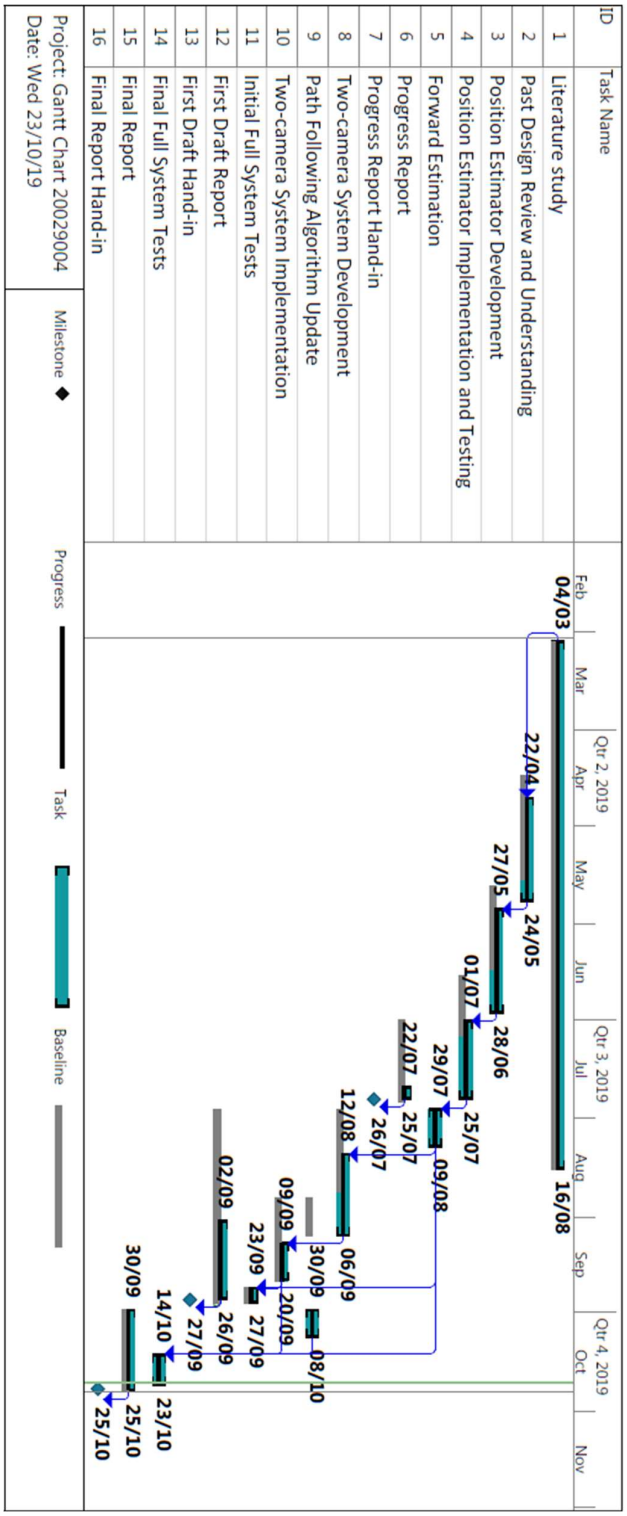


Figure A.1: Project Gantt chart

A.3 Technical Impact

The project shows that it is feasible to use an AGV with onboard processing in a static environment with multiple overhead cameras. The project provides information to future projects which can use this design to expand the system. Improvement opportunities include larger systems, multiple AGVs and new calibration methods. The overhead camera system can also be used to further investigate different camera calibration techniques and the impact they would have on this system or a similar one. The position estimator can also be improved to be able to be used for AGVs of different sizes. Dynamic systems were not investigated in this project, but this design can be used as a base to explore different dynamic implementations.

A.4 Return on Investment

As stated in section A.1, R 170 129 was spent on this project. In the short-term this project can be used as a basis for further work with AGV's in both static and dynamic environments. The position estimator that has been designed in this project can be adapted to work with most AGVs. In the long-term, AGVs can reduce the reliance of factories on human labour. Return on investment for the implementation of an AGV system needs to take into account the direct costs of the AGV system, the increase in profit from implementation, the decrease in direct costs from operator salaries, the decrease in indirect costs of health and safety compliance, damages from human error and negligence and the cost of employee claims and injuries. AGV systems can quickly pay off the initial costs, in some cases taking less than two years (ROCLA, 2018).

