Considering options(design stage)

Option1:
In the step of detecting the green room, we intend to check the color of the doorway of the room by only identifying green and red. Because color is the only criterion that distinguish different rooms. However, we found that there may exist other red or green objects around our robot that may affect the it to recognize the target. Therefore, in the step of recognizing green, we enable our robot to recognize shape and color at the same time. This would ensure that robot can find a green circle.

Option2:
In terms of distinguishing different characters, we intended to use color recognition to distinguish different characters at first. These four characters have significantly different colors, so we initially established four different color recognition methods. However, during subsequent tests, we found that the surrounding obstacles were also orange, which had a strong interference when identifying orange character ‘scarlets’. For example, robots would mistake obstacles around them for ‘scarlet’. So we took the method named ‘template matching’ provided by the teacher to avoid this problem. Basically, we import the images that need to be recognized into the robot, then let the robot find the picture that would match the images we have imported.

Option3:
There are coordinates of four key points. The robot need to walk from the entrance of the door to the center of the room. At first, We thought it could be directly realized by implemented by the goto method from the GoToPose class. However, during the testing process, we found that the robot could not recognize its current specific position. Also, the robot could not reach the given point which is the center of the target room. Because the map would be invisible when robot started to move. So we set a lot of intermediate points later, so that the robot can move step by step before finding the target point. But when we tested the program, we found that after turning off the sensor, the map is visible, so we don’t need to set these points to let the robot know its position.

Option4:
We used the cluedo character given by the teacher to match the picture on the wall directly, but it failed. The reason is that the

template is different from the actual image size. So we use resize method in template, and loop over the input image at multiple scales (i.e. make the input image progressively smaller and smaller).[1]

[1]Available from: https://www.pyimagesearch.com/2015/01/26/multi-scale-template-matching-using-python-opencv/

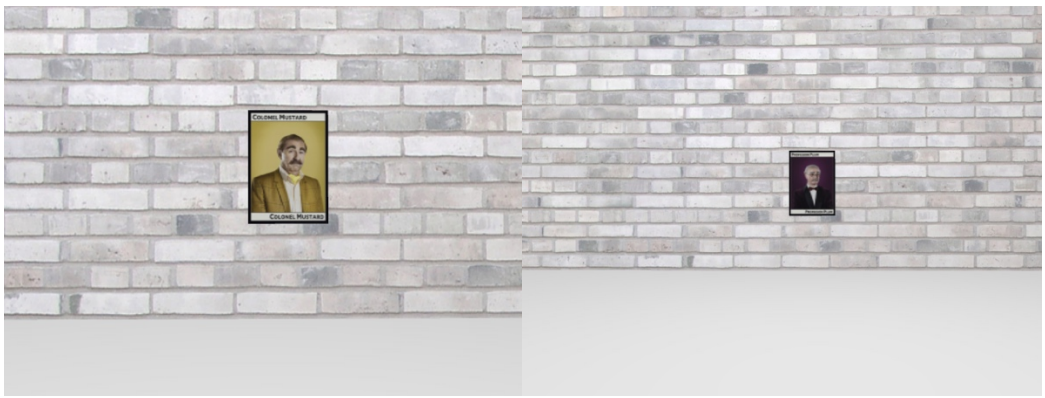
Main idea:

There are coordinates of four key points. Firstly, let the robot move to position_room1_entrance and turn around. Then check if green exists. If the green color exists, check if the green area at the entrance is circle. If the green area is circle and the area greater than 4500, it means this room is the green room and the robot will enter the center position of this room. Otherwise the robot goes to the position_room2_entrance. If the robot still could not find the green circle, it means there is no green room. After finding the correct entrance, our robot would enter the room and go to the appropriate point which is the position_room_centre. The robot rotates, taking a picture every 45 degree, and using the template to match. Finally, the picture with the highest matching degree is output save the name of this cluedo image in a txt file.

Solution test

Cluedo image test

In the map given by the teacher, we changed the four different character images to test. Four cluedo images can be identified.



Green room test

In addition, we changed the position of the green circle and the red circle on the door to test whether the robot can find the green circle.

Here is a video of a test that succeeded after we changed the map.

https://youtu.be/bdm4ml7IAZA

Room display test

Finally, we changed the structure of the room to test whether the robot can accurately find its position.

Here is a video of a test that succeeded after we changed the map.

https://youtu.be/NJTePYBQuJQ

Test record

The link below is a video record of running our program using the map given by the teacher. The program runs successfully.

https://youtu.be/WruXwtANeso

Limitation

Similar obstacles or non-standard images can affect finding the green room entrance

We distinguish the green room entrance by determining the shape and color of the mark. The green range is set from [40, 0, 0] to [80, 255, 255]. The range to recognition circles is corners >= 10. If

there are similar color obstacles, it will affect the robot to find the green room entrance. Or the shape and color of the mark is not standard enough(for example a ellipse or a polygon), the robot cannot recognize it.

Keep a distance when identifying pictures to ensure full scanning
We use class template() to find the image which is matched with the given images. The robot should keep the appropriate distance from the wall. If it keeps too close or too far from the wall to scan the whole image, the matching will fail.

Need to manually set the robot to cluedo image
The robot turns to the location of the picture according to the coordinate direction we manually set. Therefore, each time you change the map, you need to manually set the coordinate position, otherwise the robot cannot find the position of the picture, and it cannot match the template.

Scenarios where the robot might not work:
In the distinguish the green room part, if too many obstacles which has similar green color and similar circle shape with the green circle, the robot might recognize the obstacle as the green circle. In the distinguish the cluedo image part, the robot only rotates 45 degree every time, if the cluedo image is out of sight of the robot, the robot cannot find the image and cannot perform the next step of identifying the cluedo image.

