COURSWORK 2 PROJECT REPORT

Shan Devraj – ed19sehdMathew Fuller – el18mfLeonardo Dashi – sc20ldAmer Al-Hashimi – sc20aaaa

Mathew Fuller

Table of Contents

Introduction:

The brief of this project is to develop an intelligent system in the scenario of a “robot detective”. The project aims to develop a system which is responsible for controlling a TurtleBot which is placed in a virtual world. It will be presented with 2 door ways, one with a green circle outside of the entrance and one with a red circle outside of the entrance. Both of these rooms will contain an A4 sheet of paper with the image of 1 of 4 Cluedo characters. The correct character to identify is in the green room. Therefore, the TurtleBot must identify the room with a green circle and navigate its way inside of this particular room.

Once the TurtleBot is in the green room, it must locate the image of the Cluedo character and cross check this image with the images of the characters that are saved and thus identify the correct character.

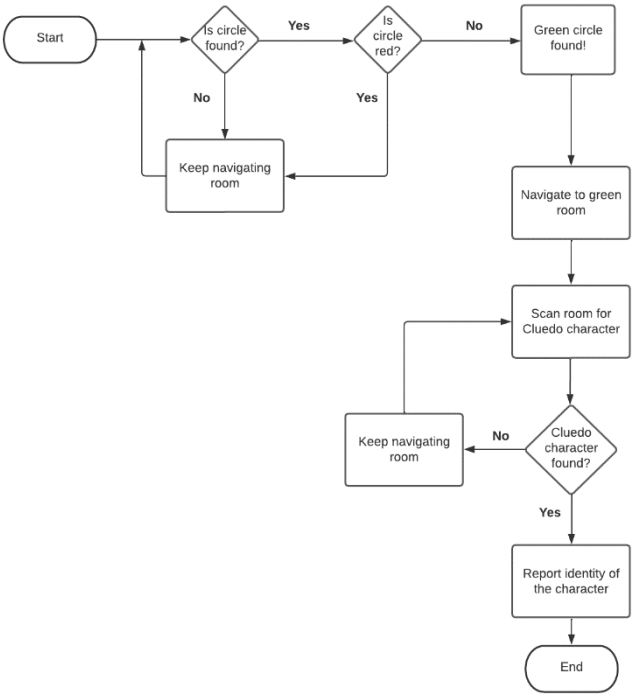
This is the basic premise of the task, however the system must be able to accurately perform in a range of different virtual worlds. Obstacles may be present throughout these worlds, and the TurtleBot should be able to overcome these difficulties.

Design:

The problem can be best broken down into a flow diagram to illustrate the scenario:

In order to tackle the problem, we chose to break the problem down into 3 main sub-sections. These sub-sections are:

1. Circle Detection
2. Image Detection and Matching
3. Autonomous Movement



Due to mitigating circumstances and external pressures, as a group we decided that it would be most efficient to tackle each problem individually as a group rather than working on different problems at the same time and implement them together at the end. Many members were at home during the Easter break and thus it was optimal to pair and group program sections together. In respect to this, the first task we tackled was detecting red and green circles.

Room Detection:

For room detection, it was decided to make use of more simpler methods, that would be able to be used both for the circle detection and cluedo detection, as the nature of this task is simple and the targets set in stone, it would not require adaptive or flexible facial detection methods as the actual targets are not in flux, instead just the scale, lighting and rotation of these targets occur. Firstly, we utilised the TF library to determine the exact location of the robot with respects to the world. This information was the first set of data needed to navigate the robot. Next, we were able to calculate the distance between the Turtlebot's location and the locations of both doorways. The robot would then head towards the doorway which was the closest in order to save time. Once present at the doorway, the Turtlebot would rotate on the spot to search its environment and detect a

green or red circle. If the circle identified was green, it would head into the centre of that room. However, if the circle was red, it would head into the centre of the room in the other doorway as we can accurately assume this is the target room.

Circle Detection

In order to accurately and robustly detect red and green circles there were a number of approaches that we could take. The way we tackled this is to first detect if the frame was either green or not.

When we first tried to implement a colour match to check whether the frame was either green or red, we realised that the colour was not the perfect green or red. For example, the green circle had shades of blue and thus the system did not recognise it as green.

