Pathfinding and positioning in a labyrinth game using a wide-angle camera

Stefan Olsson

LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
ISRN LUTFD2/TFRT--5964--SE
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

© 2015 by Stefan Olsson. All rights reserved.
Printed in Sweden by Tryckeriet i E-huset
Lund 2015

Abstract

Alten AB has a technology demonstator in the form of a motorized and camera equipped large scale labyrinth game. The ball position is controlled by a ABB industrial PLC connected with Android tablets for user interface and a camera as a sensor for the ball position.

This thesis demonstrates the ability to place a wide angle camera inside the cabinet, correcting the lens distortion caused by the wide angle lens and detect the ball with the use of a circular Hough transform. A path is also generated from the ball position to any position of the maze by capturing an image from the camera, generating a map for subsequent pathfinding, using an improvement of the Dijkstra's pathfinding algorithm named Theta*. It further demonstrates the feasibility of using the computing power of the camera for both pathfinding and ball positioning.

Acknowledgments

I would like to thank my supervisor Anton Cervin for his valuable input to the work. I would also like express my gratitude to Erik Lindmark from Alten AB for the opportunity to work on their labyrinth demonstrator. Further I would like to thank Fred Juhlin and Jonas Nilsson from Axis for all their knowledge of the cameras and input during the selection of a viable camera as well as supplying the new camera needed for the thesis.

Contents

1. Introduction **9**
- 1.1 Background . . . 9
- 1.2 The labyrinth . . . 9
- 1.3 Goals . . . 10
- 1.4 Method . . . 11
- 1.5 Limitations . . . 11
- 1.6 Related work . . . 12

2. Theory **14**
- 2.1 Introduction . . . 14
- 2.2 Ball and beam for one dimension . . . 15
- 2.3 Lens distortion and correction . . . 16
- 2.4 Ball detection . . . 18
- 2.5 Navigating the maze . . . 21
- 2.6 The map . . . 21
- 2.7 Pathfinding . . . 23

3. Design **29**
- 3.1 Overview . . . 29
- 3.2 Camera software . . . 32
- 3.3 A note on software performance . . . 32
- 3.4 Twodimensional lookup table (LUT) . . . 34
- 3.5 Fish-eye distortion . . . 35
- 3.6 Generation of the distortion algorithm and LUT . . . 35
- 3.7 Camera ball detector . . . 35
- 3.8 Camera map generation . . . 37
- 3.9 Camera pathfinding . . . 38
- 3.10 Using the system . . . 40

4. Discussion and results **42**
- 4.1 Camera performance . . . 42
- 4.2 Control performance . . . 42
- 4.3 System comparison . . . 43

4.4 Ball detection 45
4.5 Path generation 46
4.6 Future improvements 46
Bibliography **48**

1

Introduction

1.1 Background

Alten AB has designed a large labyrinth game to be demonstrated at exhibitions and fairs as a way of attracting interested persons to their stand. Alten is a French consulting enterprise employing around 16000 persons. Alten AB, the Swedish part of Alten, employs 1200 persons working mainly in the areas of embedded systems and mechanical design. The customers in Sweden include Axis AB and ABB AB. ABB has supplied a PLC (a industrial control system normally used for automation) and servo motors to the labyrinth as a good way to showcase their products.

A previous thesis, [Persson and Wadman, 2014], has extended the labyrinth system by adding automated positioning of the ball with an Axis network-connected camera mounted on a pole above the playfield. The camera continuously transfers ball position data to the PLC. The PLC can regulate the ball position using the servo motors to control the angle of the maze playfield. The camera above the board detects the ball by comparing camera images on a frame by frame basis extracting the changes between the frames and from that calculating the position of the ball. A deficiency of this solution is that the ball needs to be moving to generate changes in the frame data. With a slow moving or stationary ball no positioning information is available. The algorithm only locates the ball within a small part of the frame and needs information about the current ball position before starting the algorithm. It is possible for the algorithm to lose track of the ball, being unable to locate it again if the ball is outside of the search area.

1.2 The labyrinth

The labyrinth consists of a box 73x73 cm in size with a transparent maze playfield with holes and walls where a metal ball with a diameter of 3 cm is placed. The playfield is positioned on ball bearings at the center of the sides with a gimbal mechanism to allow the playfield to tilt in the X and Y directions. Figure 1.1 shows the labyrinth game.

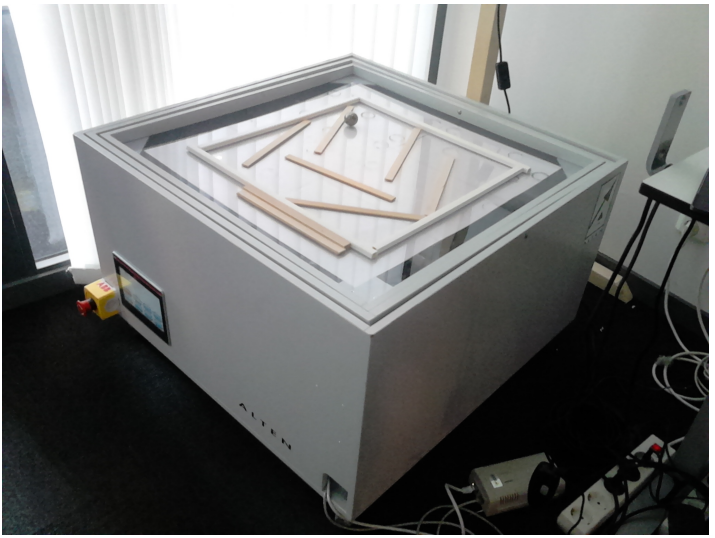


Figure 1.1: The labyrinth game with the adaptable maze plate

Two servo motors control the playfield angles via moment arms connected to the frame of the board. The moment arms are restricted to a travel of $\pm 20^{\circ}$ by a failsafe mechanism in the PLC to stop them hitting the floor of the box or destroying any electric devices in the cabinet.

An industrial PLC system, ABB PLC 500, is responsible for controlling the servo motors via PID control. Two Google Nexus-7 tablets are running software that commands the PLC by sending setpoints to the PLC through a WiFi network. All other components communicate via a 100 Mbps Ethernet network.

1.3 Goals

The aim of this thesis is to replace the overhead camera and place a Axis network camera inside the labyrinth cabinet looking up to capture the backside of the transparent playfield. The labyrinth is intended to illustrate the capabilities of the ABB PLC, the use of a network-connected Axis camera as a sensor, and work as a crowd catcher at seminars and exhibitions. The current setup with a camera on a 200 cm pole is hard to transport and needs to be calibrated for the surrounding environment. By placing the camera inside the cabinet the labyrinth becomes easier to transport and handle.

Another goal is to replace the camera algorithm to improve the localization of the ball and allow the position to be detected over the entire playfield, regardless of whether the ball is moving or not. The current solution needs to have the ball at a

known position in order to detect it properly, and the ball must also be moving to find its position.

A final aim is to have the ball navigate the maze to an arbitrary point regardless of where the ball starts. The current solution of navigating the maze consists of a file with requested ball positions. This solution requires the ball to start and stop at predefined positions. This thesis intends to improve this by allowing the ball to start and end at arbitrary positions in the maze.

1.4 Method

The thesis is based upon the previous work performed on the labyrinth, more specificly the work performed at Alten to create the labyrinth and the thesis [Persson and Wadman, 2014]. Literature studies were performed to identify viable candidates for the various algorithms needed for the thesis. The literature was located by using the university library search engine for scientific works. Ball detection algorithms were studied, camera lens rectification was studied, and pathfinding algorithms were studied. Good candidates for the various algorithms were selected based on computing efficiencies under the assumption that the camera performance would be a major bottleneck.

The hardware of the labyrinth was adapted and the new camera was mounted in the proper location at the center inside of the cabinet. Some electrical connections were rewired and the mechanics of the moment arms were improved to remove any play that would impede the control of the board.