A.5 Potential for Commercialisation

For this AGV system to be commercialised there would need to be further investigation. The motivation behind this project was to develop an AGV that could be used as a substitute for conveyor belts in a factory. The system to date has focused entirely on controlling the movement of the AGV. Further steps are necessary for the system to function in a factory or warehouse. The current system only works for a single AGV moving around a clearly defined area where thresholding can be used. To be commercialised the system would need to be expanded to include multiple AGVs that could work in any environment.

Appendix B Risk Analysis and Safety Report

A large amount of the work during this project was conducted in the automation laboratory. The risks involved in working in the laboratory setup were identified and assessed in a safety report to ensure that all guidelines were followed. The laboratory risk assessment is shown below.

B.1 Safety Report Summary

Table B.1 Safety report details

Location of setup	Automation Laboratory
Project Title	Development of a self-navigating AGV for use with multiple overhead cameras
Student Details:	
Name	Sam Jeffery
Student No.	20029004
Contact No.	083 441 7399
Supervisor	Dr K. Kruger
Laboratory Engineer	Mr R Rodriguez

B.2 Overview of Work Performed

The operation of an automated guided vehicle (AGV) for Mechatronics Project 478 is to be performed. The process will include the programming and fixing of a previous build. Before any programming or construction can commence a thorough investigation regarding risks and safety must be performed. This will provide a basis for the level of safety that has to be applied when conducting any work in the mechatronic labs.

The AGV, shown in Figure B.1, is designed to navigate around obstacles in an enclosed space using feedback from an overhead camera. The specific task of this project is to implement a position estimator on the current structure in order to improve robustness.

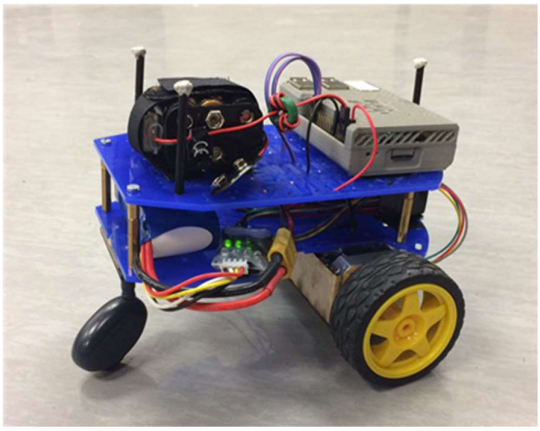


Figure B.1 AGV used for development and testing

The components and equipment used to be used are detailed in Table B.2 and Table B.3 respectively:

Table B.2 Components used in automation laboratory

Component	**Sub-System**
2x Stepper Motor with Drivers	AGV Movement
Chassis	AGV Structure
Raspberry Pi	Controller
AA Battery Module	Power the Raspberry Pi Module
Rechargeable LiPo Battery Module	Power the stepper motors

Table B.3 Equipment used in automation laboratory

Equipment used:	
Lab bench power supply	Hacksaw
Wire cutter	Soldering iron
Ladder	

B.3 General Lab Safety

The following general lab safety instructions are applicable:

- No afterhours testing may be performed without the necessary permissions.
- An induction is required before testing may be undertaking.

- Access to the labs are only allowed after attaining approvals and signatures from the project facilitator, lab engineer and lab manager.
- Closed shoes must always be worn.
- Loose clothing may not be worn.
- Loose hair should be tied back.
- Good housekeeping practised should be kept during testing.
- No food or drink is permitted in the lab.
- PPE should be worn at all time when conducting tests in the lab.
- Observe emergency exit and fire extinguisher locations.
- Take care when working with the PLCs. There is a power supply which has 220VAC running to it (mounted on the far right of the tutorbox).
- Avoid touching the power supply and make sure jewellery etc. is not hanging near it.
- Make sure the power is off and unplugged before changing fuses.
- Double check your connections.
- Beware of all moving parts. Keep all body parts away from moving parts.
- Safety glasses should always be worn when working with tools.
- Alert other lab users before operating any powered tools.
- Safety report must be visible and accessible during testing.
- If uncertain, ask for help – it will be willingly provided!