With robustness in mind, we had to ensure that the system could detect all shades of green to deal with similar issues. Therefore, for each red and green, we set upper and lower boundaries for the colour hue, saturation, and value (HSV format). This describes colours in terms of their tint, their shade, and their brightness and if a colour was to fall in between these ranges, we can confidently confirm that it was the correct colour.

However, as this was not a fool-proof method, with rogue objects in the environment possibly providing false positives if they share a similar colour or shades with near enough HSV values, the need for a redundancy in the form of a shape matching/confirmation method to ensure that the robot confirms that not only is the object in the correct HSV range, but also that it matches to a satisfactory degree, the shape of the circle. This allows us to have a system akin to that of a 2-factor authentication system, such as Duo, to avoid unneeded errors and create a robust yet simple solution. To tackle this, we used Hu Moments which is a powerful technique to detect circular shapes in particular.

Hu Moments are a set of mathematical moments which can be used to describe shapes of objects, and this method was especially useful in our scenario due to the fact that they are invariant to translation, scale and rotation and thus it would work even when the robot was

not in the perfect orientation or position (Bapat et al., 2018). Ordinarily, we would have to normalise the moments to calculate how circular an image is, but as we used Open CV's matchShapes library, it done the normalisation for us. Once matchShapes provides a score which determines how close to a perfect circle the image is, if the score was on or above 85%, we could confidently determine that the shape was indeed a circle.

Character Detection

Once the green circle has been identified, the system is then responsible for identifying the correct Cluedo character. Similar to the circle recognition, the system uses OpenCV's matchShapes function and Hu Moments.

The first step is to obtain the contours of each character that may be detected within the room. In this context, contours refer to the boundaries of the image. The contours of the characters are obtained using the findContours function available from OpenCV. To speed up computation time for the algorithm and ensure that the robot is able to identify the character under the 5-minute time limit, a minimum threshold is set on the contour area which filters out small contours that are unlikely to be a Cluedo character.

Once the contours are obtained, the system needs to ensure that it can detect these images from any angle and distance and thus each contour is iterated over and a Hu Moment value is calculated. As discussed previously, the Hu Moments values are invariant to translation, rotation and scaling and so this increases the robustness for the character identification.

Once the Hu Moments values are calculated, the system then compares them with the Hu Moments values of each of the template character images. The system uses the matchShapes function which makes use of Hu Moments to compare each image and returns a value based on these, which we then compared to a set value that was calculated by matchShapes, that was output by this function when each character was shown to the robot in perfect lighting and framing, converting the value into a percentage match to said number. The template image used to match against the detected template is determined by

the detected colour that is detected by the robot. This was done as it was found to increase the efficiency of the robot instead of comparing each template to the detected one.
A character is considered accurately identified if there is above an 85% match. If the match is between 55%-85%, the character is considered as a potential suspect whilst any lower than this indicates that the system is not certain of a match and continues to search for other suspects. It has the ability to output suspects if the internal timer within the code runs out, making sure that even if a match of 85%+ is not reached, that there is still an output, even if it is not certain.

One of the main benefits of this approach of using Hu Moments is the fact that the system is more resilient to noise and artifacts within the virtual world. Furthermore, using Hu Moments compared to using other image matching techniques such as SIFT has a range of advantages such as increased computation speed. Computing Hu Moments values can be faster than computing SIFT descriptors especially in real-time scenarios. It is also a more robust approach as the matchShapes function compares the shape of the characters rather than analysing pixel intensities and gradients like SIFT.