The algorithms were implemented in the camera, and adaptations in the tablet software and the PLC software were made to use the new camera software.

The positioning accuracy of the ball was measured and experiments to measure the step response of the control loop were made. No measurements of the pathfinding were performed since it has no other impact on the performance of the system and will always find a path to the destination if there exists one.

Both the total processor load and the load by the camera algorithm were measured with the help of built-in monitoring in the camera. Linux provides commands for measuring the processor load by the commands top and mpstat to check the load on the various processes in the camera.

1.5 Limitations

The size of the cabinet, both the distance from the bottom to the playfield as well as the dimensions of the playfield, narrowed the selection of cameras available. The camera selected would have a good amount of processing power as well as a field of view large enough as to catch the entire height of the playfield. A further limitation was the maximum frame-rate of the camera when using the maximum field of view. The definitions of the project were set based on camera restrictions. The used field

of view was changed to improve the frame-rate of the camera and to improve the control of the ball. The implementation effort was concentrated on the ball detection and path generation software in the camera.

Several restrictions were encountered when the camera was to be placed inside the cabinet. The first was that in order to get a good picture the camera needed to be placed at the center of the box to symmetrically capture the playfield. That space was occupied by the PLC, but this was easily solved by moving the PLC somewhat to the side.

Another restriction was that the distance between the cabinet bottom and the playfield was a mere 60 cm. Since the playfield has a width of 71 cm, a camera with a normal capture area was not able to see the entire playfield. A camera with a fish-eye lens was needed to capture the playfield. A camera with that type of lens would produce a severely distorted image, which would interfere with the ball positioning algorithm. An Axis M3006-v camera was selected since it had a field of view (FOV) of 134°. At this FOV the camera would produce an image of 3 Mp at 16 frames per second (FPS). The camera capturing resolution is fixed with regards to the FOV. The camera capture image has a height that exactly matches the height of the maze playfield.

The camera is running Linux internally and has provisions for creating and running external applications written in C that can extend the functionality of the camera. The framework for creating the camera applications is called ACAP (Axis camera application framework).

1.6 Related work

Variants of the ball and plate (or ball and beam in one dimension) appears frequently in the literature and many variations of the control approaches exist.

Ball and beam related work

The ball and beam is a good example of an unstable system, and the control of the system is studied in many control theory courses. A ball is positioned on a tilting beam and by controlling the angle of the beam with a motor, the ball position on the beam is controlled, [Virseda, 2004] has written a Master's thesis modeling the system using Modelica and controlling the ball and beam using a observer based controller. The experimental setup uses a camera for ball positioning with the control system running on a PC computer. The vision system is preconfigured as a part of the experimental setup and no specific work on the vision system were performed.

A different approach is taken by [Chang et al., 2013] that models the system with a fuzzy logic approach. An experiment is conducted on a ball and beam system with a DSP/FPGA based controller during which an iterative dynamic control process is run to improve the system robustness. The experiments shows good performance of the proposed control.

Ball and plate with camera

The transformation of the ball and beam system to two dimensions does not create any serious difficulties above those imposed by the one dimensional system.

[Galvan-Colmenares et al., 2014] uses a plate with a camera to control a ball. The ball is located by a camera above the plate as opposed to this thesis where the camera is placed below. A PD controller with a nonlinear compensation is used to control the ball.

[Espersson, 2010] has looked into various vision algorithms for the ball and plate as well as the ball and beam systems. A web camera connected to a PC was used as a sensor, and a open source image processing library was used. With faster processor and a complete library with image processing functionality the process of detecting the ball becames easier. Tests were also conducted with the ball on plate system connected to an ABB robot.

2

Theory

2.1 Introduction

To get a complete system, several algorithms need to work in unison to complete the desired task. The ball needs to be positioned by the camera, it needs to be controlled by the PLC, the algorithm for pathfinding needs to create a viable path, and a means for generating a map is needed.

The used camera has a very short focal length. This makes the images captured by the camera having a very wide area with the drawback of a distorted image at the edges.

All algorithms involving images rely on the image having the correct perspective, so a means for stretching the camera image to resolve the nonlinearity is needed because of the wide FOV used to capture the playfield.

With the undistorted image it is possible to capture a image of the entire playfield to generate a map. But since the center of the ball is at a finite distance from the edge, a offset need to be generated from all obstacles in the image, such that the path generation creates a path that will stay clear of all obstacles. The distance transform will perform this by thickening all obstacles enough that the ball will pass all obstacles.

The image created by the distance transform represents a map where the center position of the ball is allowed to traverse.

To navigate the maze a pathfinding algorithm is used. A* [Hart et al., 1968] and Theta* [Daniel et al., 2010] are improvements of Dijkstra's pathfinding algorithm wich is described in [Kingston, 1995]. In this thesis the Theta* algorithm is used to find a path between the start and destination points. The Theta* variant is able to generate a route as a straight line between two points, whereas the other variant is restricted to the grid intersections. A means to generate a straight line is needed for this, and the Bresenham line generating algorithm [Bresenham, 1965] is used for this.

The PLC needs to know the position of the ball in order to control it. The Hough algorithm [Weiss, 2009] is used to locate the center of the ball in every frame once it has been undistorted. A lot of image processing algorithms exist to find shapes

in images, all with its pros and cons. The previous version of the labyrinth control [Persson and Wadman, 2014] detected the ball by finding the changing parts of an image. This may generate false positions if other parts of the image is also moving, and it may be impossible to locate the position of a non-moving object.

2.2 Ball and beam for one dimension

The ball on plate can be viewed as two ball and beam systems, one for each dimension X and Y. The ball and beam system is a useful application to demonstrate an unstable control system.

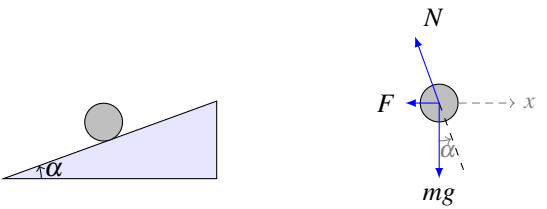


Figure 2.1: The ball and beam system

A ball is placed on a horizontal beam where the beam angle can be altered. The acceleration of the ball is affected by the gravity and when the angle of the beam changes, the horizontal force F affecting the ball is

$$F = mg \cdot \sin(\alpha) \tag{2.1}$$

where m is the weight of the ball, g is the gravity constant and α is the angle of the beam. We are only dealing with small angles of the beam and thus we can replace $\sin(\alpha)$ with its first order Taylor expansion, α. This gives

$$F = mg\alpha \tag{2.2}$$

The position of the ball can now be expressed as the second derivate of the acceleration, using Newton's second law of motion we get

$$F = ma \Leftrightarrow F = m\ddot{x} \tag{2.3}$$

Here x is the horizontal position of the ball. Combining Equation (2.2) and (2.3) gives

$$m\ddot{x} = mg\alpha \tag{2.4}$$

$$\ddot{x} = g\alpha \tag{2.5}$$

By Laplace transforming (2.5) we have the transfer function

$$X(s) = bs^{-2}U(s) \tag{2.6}$$

The system is not affected by the mass of the ball, and by altering the angle the position is controlled.