B.4 Activity Based Risk Assessment

General

The Table B.4 provides a list of activities that are to be performed every time when entering or exiting the automation labs at all times through the programming and construction of the system.

Before using any equipment present in the lab one should ensure that they have sufficient knowledge as to the location of the emergency stops and warning

indicators. The lab manager should also do a check on all equipment to ensure its safe for further operation.

Table B.4 General safety procedures

Activity	Risk	Risk Type	Mitigating steps
Entering the lab	Hand injuries from gate	P	Be careful not to get your hand stuck in the security gate.
Turning equipment on and off	Electrical shock	P	Briefly check over cables to ensure that insulation is intact
Working with the bench power supply	Electrical shock	P	Make sure that the bench output is toggled off when making any connections.
Moving around lab	Tripping	P/E	Be aware of your surroundings. Do not trip over cables or surrounding equipment.
	Using ladder	P	Ensure ladder is firmly secured and will not fall under any circumstances Ensure no objects are near the base of the ladder
Working with data	Losing data	P	Store data as per guidelines and ensure that sufficient independent backups are kept.
Tidying lab	Tripping	P/E	Be cautious and constantly clear cables so they are not in the way.
	Cuts	P	Beware of sharp edges on samples that are to be discarded
Locking lab	Hand injuries from gate	P	Do not get your hand caught in the security gate.
Lab Code	Loss of equipment due to unauthorised access people entering the lab	E	Adhere to arrangements for code control. Do no distribute code.
Working Alone	Personal injury/ risk without help	P	Ensure that there is always someone aware of the situation. Always carry a cell phone.
Use, recharge and storage of LiPo rechargeable batteries	Fire from overcharging damaging equipment and endangering people.	P/E	Ensure batteries are never left charging overnight or out of sight. Stop charging batteries as soon as the charged light goes on.
	Performance loss due to discharge at too low voltage	E	Confirm battery output is in the correct range when it's being used. Regularly check individual voltages with a multi-meter.

	Shocking danger during use	P	Use care when connecting external wires connected to the battery, especially when all batteries are in series.
	Loss of performance or short circuit during storage	E	Remove one battery from chain to break loop when not in use.

Updates to build

In order to improve the design of the past project the design and build will need to be updated. All updates will be thoroughly planned and executed with safety being of the utmost importance. Table B.5 lists the safety procedures to be followed during development.

Table B.5 Safety procedures for updates

Activity	Risk	Risk Type	Mitigating steps
Fastening	Getting skin caught between two mating surfaces	P	Ensure that nobody has any body parts between two gaps that will be fastened together.
Soldering Wires	Burning your skin	P	Ensure that no hot solder can drip on to your skin. Ensure that the solder has properly cooled down before touching it. Ensure soldering iron is turned off when not in use.
	Damaging components	E	Ensure that the soldering iron is set to a reasonably low temperature (<320°C)
Writing code	Damaging components	E	Ensure that no code could force the components to operate beyond their rated specifications

Testing

During this phase the system will be tuned and tested for functionality, accuracy and reliability. The following section will provide a list of activities that is to be performed during the testing phase of the system.

Before using any equipment present in the lab one should ensure that they have sufficient knowledge as to the location of the emergency stops and warning indicators. The lab manager should also do a check on all equipment to ensure its safe for further operation. Table B.6 lists the safety procedures to be followed during testing.