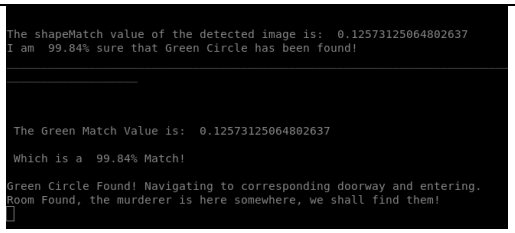
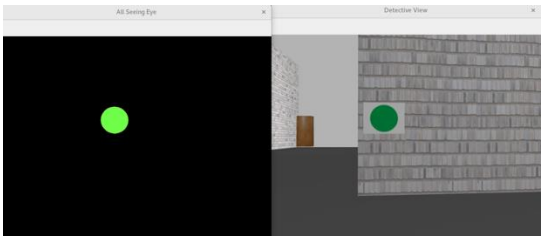
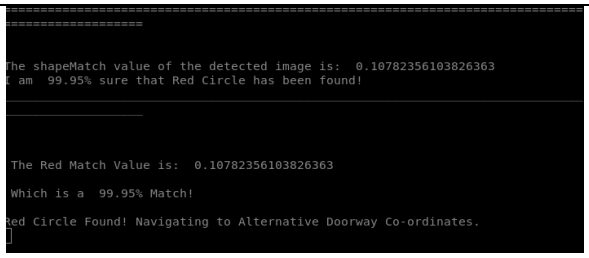
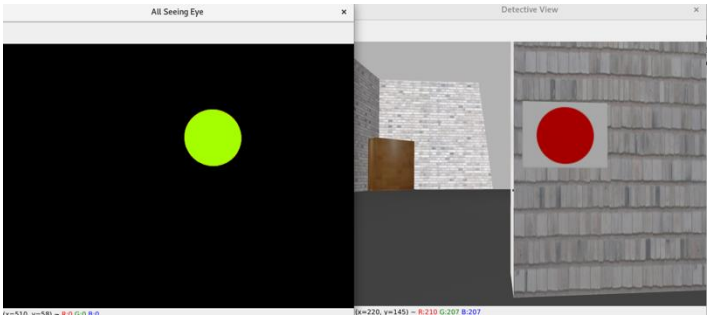
Autonomous Movement

Efficient autonomous movement was unfortunately not achieved however we were able to implement a part working solution which enables the robot to enter the correct room which has a green circle and partially explore the target room. Initially we explored using the A* algorithm and Frontier Exploration but were unsuccessful in implementing these.

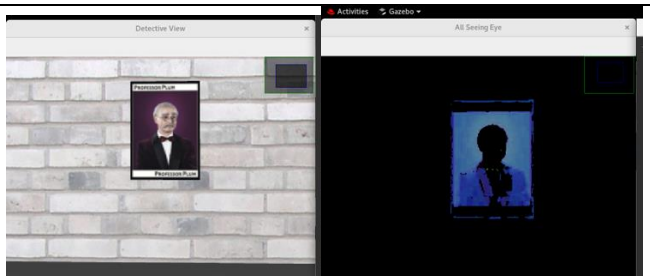
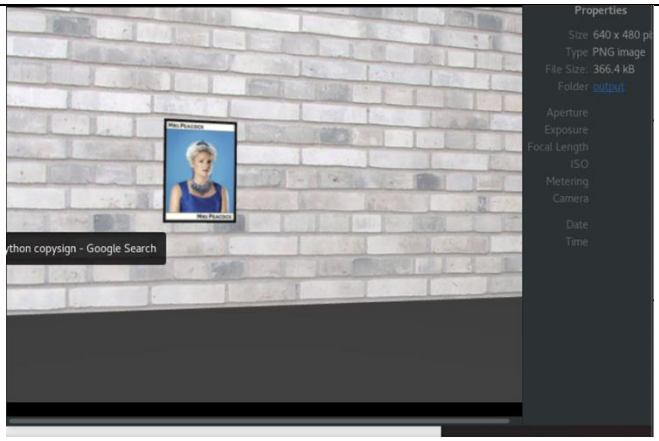
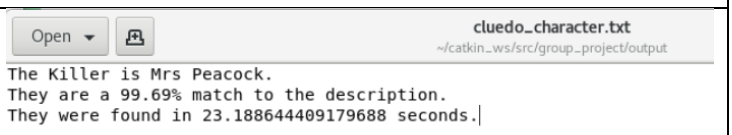
Room Exploration:

Once in the target room, the Turtlebot attempts to move in a spiral motion and thus find the relevant Cluedo character. However, it sometimes displays erratic movement patterns and the spiral motion is not quite as smooth as we would have hoped. This is not ideal in rooms which are not rectangular or ones with significant obstacles as the movement patterns are not optimised for these scenarios.