The system is open-loop unstable, and in order to control the ball position we need to have a regulator. One of many ways to solve this is to use a PID controller. The most basic form of the controller is

$$u(t) = K\Big(e(t) + \frac{1}{T_i}\int e(t)dt + T_d\frac{de(t)}{dt}\Big) \tag{2.7}$$

$$e(t) = r(t) - y(t) \tag{2.8}$$

where $r(t)$ and $y(t)$ is the reference and measured ball position and $u(t)$ is the controller output. This is the analog variant, and because we are using computers we need to have a discretized, sampled version of the above instead. The theory behind this is described in [Årzén, 2012]. With the added improvements to the control such as anti-windup and setpoint weighing the resulting formulas are described below.

$$v(t_k) = P(t_k) + I(t_k) + D(t_k) \tag{2.9}$$

Where the details of the formulas are

$$\begin{aligned}
P(t_k) &= K(\beta r(t_k) - y(t_k) \\
I(t_k+1) &= I(t_k) + \frac{Kh}{T_i}e(t_k) + \frac{h}{T_T}(u(t_k) - v(t_k)) \\
e(t_k) &= r(t_k) - y(t_k) \\
D(t_k) &= \frac{T_d}{T_d + Nh}D(t_k - 1) + \frac{KT_dN}{T_d + Nh}(y(t_k) - y(t_k - 1))
\end{aligned}$$

Here, the sampling time is h and sampling points are t_k. β is the setpoint scaling, u and v is the saturated and non-saturated output signal, N is the maximum derivate gain and, finally, T_T is the anti-windup parameter.

2.3 Lens distortion and correction

In order to position the camera at a short distance from the playfield, a camera with a very wide field of view is needed. To capture such a large angle, a lens with nonlinear magnification is used, a so called fish-eye lens. The magnification changes radially from the center of the lens, with a larger magnification further away from the lens. The effect can be seen in Figure 2.2.

Figure 2.2: A example of how the image is distorted by a fish-eye lens

To undistort the image there is a need to look at every pixel in the distorted image and calculate a new radial position in the transformed, undistorted image. The radial distance from the center of the image can be calculated with the help of physics and Snell's law. [Dhane et al., 2012] showed the relation to be

$$R = f * \tan(\sin^{-1}(\sin(\tan^{-1}(\frac{r}{f})) * k)) \tag{2.10}$$

where we have R as the radius from the center of distorted image, r is the radius from the center of the corrected image, f the distance from lens to picture plane and k a constant dependent on the specific lens. The exact solution can be quite taxing on computing resources where a lot of trigonometrics have to be calculated and the f and k constants need to be known or approximated. It is possible to precalculate the values, but the k constant is not normally known and the large reliance on trigonometric functions may create a error when calculating R. As a consequence it is often better to approximate the distortion which can be done using a polynomial least squares fitting.

Polynomial least-squares fitting

A method of undistorting the image is to use a transformation based on a polynomial least-squares fitting. It has previously been used by [Brown, 1971] and [Shah and Aggarwahl, 1996]. A property we want in the corrected image is that a straight line is straight. By placing a calibration image in front of the camera it is possible to extract enough information from it to generate a transformation function. [Brown,

1971] uses a checkerboard image and [Shah and Aggarwahl, 1996] uses an orthogonal grid of points as the calibration image. The polynomial is expressed as

$$\begin{aligned} x_u &= x_d + \Delta x(K_1R^2 + K_2R^4 + K_3R^6 + \ldots) + [P_1(R^2 + 2x^2) + 2P_2xy][1 + P_3R^2 + \ldots] \\ y_u &= y_d + \Delta y(K_1R^2 + K_2R^4 + K_3R^6 + \ldots) + [P_1(R^2 + 2x^2) + 2P_2xy][1 + P_3R^2 + \ldots] \end{aligned} \tag{2.11}$$

x_u, y_u are the coordinates in the corrected image and x_d, y_d represent the coordinates in the camera image.

$$\Delta x = x_d - x_{dc} \tag{2.12}$$

$$\Delta y = y_d - y_{dc} \tag{2.13}$$

$$R = \sqrt{\Delta x^2 + \Delta y^2} \tag{2.14}$$

x_{dc}, y_{dc} is the center of the radial distortion in the camera image. K_1, K_2, K_3 are the radial distortion coefficients and P_1, P_2, P_3 are coefficients for center distortion. Since we know that a straight line should be just that, positions from the calibration image (checkerboard) are selected and matched with corresponding points from a straight line, where the errors are used to calculate the coefficients in (2.11).

2.4 Ball detection

The detection of the ball is based on the round shape. By finding all edges in the image with an appropriate algorithm and then using the resulting edges that most likely correspond to a ball with radius R we have a robust solution that accurately selects the ball from an image with other objects.

Edge detection by Sobel filter

The Sobel filter is used to detect edges by calculating abrupt changes in color in the image and produces a separate value for gradients in the X and Y directions.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A \tag{2.15}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A \tag{2.16}$$

A represents the image where the edges should be detected and it is convoluted with the corresponding 9-element matrix. The convolutions in (2.15) and (2.16) produce a result with the same dimensions as the original image (A), responding maximally to edges running vertically (2.15) or horizontally (2.16) at the same time as edges

(a) G_x

(b) G_y

Figure 2.3: The effect of using the Sobel operator on image

in the other direction are ignored. The effect can be clearly seen in Figure 2.3 where the horizontal or vertical lines are suppressed. Note that the sloped lines are present in both images but with opposite sign.

From this the sum of all edges is given by the magnitude

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.17}$$

which displays all edges in the image.

Hough transform

The idea behind the Hough transform is to utilize the edges located by a edge detection algorithm such as the Canny edge detector [Canny, 1986] or the Sobel edge detector and from the edges derive information about the shapes in the image. The Canny edge detector is good at detecting the exact location of the edges but with a larger computing effort than the Sobel edge detector. The computing resources in the camera is limited and as a result the Sobel edge detection algorithm is used. The first use of the Hough transform was to get information about straight lines in the images and it has then been extended to find other shapes as well.

Circular hough transform In this thesis the focus will be on the circular Hough transform in order to detect the ball. The Hough transform is able to locate circles of radius R in a image. The algorithm is based on a voting scheme for the most probable positions of circle centers. By looking at every detected edge point and increasing a two-dimensional accumulator for the positions that may be the center of a circle, we will get local maximums in the accumulator were we have a high probability of circle centers, see Figure 2.4. For every diameter of circle that we need to discover, a new accumulator is needed, so the memory footprint increases with every diameter that is searched for.

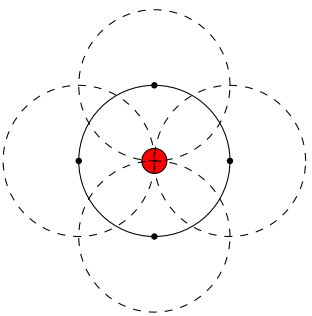


Figure 2.4: Circular Hough, the probable center of the circle is where we have the most intersecting points, the dashed circles are possible center positions from the edge-points

Gradient direction By utilizing the knowledge of the direction of the edges we get when using the Sobel operator, it is possible to narrow down the possible center positions for each edge point as described in [Weiss, 2009].

The direction of the gradient is calculated with the help of the Sobel operators

$$\theta = \tan^{-1} \frac{Gy}{GX} \tag{2.18}$$

When the requested radius (R) is known it is then possible to find the center of a circle as

$$x_c = x - R \cdot \frac{G_x}{G} \tag{2.19}$$

$$y_c = y - R \cdot \frac{G_y}{G} \tag{2.20}$$

which represents a huge speedup as only one probable center position for each edge pixel needs to be calculated. An overview of the algorithm can be seen below, and the result can be seen in Figure 2.5.

```
Apply The sobel operator to image F(x,y), calculating Gx,Gy,G

for each pixel in F(x,y) where G>threshold
  xc=x-R*Gx/G
  yc=y-R*Gy/G
  increase accumulator Acc(xc,yc)
locate maxima in Acc wich gives ball centers
```

As the next step, the accumulator needs to be searched to find the possible center positions, which are the positions where we have a local maximum.

2.5 Navigating the maze

To steer the ball through the maze the system needs to have knowledge of how the maze looks and where the ball is allowed to roll without hitting obstacles or falling through any holes. With this knowledge it is possible to examine different paths through the maze and conclude whether there exist a possibility to reach the destination with available routes and then select the route that is the shortest in terms of cost.