Table B.6 Safety procedure for testing

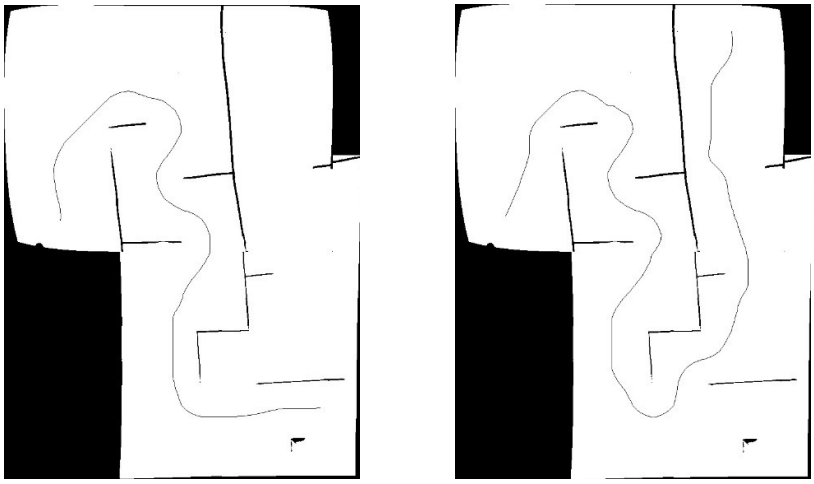
Activity	Risk	Risk Type	Mitigating steps
Motors	Injuring body parts	P	Ensure all moving parts are enclosed. Keep all body parts and clothing away from motor output shaft.
	Burning out the motor	E	Allow the motor to cool down and ensure that the motor is not ran above its rated conditions. Emergency stop button in constantly reachable location.
	Electrical Shock	P	Ensure that all wire are properly insulated, labelled and tied down.
Adjustments to moving parts	Injuring body parts	P	Ensure that the supply to all moving parts are cut before making any adjustment by hand.
Overhead camera placing and adjustment	Injury body parts	P	Ensure that the ladder is firmly secured. Ensure that one always has a free hand to catch one's self.
	Breaking component	E	Take care not to drop component during placement or adjustment. Ensure that the device is securely fastened into place.

Appendix C Path Planning Test

In order to test the path planning algorithm with larger maps, a map was selected, and paths were planned between randomly selected points. Table C.1 shows the time taken to plan each path in a sample of eight tests. Figure C.1 shows examples of two of the paths planned during testing.

Table C.1 Results of path planning test

Test no.	Time to plan path
1	73.4
2	35.4
3	11.6
4	29.1
5	76.1
6	81.2
7	83.4
8	47.1



(a) (b)

Figure C.1 Examples of planned paths (a) test 2 (b) test 7

Appendix D Pixel Conversions

To use the position estimator, real world distances are converted to pixel values in order to work with the feedback provided by the overhead cameras. The cropped *chessboard* shown in Figure D.1 is placed in the workspace and used to calibrate the camera. The board has known dimensions of 277 mm x 277 mm. The coordinates of each of the four corners of the chessboard are found in the undistorted image of the workspace shown in Figure D.2.

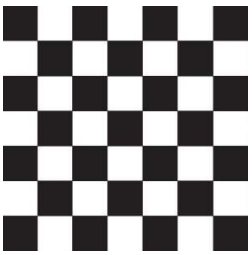


Figure D.1 Cropped chessboard used for calibration

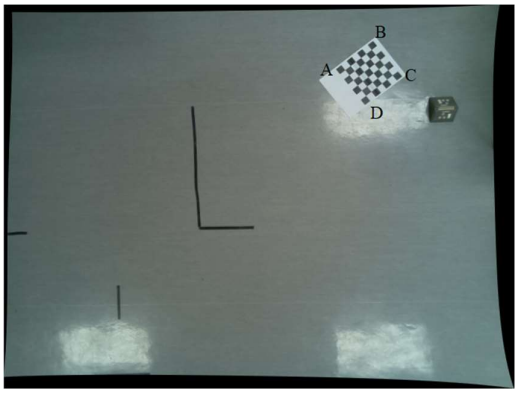


Figure D.2 Undistorted image containing chessboard

The pixel coordinates and lengths of each side are shown in Table D.1. The conversion between pixels and distance was found to be 4.83 mm/pixel.

Table D.1 Coordinate conversions

Point	**A**	**B**	**C**	**D**
Coordinates	(417, 81)	(463, 46)	(499, 90)	(453, 125)
Line	AB	BC	CD	DA
Length (px)	57.8 px	56.85 px	56.85 px	57.80 px
Average	57.33 px			
mm/px	4.83			

Appendix E Creating Images with GIMP

E.1 GIMP Overview

GIMP is a graphics editor that can be used for editing any images. GIMP stands for GNU image manipulation program. The program is free to download and used by designers, photographers and many other professions. It is used in this project to join images accurately. Specific pixels are overlapped and the second image is rotated around that overlap.