Implementation and Testing:

Description	Outcome	Evaluation	Evidence
Does the robot successfully identify instances of a green circle and recognise them as green circles?	Successfully identified..	The robot was placed at various distances and angles around the green circle and accurately identified instances of green circles above the threshold in all scenarios.	 
Does the robot successfully identify instances of a red circle and recognise them as red circles?	Successfully identified.	The robot was placed at various distances and angles around the red circle and accurately identified instances of red circles above the	 

		threshold in all scenarios.	
Does the robot successfully identify the Cluedo character Colonel Mustard?	Successfully identified.	In all angles and instances, the robot was able to successfully identify Colonel Mustard.	
Does the robot successfully identify the Cluedo character Miss Scarlett?	Successfully identified.	In all angles and instances, the robot was able to successfully identify Miss Scarlett.	
Does the robot successfully identify the Cluedo character Mrs Peacock?	Successfully identified.	In all angles and instances, the robot was able to successfully identify Mrs Peacock.	

Does the robot successfully identify the Cluedo character Professor Plum?	Successfully identified.	In all angles and instances, the robot was able to successfully identify Professor Plum.	
Does the robot successfully take a photo of the Cluedo characters?	Successfully took a photo.	In all angles and instances, if the robot was able to successfully identify the Cluedo character, it was able to take a photo of it.	
Does the robot successfully write to a file if a Cluedo character is identified?	Successfully wrote to file.	When a Cluedo character was identified to a suitable accuracy, the system was able to	

		write to the file.	
Does the robot autonomously move to find the green circle?	Failed to move autonomously.	The robot was unable to move around its environment and search for a green circle.	No evidence.
Does the robot autonomously move to find Cluedo character?	Failed to move autonomously	The robot was unable to move around its environment and search for a Cluedo character.	No evidence.

Final Testing with combination of autonomous movement and detection systems.
During this testing, code was included in the files to create files at each detection and confirmation stage, including time stamps.

Over 20 automatic tests were conducted by Mathew, with the average time passed since the beginning of the test at each stage of each world as follows:

World 1

- Arrival at nearest door x,y co-ordinates to the robot's starting point
    - Number of successful runs out of 20 - 20
    - Minimum time taken - 10.54 seconds.
    - Maximum time taken - 34.12 seconds.
    - Average of 20 tests - 19.43 seconds
- Detection and confirmation of circle (as these happened simultaneously on each run)
    - Number of successful runs out of 20 - 20
    - Minimum time taken – 35.13 seconds
    - Maximum time taken – 129.16 seconds
    - Average of 20 tests – 71.33 seconds
- Arrival in correct room centre
    - Number of successful runs out of 20 - 20
    - Minimum time taken – 63.19 seconds
    - Maximum time taken – 162.55 seconds
    - Average of 20 tests – 119.34 seconds
- Detection of Cluedo Character
    - Number of successful runs out of 20 - 7
    - Minimum time taken – 193.84 seconds
    - Maximum time taken with successful detection - 243.98 seconds
    - Maximum time taken – Timer ran out/program closed before it could create the file
    - Average of successful runs – 219.54 seconds

World 2

- Arrival at nearest door x,y co-ordinates to the robot's starting point
    - Number of successful runs out of 20 - 20
    - Minimum time taken - 7.09 seconds.
    - Maximum time taken - 28.89 seconds.
    - Average of 20 tests – 24.49 seconds

- Detection and confirmation of circle (as these happened simultaneously on each run)
    - Number of successful runs out of 20 - 20
    - Minimum time taken – 39.13 seconds
    - Maximum time taken – 158.16 seconds
    - Average of 20 tests – 87.13 seconds
- Arrival in correct room centre
    - Number of successful runs out of 20 - 20
    - Minimum time taken –58.43 seconds
    - Maximum time taken – 193.96 seconds
    - Average of 20 tests – 125.86 seconds
- Detection of Cluedo Character
    - Number of successful runs out of 20 - 3
    - Minimum time taken – 215.09 seconds
    - Maximum time taken with successful detection - 237.37 seconds
    - Maximum time taken – Timer ran out/program closed before it could create the file
    - Average of successful runs – 223.77 seconds

To note was that for a single run in world 2, the robot got stuck in a loop after the detection of the first circle, where it would reach the centre of the correct room, pause, then go back to the door co-ordinates of the correct room, then the starting co-ordinates, and restart the main code. This error was not able to be reproduced nor understood as to how this happened, and so the run was not included in the final results.