2.6 The map

Three possibilities for the map of the maze have been considered.

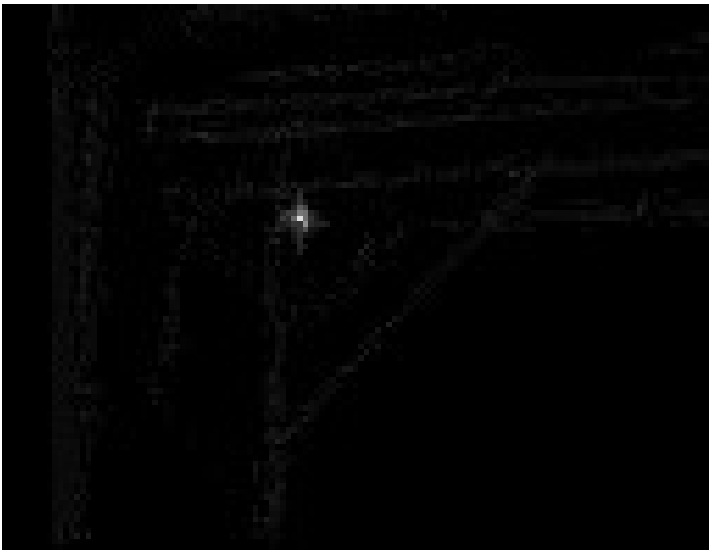


Figure 2.5: A Hough transform accumulator where the center of the circle can be clearly seen as a white star in the image

Graph-based map

A graph-based map is a neat representation of a map since the only information stored is the locations where the ball may roll in different directions, which location the ball arrives at next if it selects a specific route, and the length (cost) of that route. This type of map was not used in this thesis since the information about the map was already present in the image and further processing was needed to produce a graphbased map.

Image-based map

The image-based map is based on a *m*x*n* sized grid where each position on the grid represents a location in the maze with knowledge if this position is an allowed position for the ball or not. The available routes from one position is restricted to those of its 8 neighbors which is a free position. This restricts the path direction to 45° increments going from one neighboring position to the next throughout the entire path. It is also possible to give a cost to each position to affect the path generating algorithm.

Generate a map from the image Creating a map from a camera image implies examining each position on the map and based on the corresponding data in the image decide if the position is free or not. By utilizing a threshold-value in a gray-scale image, a binary map with free and occupied areas is created. This is not enough to create a map for the maze; the occupied areas must be extended at least up to

the radius of the ball and that is accomplished by using the distance transform, as described below, on the binary map. The resulting map contains all positions in the map where the ball-center is allowed to be positioned.

Distance transform

The distance transform generates from a binary image representing a shape or an object, a new image containing for each pixel a value describing the closest distance to the object. [Shih, 2004] describes a fast algorithm that calculates the Euclidean distance $d = \sqrt{\Delta x^2 + \Delta y^2}$ for an image with the help of a 3x3 pixel neighborhood. The effect of applying the transform is seen in Figure 2.6. The distance transform is easily used to keep a set distance from the object which is demonstrated in 2.6c where the rings represents a value with a specific distance from the object.

2.7 Pathfinding

All pathfinding algorithms inspected in this thesis need to have a map in the form of an undirected graph with all traversable positions included. The algorithms use the map to find a path with minimum cost to the destination.

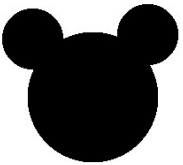
The pathfinding algorithm used in this thesis is based on Dijkstra's algorithm, but with the improved computational efficiency of the A* and the reduction of the path and waypoints the Theta* extension provides. The Bresenham algorithm is then utilized by Theta* to check the path between the waypoints.

Dijkstra's algorithm

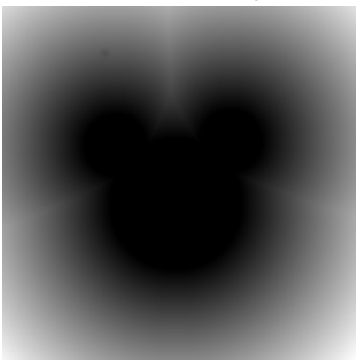
The pathfinding algorithm on which a lot of subsequent improvements are based upon was created by Edsger W. Dijkstra in 1959. The algorithm is described in detail in, e.g., [Kingston, 1995].

The algorithm is based on the idea to start at the destination and from there examine all directly accessible positions and calculate the cost to reach the destination. Now there exists a group of positions with known cost to reach the destination and a new set of directly accessible positions from the inspected group. By continuously inspecting all directly accessible positions as they become available and generate the cumulative cost to reach the destination for the position, eventually you will examine the start position, and by then the lowest cost and path to the destination will be known. A drawback of the algorithm is that it will examine all available paths without making any assumptions of which path is the best to examine. The # signs in the code indicates the modifications the A* and Theta* algorithms introduce.

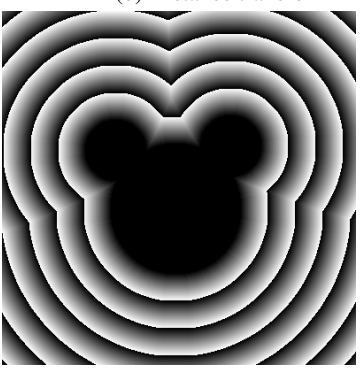
```
set all position-> cost = infinity
set all position-> not inspected

set destination -> cost = 0
```



(a) The object



(b) Distance transform



(c) Equidistant values

Figure 2.6: The effect of using a distance-transform on image 2.6a shows the picture, 2.6b shows the image with the distance transform applied and 2.6c exemplifies the equidistant points in the image.

```
set destination = inspected
position = destination

    while position !=startposition and position exist
        #A* calculates a different cost
        cost = cost to destination from position
        if not position->inspected or cost< position->cost
            position->cost=cost
            position->inspected
            set previous position
            #Theta* change the previous position
            add Directly accesible positions
        position = get Directly accessible positions
                   with least cost

    if position =startposition
       follow path from start to
       destination via previous position
    else
       No path from start to destination is possible
```

The cost function is calculated as

```
cost of edge + cost of predecessor
```

The cost of an edge is always a positive number but is is possible that the edges have different costs associated with them which enables the algorithm to avoid certain areas of the map but still use them if no other path is available.

The A* algorithm

The A* algorithm is based upon the Dijkstra's algorithm, but improves the efficiency by inspecting the routes which are closer to the start position first as explained in [Hart et al., 1968]. The addition of a term for the distance from the current position to the start will contribute to positions having shorter way to the starting position getting evaluated before positions further away from the start. The cost function is calculated as

```
cost of edge +  cost of predecessor + distance to start
```

The distance to start is an arbitrary measure to sort the best candidates for future calculations. The distance may be calculated as the Euclidean distance (2.21) or the Manhattan distance (2.22) if the map consists of an orthogonal grid. The Manhattan distance only gives an approximate distance, but is faster to use on a computer, as no expensive computation of the square root is needed. See Figure 2.7 for comparison of the distances.

$$Euclideandistance = \sqrt{\Delta x^2 + \Delta y^2} \tag{2.21}$$

$$Manhattandistance = \Delta x + \Delta y \tag{2.22}$$

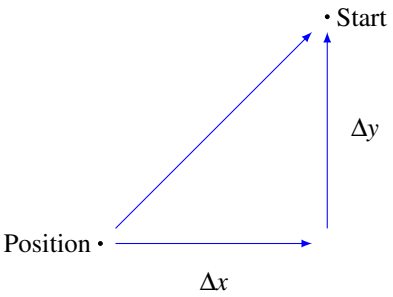


Figure 2.7: Manhattan and Euclidean distances

The Theta* algorithm

Further improvements on pathfinding is made by the addition of the Theta* algorithm which enables the algorithm to skip over waypoints along the way if there exists a straight line free of obstacles to an earlier waypoint. An analysis of the theta algorithm has been made by [Daniel et al., 2010]. The addition to the Dijkstra algorithm is small, but frees the algorithm of the restriction of keeping the path at the grid intersections as long as the map is an orthogonal grid.

```
if lineofsight between current and second last position
   skip over the previous position and record the
   second last position as the previous path waypoint
```

In order to check the free path between the two positions, a method is needed to check every position in-between to decide whether it is free of obstacles. If the map is a Cartesian grid this further implies that two waypoints on the path may not be directly accessible and the path angle may be any arbitrary angle. In the case of a grid-based map, the Bresenham line algorithm may be used as a means of generating a straight line between two points and at the same time visit all positions in-between. Figure 2.8 shows the difference between the paths generated by Dijkstra's algorithm and the Theta algorithm and how waypoints in between is removed.