Code Explanation

The whole program is divided to greenIdentifier class, cluedoIdentifier class, GoToPose class and main function. GoToPose class is given by teacher, So the code and the main function, greenIdentifier class and cluedoIdentifier class will be discussed in detail in the report.

greenIdentifier

Write the code about color recognition in the callback function, the robot will call the callback method in a loop.

Set the global variables that define the colors that the project needs to recognize and set their initial values to 0.

Firstly, the HSV color value range of green is defined. The RGB color value can be easily found online and the below website can convert RGB color value to HSV color value.[2]

Then cvColor is used to convert the OpenCV image from RGB color space to HSV color space. The inRange function can implement binarization to check if array elements lie between the elements of two other arrays. So colors outside the given range are covered by the mask.[3] The Bitwise_and function is an AND operation on binary data, extracting the image of the recognized color.[4] The findContours function retrieves contours from the binary image using the algorithm [Suzuki85]. In this way, the shape of the image can be judged whether it is circular and the area is greater than 4500, so that the next judgment can filter out non-target objects(Some objects of the same color as the target object). Then assign the contour length to the corresponding variable.[5]

Make a loop to traversal the contour length of the green circle. The arcLength function is used to calculates a contour perimeter or a curve length. The approxPolyDP function is used to approximates a polygonal curve(s) with the specified precision. The len function is used to get the number of elements in this array, which is the number of corners of the polygon.[6]
A loop is set to traverse from each of the contours to count the number of corners in the contour. If corner is bigger than or equals to 10, the image is defined as a circle. cv2.moments () returns the calculated moments as a dictionary.
If the shape of the color area is circle. If the area color is green, calculate the perimeter of the outline, draw a circle on the contour, and calculate the area of the circle. Each time the robot rotates, it calculates the area of the target object and stores the number of area in the variable green_max_area. The image matrix of the target object is stored in the variable green_image.

cluedoIdentifier class

Set the four images provided as four different templates and store these four variables in the templates array.

A loop is set to iterate over each number in the array.

Resize method is used to make the template’s size could be matched with the cluedo image on the wall. And convert the template into a

grayscale image for the next judgment. The cv2.matchTemplate method is used to simply slides the template image over the input image (as in 2D convolution) and compares the template and patch of input image under the template image. It returns a grayscale image, where each pixel denotes how much does the neighbourhood of that pixel match with template. The cv2.minMaxLoc method is used to find the maximum/minimum value. Take it as the top-left corner of rectangle and take width and height of the rectangle. That rectangle is our region of template. [7]

The template matching is used to output a matrix with the matching degree of each point in the matrix.

Then we set a threshold to 0.8 and store all the matching degrees greater than 0.8 into a tuple loc, this tuple loc contains two numpy arrays to distinguish the one array's size. If the size of the corresponding template's numpy is greater than 100000, the match is successful and the name of the picture is saved in a txt file. If the num equals 1, the image on the wall is Scarlet. If the num equals 2, the image on the wall is Plum. If the num equals 3, the image on the wall is Peacock. If the num equals 4, the image on the wall is Mustard.

Main Function

Initialize some necessary variables to null or 0.

The file provided by the teacher is opened, which contains the coordinates of the four points.

The four points provided are assigned to four different arrays. The coordinates of the starting point is set. The robot walks to room1 entrance position and this behavior is set to success this boolean variable.

When success is true, it means that the robot recognizes the green circular target object. Firstly, let the robot rotates 360° in the initial position, then run the greenIdentifier class, so the robot can calculate the area of the target objects.

If the robot recognizes green and the green_flag equals True, this shows that the robot is at the green circle doorway. Because the value of the green area is a list, take the largest value and assign it to the max_area, and at the same time assign the index of the list

with the largest green value to max_area_index. At this time, green1 is a screenshot of the green circle. The imwrite method is used to save the recognized image to a file named green_circle.png. Recognize the green circle and operate the robot to enter the green circle room, using the GoToPose class. The robot moves to centre of the green room, the position is the given coordinate position.
Now set the state of get the green room centre. When success is true, it means that the robot arrives the centre of the green room. Then, let the robot rotates 45 degree every times and use the cluedoIdentifier class, finally save the image which is highest matching rate as ‘cluedo_character.png’ in the computer and save its cluedo character name in a txt file.

If the green_flag equals False, this shows that the robot is at the red circle doorway. So the robot has to go to the coordinate position of the door of another room by the goto method in GoToPose class. Then do the same thing below when robot at the door of green circle. If the robot moves to the door of another room and the green_flag is still False, it means that there is no room with a green circle entrance.

Reference:
[1]Available from:http://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImagesPython
[2]Available from: https://www.rapidtables.com/convert/color/rgb-to-hsv.html
[3]Available from: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html
[4] Available from:https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html
https://docs.opencv.org/master/d0/d86/tutorial_py_image_arithmetics.html
[5]Available from: https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html

[6]Available from: https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html
[7]Available from: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html

Real Turtlebot test:

Here is the video when we test our program.
https://youtu.be/sBz936JNIqw

Actually, we tested the program many times, and the robot already could recognize the green circle. After that the robot stopped cause the robot always feels that there are obstacles around it, so it will not move.
We adjusted the position of the robot's initial point and tried many times. During this process, the robot may hit a wall, or the actual location does not match the location in the map.
We are trying to solve this problem, but the deadline has arrived, and we can only submit the current progress. But we will continue to study this issue later.