From these tests, it could be gleaned that the robot performed quite well, and similarly, in the initial movement and detection of the circle and finding it's way to the centre of the correct room, with the only variation being issues with it becoming stuck on the corner of the doorway, with sometimes a second or two of repathing occurring or the robot coming to a complete stop and struggling to figure out how to proceed causing a random time sink to occur. It was only found that the spiral and avoid method implemented, even in it's unfinished state, shows that this solution would work better in rooms with a more symmetrical design, such as square, rectangular or circle rooms, in comparison to rooms with irregular shapes, such as rhombi.

Evaluation:

Strengths:

- The TurtleBot's ability to detect green and red circles is a significant strength. The TurtleBot is able to accurately detect green and red circles in a variety of different angles and distances and uses a simple yet robust method. This allows for faster processing and allows for the robot to not waste significant time in its detection. Furthermore, it has a 2 factor check whereby it checks for the colour ranges to determine that all shades of green and red can be detected whilst also confirming the circularity of the shapes for further robustness.
- The TurtleBot's ability to detect all of the Cluedo characters is also a significant strength. The system is able to detect all characters from a range of different angles and distances. This in turn demonstrates the robustness of the detection for the TurtleBot. Also, the methods used are fairly simple and thus reduces computation time and keeps the robot within the time limit for the scenario.
- The system has an emphasis on robustness and simplicity when dealing with shape and character identification. This allows the TurtleBot to quickly recognise patterns in virtual worlds and allows for the TurtleBot to interact with its environment in an efficient manner.
- The TurtleBot's ability to recognise and identify green and red circles as well as Cluedo characters is a novel approach which has advantages in simplicity, efficiency, robustness as well as being computationally efficient through using matchShapes and Hu Moments.

Weaknesses:

- Obstacle detection - A significant weakness of the TurtleBot is that it does not take into account obstacles or non-rectangular rooms within its movement approach. Using an algorithm such as A* would result in better outcomes.
- Movement sticking points – It was found that the robot tended to get stuck when close to obstacles, especially on doorway corners when moving from the opposite doorways x,y co-ordinates if the door it was at had the red circle, and this was

slightly reduced by setting the initial movement co-ordinates to the other doorways x, y position then to the centre of the room, however it still did occur. This was a significant time since in comparison to the time it took to complete the actions leading up to this, and can definitely be considered the biggest weakness of the functional movement.

- Smooth movement - Once inside the centre of the room, the Turtlebot lacks a smooth spiral exploration. This can lead to inefficient exploration.
- Staggered movement - the robot seems to take a few moments between move_base movement commands, which prevent smooth and quick movement, and this may be due to the sleep length in the GoToPose class, however there was not enough time to investigate this or to attempt to fix it.
- Single file solution - The single file/class solution can be considered a weakness, as it makes any additions or changes to the code hold the possibility of rendering the entirety of the code non-functional. The use of separate files that make the use of multiple classes to cleaning call each function required to solve the components of the task would result in a more modular, clean, and most likely faster and therefore more efficient robot performance.
- Hard coded colour detection - The reliance on the specific Cluedo characters provided is a weakness, with its ability to detect them linked to the specific colours of each of the characters. If these characters are swapped out, while the matchShapes solution would still work, the initial detection of these characters which launches the matchShapes confirmation, would no longer function. A solution that solely relied on the input images, such as extracting a range of colour from within the image itself, or relying solely on real-time facial detection, would be a more poignant and flexible solution.
- Move_base tweaking - with the use of the move_base movement, it can have its movement as well as other variables altered. Finding the most efficient set of variables would most likely result in cleaner and quicker movement of the robot, keeping the default values can therefore be considered a weakness, albeit a small one

Conclusion:

In conclusion, this project has been a partial success. The TurtleBot has fantastic character and shape recognition capabilities, whilst being simple, accurate, efficient and robust. However, it also comes with significant weaknesses such as not having the ability to autonomously move around its environments and thus requiring teleoperation.

References:

Bapat, K., and Mallick, S. 2018. Shape Matching using Hu Moments (C++/Python). [Online]. [Accessed 9th May 2023]. Available from: https://learnopencv.com/shape-matching-using-hu-moments-c-python/#:~:text=Hu%20Moments%20(%20or%20rather%20Hu,sign%20changes%20for%20image%20reflection.

Statement:

We would like individual marking.