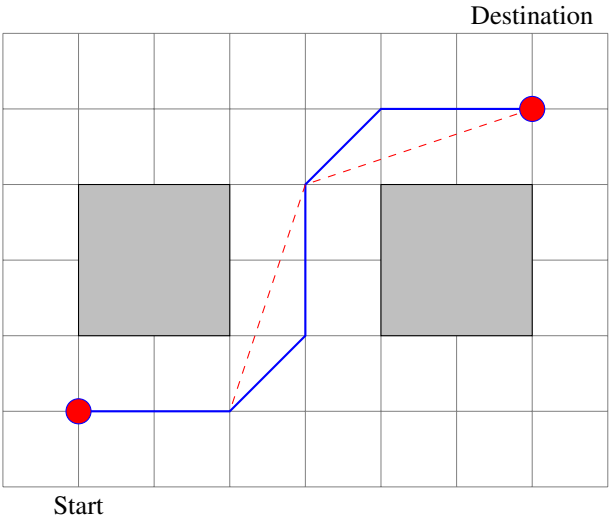


Figure 2.8: Comparison between the paths taken by the Dijkstra/A* and Theta* algorithm (dashed)

Bresenham's line algorithm Bresenham's algorithm is used to approximate a straight line between two points on a raster, and is widely used in computer graphics to generate a line. The algorithm was implemented by [Bresenham, 1965] to generate drawing paths for a pen plotter. The original algorithm relies only on integer addition and subtraction and is well suited for resource constrained computing and has been implemented in hardware in some computer graphics chip sets as mentioned in [Kingston, 1995]

The core functionality of the algorithm is described below.

```
dy=(y2-y1)
dx=(x2-x1)
delta=2*dy-dx
y=y1
for x= x1 to x2
   plot(x,y)
   delta=delta+2*dy
   if delta >=0
      delta=delta-2*dx
      y=y+1
```

The algorithm in this form is rather limited and will only work in the first octant for line angles between 0 and 45° and under the assumption that x and y are restricted to integer values.

The algorithm is extended to other slopes by changing the sign of the axes x and y but the part described above describes the fundamental functionality of the algorithm.

3

Design

3.1 Overview

The system consists of several cooperating parts in order to create a working labyrinth game, see Figure 3.1.

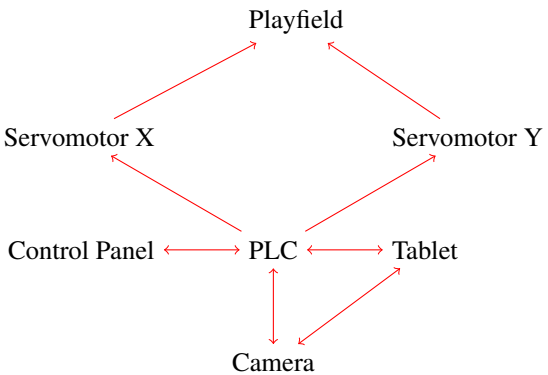


Figure 3.1: Overview of the components of the labyrinth game

PLC

The industrial PLC controller mounted inside the cabinet is handling the flow of the game and notifying the tablet or tablets of the game mode used, such as automatic play or direct control of the playfield. The PLC also receives ball position updates from the camera at the same rate as the camera frame-rate. The positioning data is used in the PLC to calculate the control signals to the servo motors using two PID control blocks, one for the X direction and one for the Y direction. The setpoints for the PID control blocks are received from the tablet or the camera depending on game mode.

PLC control panel

The PLC system has a external control panel with a touch screen that is mounted on the side of the cabinet. The panel has controls for showing a graph of the ball position, adjusting the servo motors and calibrating the plate position.

Camera

The camera, located at the center of the cabinet-floor looking up at the playfield, captures frames at the maximum frame-rate the camera can handle. The frames are analyzed to locate the ball and sends the ball center position to the PLC for each frame.

If a request for a ball destination position has been received from the tablet, the camera generates a path from the current ball position to the destination. When the search has finished, which will take several frames, the next waypoint on the path to the destination is sent to the PLC, which will move the ball accordingly. One second later a new search begins with a new ball position.

The map for the pathfinding is generated by the camera when it is instructed to do so by a command from the tablet. This makes it possible to alter the maze at will and regenerate the map.

Tablet

The tablet is used for controlling the game. A view of the board as captured by the camera is shown on the tablet, see Figure 3.8. This makes it possible to follow the ball and see the current maze used. The image is also used to set a reference point that is sent to the PLC or the camera depending on the mode. The game has two modes where the camera is used to guide the ball:

Position mode. When a setpoint is selected on the image, the position is sent to the PLC which guides the ball to the setpoint in a straight line.

Auto mode. When a setpoint is selected on the image, the position is sent to the camera which in turn calculates the proper route to the setpoint and generates waypoints which is sent to the PLC.

The auto mode allows the game to navigate the entire maze autonomously, but in the position mode, the user of the tablet needs to guide the ball through a proper route.

The tablet interface also has a button to generate a new map. The command is sent to the camera, which then generates a new map from the current image. When generating the new map it is important that only the maze walls are present on the playfield (and no ball) since everything present on the playfield will be regarded as a keep-out area for the subsequent path generation.

Cabinet

The cabinet is the box were all the hardware is mounted, see Figure 3.2. It contains the PLC, the two servo motors with their power units, the camera, various electric wiring, the wireless access point and it holds the gimbal mechanism for the playfield The cabinet measures approximately 80x80x60cm.

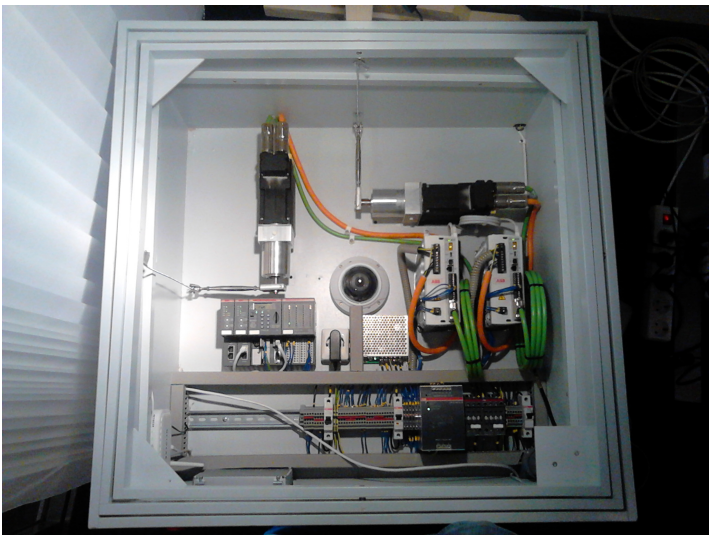


Figure 3.2: A look at the components inside the cabinet

Playfields

Three different play-fields exist for the labyrinth. The play-fields consists of a 5mm Plexiglas sheet with holes and walls.