E.2 Steps to Join Images

1. Import thresholded and undistorted image from first camera (for example, Figure E.1 (a))
2. Change the canvas size to 1024 x 1024
3. Import a 1024 x 1024 black background
4. Move black layer to bottom
5. Import thresholded and undistorted image from second camera (for example, Figure E.1 (b))
6. Place guides at x1 and y1
7. Move image from second camera to correct location using guides
8. Rotate second image to correct orientation about overlap point
9. Fix overlaps with transparency and edits to create logical map
10. Save as Background0.jpg (for example, Figure E.2)

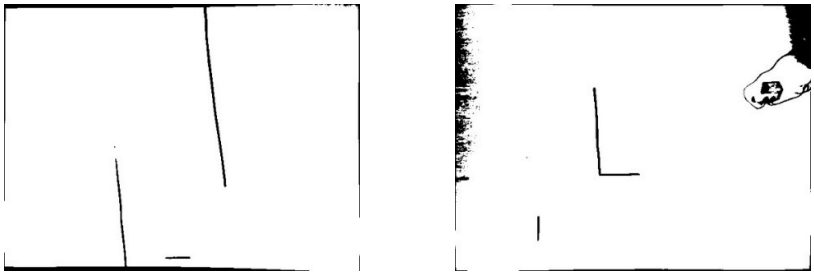


Figure E.1 Thresholded images used to create map, (a) camera one view, (b) camera two view

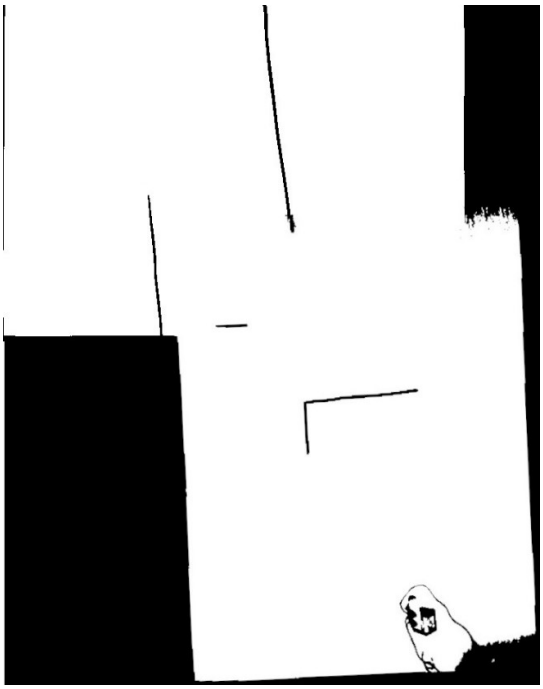


Figure E.2 Example of map generated with GIMP

Appendix F Camera Pass Examples

This appendix gives examples of the image feedback sent to the onboard processor by the overhead cameras.

F.1 Correct Camera Pass

When the system works correctly the camera changes from the first view to the second once and then does not return to the first view. Figure F.1 shows the sequential photos as the AGV moves from camera one to camera two.

