1 Vision

2 Introduction

As the vision segment of this project, the goal is to determine the method which leads to the best detection of the marbles within the simulated area. Describing a methods as the best is here determined by its ability to fulfill the given criteria.
From the criteria specified in **??** its known that marbles of all colors must be detected adequately with the contrast between object and scene in mind, such that the agent can act intelligently based on this. Since it is the case that the agent needs a single x-value based on the marbles location in the camera image, the x-coordinate of the detected marble's center is chosen to be sent to the agent, combined with a boolean, depending on rather or not a marble has been detected.
The methods chosen to test for marble detection are as follows.

- Hough transform.
- Background projection and contour detection.

3 Hough Transform - Theory

As detection of marbles can be reduced to the problem of detecting certain shapes in an image, edge detection is key. For this to be possible the contrast between object and scene is crucial. When applying the Hough transform, the image given is converted to greyscale making edges easier to detect. By detecting the points making up the contour, these can now be converted into Hough space, by taking the angle to the point θ and the distance to the image's grid's origin s.
Plotting these points in Hough space results in a new sinusoids for each point. These sinusoids will then group of sinusoids with similar frequency, resulting in n number of sinusoid groups. The maxima of each group then represents the parameters best describing the shape. As an example we look at a straight line, which we will describe by the following model.

$$i \cdot \cos(\theta) + j \cdot \sin(\theta) = s$$

by applying this model to the points (i, j) the line parameters best describing this edge, can be found in the corresponding Hough space. Since the shape here searched for is a circle, the model here used is as follows:

$$(i - a)^2 + (j - b)^2 = r^2$$

The nature of this operation results in the groups having point with a high concentration of crossings. These points corresponds to the line formula best representing the edge found. Since the shape we want to find depends on its center (a, b) and radius r, three parameters are unknown. To solve this problem, either the radius must be known, which in this project it is not the case, or a three dimensional representation of the Hough space must be applied. The maxima for this three dimensional sheet will then represent the center coordinates and the circle radius bes representing the curved edge found. Based on this theoretical background the OpenCV function `HoughCircles()` detects and returns a vector of the x-coordinate, y-coordinate and radius for each circle detected.

4 Hough Transform - Implementation

As a way of implementing the Hough transform the OpenCV function is used.

```
HoughCircles(src_gray,circles_list,CIRC_DETECT_METHOD,inv_resol_ratio,u_Canny_det)
```

By observing the camera image, a large proportion of noise can be seen. To reduce the effect of this, a Gaussian filter with a mask size of 5 is applied with `GaussianBlur()`. The image from the camera is then converted to grayscale and used in `HoughCircles()`.

For appropriate use this method, default inputs are used for the circle list, detection method, inverse ratio of solution, Canny detection's upper threshold, minimum radius and maximum radius.

Therefore the tuned values are the minimal distance between circles and the threshold for center detection.

When determining threshold for center detection, it is important to consider the instability of the Hough transform. Therefore the value of center detection is tuned to reduce this instability, resulting in a more robust evaluation of contrast in the image, when detecting marble edges.

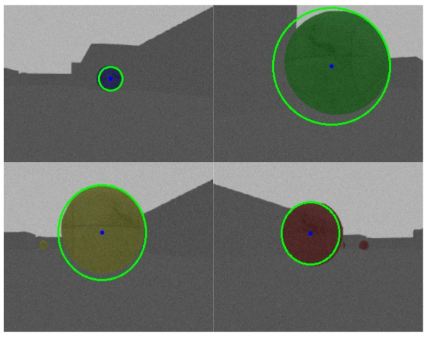

Figure 1: *Marble detection with colors.*

The instabilities regarding detecting circles outside marbles has now been reduced, such that circles far from being the marbles now are not detected. When testing the performance of this tuning, another problem occurs surfaces. When the distance between marbles are either very short, or the agent are close enough to a marble that it takes up most of the scene. If the case where the distance between marbles are short are met, the Hough transform seems to flicker between the marbles, thus causing the agent to move in a sinusoidal pattern. When getting too close to a marble the instability of the Hough transform becomes more influential and a large number of widespread Hough circles are detected. To solve both of these problems, the parameter determining the distance between marbles we wish to detect is tuned.