- ABB plate, A transparent plate with a predesigned maze with ABB logo.
- Alten plate, transparent plate with a different desigh of a maze with Alten logo.
- Adaptive plate, semiopaque plate with a frame defining the play area and moveable wall segments to change the maze appearance during operation.

The adaptive plate is the one that is used the most in this thesis. The plate is semi-opaque in order to improve the detection of the maze walls. The walls consist of 12 pieces of 20 cm long bars intended to be placed on the playfield to create a maze for the specific moment.

3.2 Camera software

The software in the camera is split in two sections.

The first section is run at software startup and handles the initial setup of the software, setup of threads and allocating all memory buffers.

After initialization the main thread loop is started. This loop runs continuously and runs to completion whenever a new image frame captured from the camera is available. It will also check incoming communications from the PLC or the Android tablet, and send the ball position to the PLC. After that it is blocked by the camera until the next frame is ready.

The frame-rate of the camera is set to 20 FPS @ 2048x1536 (3 MP). This is the highest frame-rate the camera can handle when the CCD chip capture area is large enough to capture the entire maze. To improve the performance the frame-rate has in later tests been changed to 30 FPS @ 1600x1200. This resolution is scaled down to 320x240 by the camera firmware, and the fish-eye algorithm scales this further down to a square of 200 x 200 pixels.

The second thread is a low priority thread handling the pathfinding algorithm. The process will wake up once every second and calculate a new path to the destination dictated by the tablet from the current ball position. The first waypoint from the current ball position will be sent to the PLC. The pathfinding calculations are interrupted by the main thread whenever a new frame needs to be processed and as consequence it will process the pathfinding during 5-10 frames until the next waypoint is calculated. It will then sleep until it is time to work again. The pathfinding algorithm is not time critical since the PLC will position the ball on the last received waypoint until it receives a new position. A overview of the software flowcharts can be seen in Figures 3.3 and 3.4

3.3 A note on software performance

Some general ideas when writing this software has been to improve the speed by selecting which features of the language to use. The time-critical section of the program is the analysis of each frame to detect the ball position and is all done in the main loop. Most performance enhancements have been concentrated there.

The processor in the camera has no FPU and as a consequence the camera software has been designed using only integer arithmetic. The allocation of memory areas for image buffers and map data is done before the main loop is started. The allocation of memory is nondeterministic, and may take some time, therefore no memory is allocated or deallocated in the main loop, and all buffers are reused for each frame. The use of addition and subtraction has been favored over multiplication within loops.

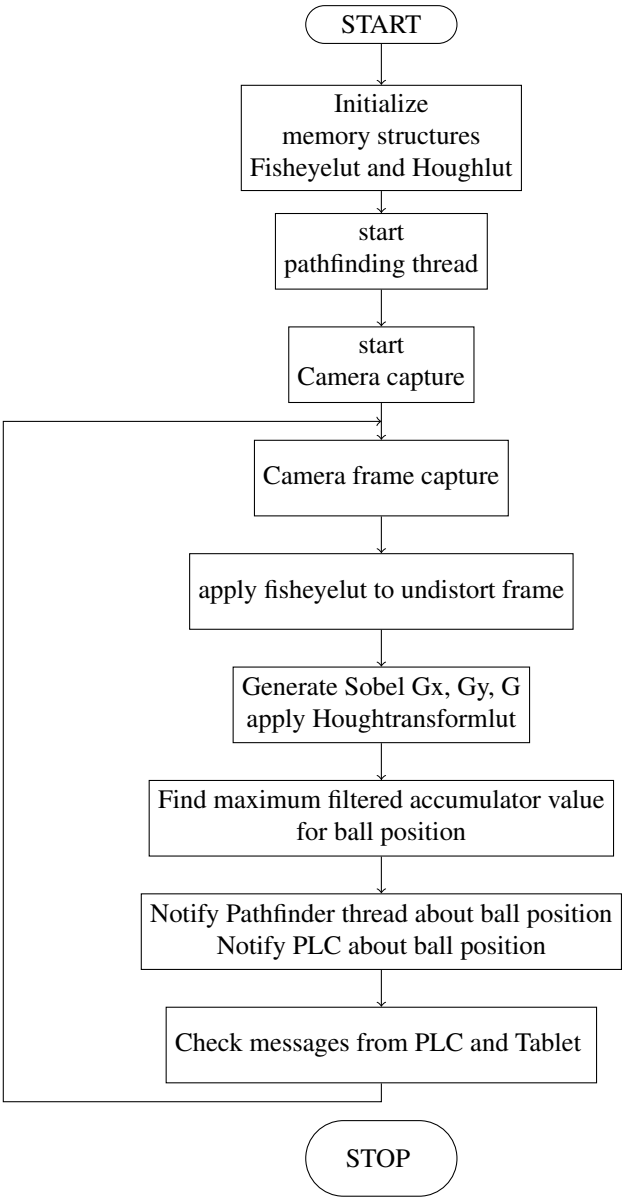


Figure 3.3: Flowchart of the camera software - main thread

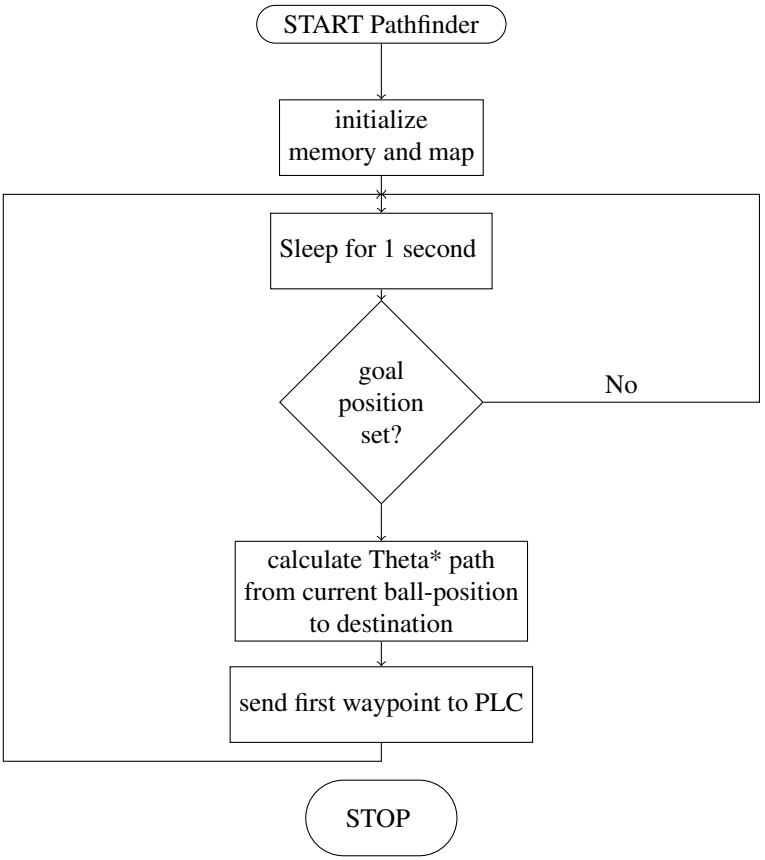


Figure 3.4: Flowchart of the camera software - pathfinder thread

3.4 Twodimensional lookup table (LUT)

A lookup table is used to store a array of precalculated values in the memory of the camera in order to speed up the calculations. The values are read from a file in the camera before running any time-critical parts of the program. A one-dimensional LUT is as simple as using a index as an offset into an array containing the precalculated values.

When a two variable function is precalculated a two dimensional array can be used. By proper selection of the variable range it is possible to speed up the lookup functionality by using bitshifting. The resulting access-functionality is shown below, in the C programming language, when the variables has a range of $0-255$ and fit in a 8 bit datatype.

```
result= *(arraybase +variableA+variableB<<8)
```

where the result is the value located at the position of arraybase + variableA + 256*variableB

3.5 Fish-eye distortion

The fish-eye algorithm generates a 200x200 undistorted image of the maze by utilizing a polynomial least-squares transformation function for each pixel in the undistorted image to pinpoint the corresponding pixel in the distorted camera image of 320x240 pixels. This is repeated for every frame.