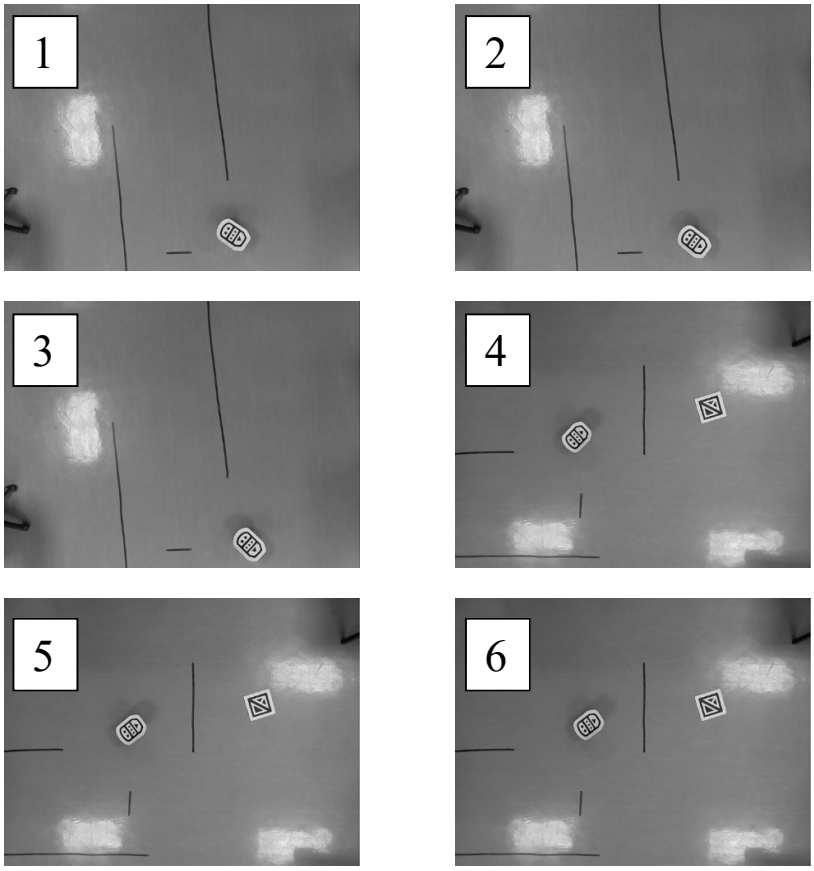


Figure F.1 Correct camera pass using hold-on range

F.2 Incorrect Camera Pass

When the system works incorrectly, the camera changes from the first view to the second view and then returns to the first view after receiving feedback from the second view. Figure F.2 shows the sequential photos as the AGV moves from camera one to camera two and then back to camera one.

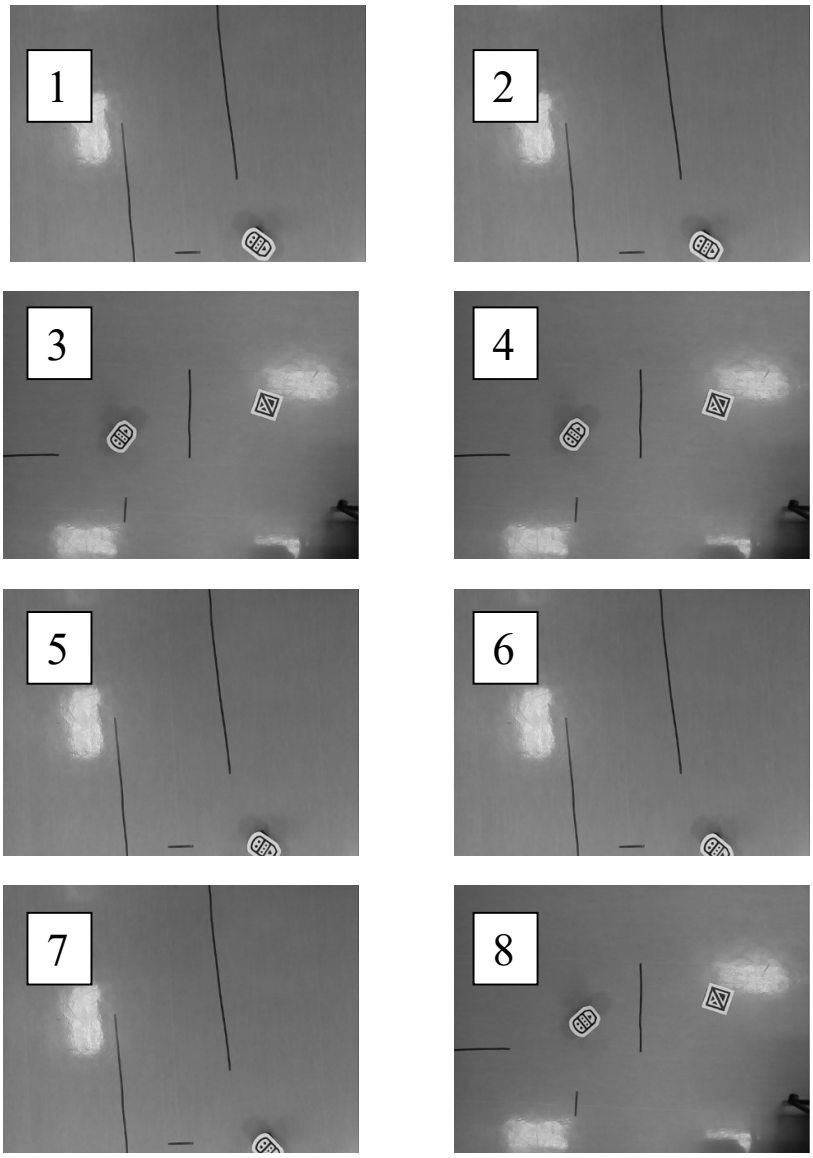


Figure F.2 Incorrect camera pass without hold-on range