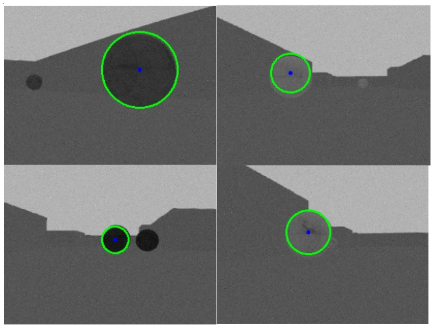

Figure 2: *Marble detection without colors.*

Now having tuned these parameters, the results can be seen applied on different colors on figure 1 and 2.

5 Background projection and HLS - Theory

A different algorithm to determine the presence of marbles are contour detection, which here will be discussed performed with HLS conversion and with background projection.

While these method shows great potential for robust detection and localisation of marbles, they are highly dependent on knowing the color of the marbles. Since the goal for this project involves solving the problem regardless of the marble color, these algorithms will not be sufficient, yet are discussed with the goal of future implementation with other goals in mind.

To perform edge detection by use of HLS conversion, we first convert the image to HLS, representing the Hue, Lightness and Saturation of the image. By then going through the image checking each pixel's hue channel, the color of the given pixel can be found. Comparing this value to the threshold representing the known color of the marble, a binary image can be made, and contours can be detected. By then applying the contours onto the original image, the contours can be seen by the agent and responded to.
The other method mentioned for determining the whereabouts of the given marbles, is by applying background projection. This method checks for the presence of marbles and crops a region of interest (ROI) being the colored pixels making up the marble. This again is dependent on the ability to locate the marble based on color. When the ROI then is found the image then can be converted to binary, by the use of thresholds on the RGB channels for the image. The binary image can then be treated in the same manner as described above, and a contour can be found.
Further use of these methods will be discussed in section 7.

6 Background projection and HLS - Implementation

In this section the implementation details related to the algorithms mentioned in section [section ref] will be described.

To execute the algorithm using HLS, the OpenCV function `cvtColor()` which takes the original image, the new image and an indicator of the type of conversion wanted. Default inputs are used with the indicator being `CV_BGR2HLS`.
Now having converted the image's channels from BGR to HLS, thresholds are applied to determine the pixels with a certain color.
For example red pixel detection would have the threshold interval $[170; 10]$, here the order indicates the interval with regards to the circularity of the hue channel in the HLS spectrum.
To apply this threshold the OpenCV function `threshold()`, which takes the original image, the new image, the lower and upper bound for the threshold and the method used. The bounds are chosen based on the HLS spectrum and method here used are `CV_THRESH_BINARY`.
A binary representation of the image has now been found and the OpenCV function `findContours()` can now find the contours. Standard inputs are used when executing this function.

To now introduce the implementation of background projection, the marble must first be detected using the OpenCV function `threshold()`. When the marble is detected a fitting ROI is extracted containing the colors of the marble. The region as a subpart of the frame from the camera now contains the colors making up the marble surface. Based on this, the threshold conversion to binary as described above is applied in conjunction with the `findContours()` method, the contours are found and placed on the original image.

7 Discussion

Given the algorithms implemented and described in section [ref to section], quality of results together with improvement possibilities are discussed in this section.

To first consider the results shows in section 4, As expected the marbles being different colors are not the attribute effecting marble detection the most, instead it is the contrast and with great certainty marbles can now be found.
As a non optimal attribute of the algorithms used a small uncertainties occur do to the instability in the Hough transform.
To solve this problem and optimize performance a combination of the methods described in section [section ref] could be applied for future implementation.
The method here would determine the location of the marble with the use of `HoughCircles()`, thus center and radius are determined. Based then on radius and center a ROI i chosen within the found circle. The found ROI would then be used for background projection, thus being able to detect the actual colors within the marble. Having done this the image can be converted to binary and a contour can be found.
By applying this method marble detection would be independent of marble colors, while having the contour accuracy.
If further precision would be needed, camshift can be added by applying the OpenCV function `CamShift()`. By passing `CamShift()` the previous frame, the current frame and some ROI for analysis, movement can be compensated for such that a more smooth following of the marbles when the robot i moving is achieved.

8 Conclusion

Based on the project criteria presented in section **??** it can be concluded that the vision segment of the project has been a success with regards to reliably detect marbles.
Further improvement has been presented for future implementation in section 7.