The offset between pixels in the distorted and undistorted images does not change between frames and the values are precomputed on a laptop and stored on non-volatile memory in the camera to avoid any division and floating point operations in the camera. The images in the camera is stored in a contiguous memory area where each pixel is stored sequentially. The relative position of any two pixels in an image can then be precalculated as a single value and the offset can be made by a mere addition of a pointer value.

$$\Delta Pointer = \Delta x + \Delta y * imagewidth \tag{3.1}$$

3.6 Generation of the distortion algorithm and LUT

The image captured by the camera is severely distorted by the fish-eye lens. In order to correct the perspective the positions of the maze frame corners and center were recorded and a Python script was used to calculate a second degree least squared error in comparison with a straight line. This calibration was done on a laptop where the subsequent result was used to generate the lookup table in the camera. A graph of the calibration can be seen in Figure 3.5.

The formula in (2.11) allows for a higher order polynomial for K_n, which corresponds to the radial error correction and P_n, but after a test it was concluded that a second order polynomial with $P_n = 0$ would be enough to undistort the camera images. The resulting equation from the Python script was used to generate a look up table stored as a file in the camera and loaded to the camera memory at every startup, before the main loop was started. For each position on a 200x200 grid with a center position of (100,100) the corresponding pixel in a 320x240 image was calculated and the combined x and y offset were calculated with the help of Equation (3.1).

3.7 Camera ball detector

The ball detection is based on the fact that the ball is round and the edges with the most resemblance to a ball with the correct size is used as the ball position. To

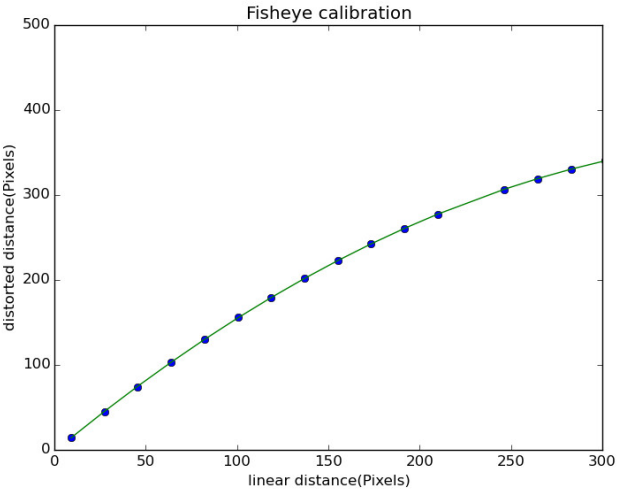


Figure 3.5: Correlation between distorted and undistorted image position in the x or y direction

minimize false detections, a thresholding check is done at the position to verify that the ball has the proper color.

Edge detection

The edge detection algorithm were selected to use the Sobel gradient because the algorithm is fast, it provides G_x and G_y needed for the subsequent Hough transform and a optimized library for the camera existed.

Hough implementation

It was decided that the calculations regarding the angle of the circle center were too taxing for the camera. Instead all these calculations were precalculated on a PC, and a two-dimensional table was stored in the camera speeding up the calculations to a mere lookup of a value.

```
Read all precalculated lut values for each frame pixel
   dx =getoffset(x,y)
   dy =getoffset(y,x)
   increase accumulator(x+dx,y+dy)
```

When as much calculations as possible are precalculated the Hough implementation can be generated surprisingly fast with only additions and bit-shifting as shown below in C:

```
// perform for all x and y in image
// two rows for calculating the hough offset for pixel (x,y)
dx=*(houghlut+G_x+G_y<<8);
dy=*(houghlut+G_y+G_x<<8);
// calculate the pixel position of the center position
// relative to the accumulator pointer
centerposition= accumulator+(x+dx)+((y+dy)*width)
// increase the pixel by one using pointer arithmetics
*(centerposition)++;
```

where G_x and G_y are the values for pixel (x, y) and houghlut contains precalculated values for

$$\Delta X = \frac{G_x}{\sqrt{G_x^2 + G_y^2}} \cdot R \tag{3.2}$$

for all G_x and G_y beween -127 and 127 where ΔX is the distance to the circle center in the X direction. Accumulator is a pointer to a memory-area with the same size as the frame image. Note that the same accumulator can be used to get the circle center distance in the Y direction by exchanging the position of G_x and G_y.

What is left after the image has been processed with the above is to find the accumulator position with the maximum value which corresponds to the most probable ball position in the image. By checking other attributes of the position such as color and previous ball position the probability can be improved further. Only the color of the ball is checked in this implementation.

3.8 Camera map generation

A map of the maze is generated by using a camera frame with the maze shown as dark shadows on the playfield as can be seen in Figure 3.6. The image is captured by the camera and the same fish-eye correction is used as in the pathfinding algorithm in order to get an undistorted image of the maze with a size of 200x200 pixels. A thresholding operation is done on the maze image where all pixels with a color value < 40 are set to 0 (black), all others are set to 255 (white). The ball position is marked by the center position. Since the ball has a finite radius the map area available to the pathfinding algorithm has to be defined by adding an offset to the captured wall positions. By applying a distance transform algorithm, a gradient map is generated where it is easy to read the distance of any point to the nearest wall. The ball has a diameter of 8 pixels, which means any position on the map where the gradient value is $<= 5$ will be regarded as a no-go zone for the pathfinder algorithm.

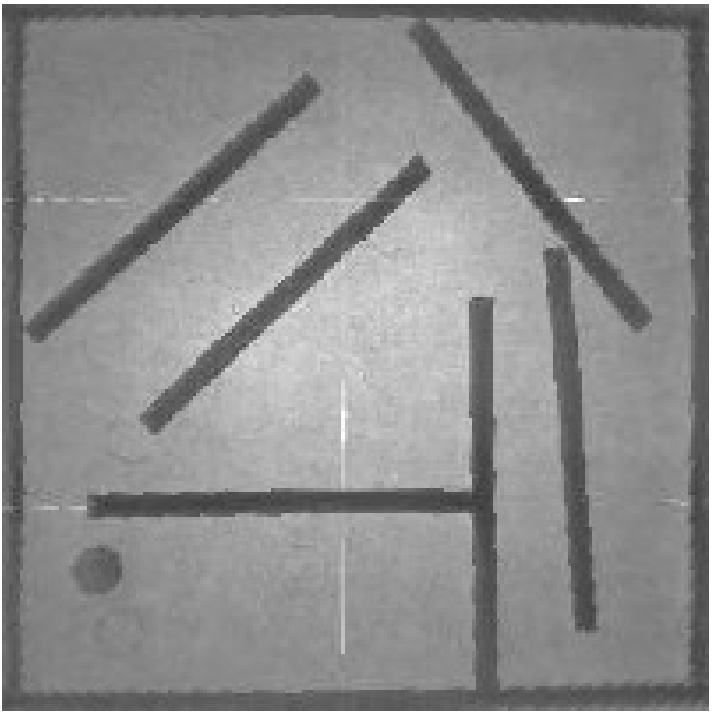


Figure 3.6: The undistorted image where the wall segments are placed to create a maze.

3.9 Camera pathfinding

The task of the pathfinding algorithm is to produce a good waypoint to move the ball to; a bit closer to the destination than the current position. The camera produces an orthogonal grid of positions both for the ball position and the walls and holes, and this is used as a basis for the pathfinding graph where each and every pixel is connected to its eight neighbors but no others. There exists a map where every pixel is known to be traversable or not for the center of the ball, and both Dijkstra's algorithm and A* will produce a path from the current position of the ball to the destination. There exists one large problem for both of these algorithms in the physical world. The labyrinth will need to control the ball to every pixel in the path to the destination as the only connected positions are the adjacent pixels. The Theta* addition to the algorithm helps a lot, and with this addition it is possible to skip all pixel positions along the way as long as there exist a straight line free of obstacles between the current ball position and the next waypoint. By doing this we allow the control system of the PLC, the PID controller, to have larger authority over the ball trajectory.

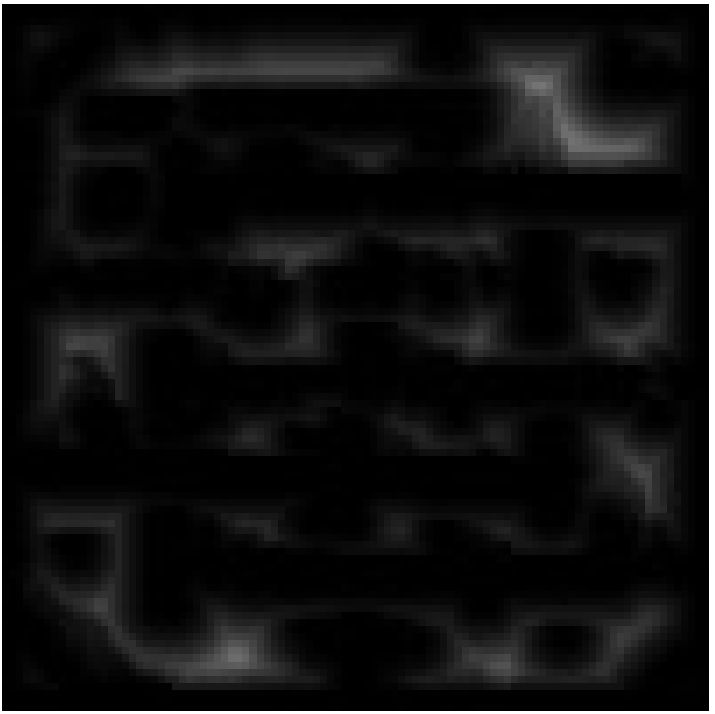


Figure 3.7: A gradient map of the maze where the distance from the nearest wall is reflected in the brightness of the pixel, the darkest areas correspond to a wall or the center of a hole.

The pathfinding algorithm is run once a second as a low-priority task. When it starts, the latest known ball position is taken as starting point and the destination point is given from the tablet application when the first search starts. The algorithm relies on two data structures:

- Map: the map with distance information from every wall and hole
- Searchspace: waypoints and cost from destination and start point

Both the map and the search space is a grid of 200x200 positions. Each search space position holds information about

- X,Y: this position
- previous: position closer to destination
- cost: accumulated cost to reach from destination
- projection: Manhattan cost to reach start position

When the Theta* algorithm is finished the parent pointers are used to locate the first waypoint after start, which as stated earlier may be several pixels away depending on the geometry of the walls in the vicinity. The waypoint is sent over the network to the PLC, and the thread sleeps for 1 second until a new search begins with a new current position.

The thread is interrupted several times during the calculation because the main thread has higher priority and as a consequence the calculation may take up to 0.4 seconds to complete its task. This has no influence on the functionality of the system as the ball will be controlled to the requested position by the PID controller of the PLC.

Cost calculations

The destination position is selected in the tablet and there is a possibility that the selected position will be to close to or inside a wall element. This will make the algorithm fail since there exists no traversable path all the way to the destination. To work around this, all positions inside the maze not intended as free positions are given a finite but high cost. A wall is given a cost of 10000 to be compared to a free position with a cost of 1, and all positions close to a wall are given a value inversely proportional to the distance from the wall

```
2*((Ball radius)-(Distance to wall))
```

The addition of these costs enables the algorithm to generate a path even if the destination is selected such that the ball is unable to reach it. It is important to select the costs so high that the algorithm only considers those positions as a viable means of reaching the destination after all other options has been exhausted.

3.10 Using the system

When the cabinet is powered on, the PLC, camera and wireless network is started. The startup process takes about 30 seconds. Before the labyrinth can be played, the playfield needs to be leveled. The PLC control panel on the cabinet is used to adjust the servo motors position in X and Y, and when the ball is stationary on the playfield the angles are set to 0.

The tablet consists of a main screen showing the camera image from the labyrinth. Three different play modes are available.

Angle: The angle of the playfield is controlled by tilting the tablet.

Position: The user selects a position on the camera image on the tablet, and the ball is moved in the shortest path to the position.

Auto: The user selects a position on the camera image on the tablet, and the ball is guided to the position by the system.

Figure 3.8: The Auto view of the tablet control software.

A second tablet can be used in the Angle mode giving the tablets control over one axis each.

Three plates are available for the game; two transparent labyrinth plates to be used in the angle mode and one opaque plate to be used in the auto or position mode.

Preparing the auto mode

When auto mode is selected for the first time, no map exists and the user needs to create the map. A suitable plate for the auto mode is placed on the labyrinth (the blank opaque plate), and the wall elements are placed on the plate to create a maze. When the maze is completed the button named “Create map” is selected. This generates a map that will be used for guiding the ball through the maze. The generation will only take a moment. It is important that the ball is not on the plate when generating the map. When the map is generated the ball may be placed on the plate, and a destination may be selected.

At any time it is possible to regenerate the map by pressing the button, as long as the ball is not present during the regeneration.

4

Discussion and results

4.1 Camera performance

The positioning functionality of the camera is providing a positioning accuracy with a resolution of $71/200 = 0.36cm$ at a repetition rate of 20 Hz/50 ms. The accuracy of the positioning was measured as $\pm 1 pixel = \pm 0.36cm$.

A setting with a higher frame-rate was also tested, which generated the following resolution: $55/200 = 0.28cm$ at 30Hz/33 ms with a accuracy of $\pm 0.28cm$

The pathfinding functionality of the camera is providing the same resolution, 3,6/2.8 mm at a repetition rate of 1 Hz with a latency of < 0.4 seconds.

The camera processor load has been measured with a total maximum load of 80% when both positioning and pathfinding were running. Out of this, 45% processor load was camera internal processing, which is a result of the selected camera capture size and not possible to improve from the user software side, and 35% was the load caused by positioning and pathfinding.

The largest limitation for the control loop is the repetition rate of the position. By lowering the capture area of the camera and as a result not capturing the full area of the playfield, it is possible to increase the frame-rate of the camera and improve the stability of the control loop, increasing both the speed and minimizing the overshoot. The limitation on the frame-rate is governed by the camera. The algorithms calculating the position are no bottleneck here.

The placement of the camera inside the cabinet has eliminated any adjustments and calibrations needed for the previous setup. The illumination of the playfield need to be good, otherwize the camera will adjust the exposure time in order to compensate for the lower lightning conditions and decrease the framerate wich will affect the control performance of the labyrinth.

4.2 Control performance

The positioning accuracy of the ball was not fully satisfactory at 20 Hz. In response to a step input, the ball will perform an overshoot when moving to the new position.

The PID parameters were not possible to adjust to completely avoid an overshoot at a step response. The reasons for this was that, in order to overcome the static friction of the ball and get it to roll, the proportional constant K needs to be large enough. Due to the slow update rate, such a large value, however, leads to oscillations and self resonance of the system.

A noticeable decrease of the system's tendency for self resonance was observed when the camera repetition rate increased from 16 to 20 Hz, which in turn improved the control of the ball.

An increase up to 30 Hz improved the control considerably. At this rate the positioning accuracy was good with minimal overshoot and good control even when the ball was rolling fast.

The final parameters of the control-loop depend on the physical system, but the relative amplitudes can be compared here (the values are related to the settings for 30 Hz):

$$\begin{aligned} K &= 0.07 \\ T_i &= 5 \\ T_d &= 1.5 \\ N &= 20 \\ T_T &= 71.4 \\ \beta &= 1 \end{aligned}$$

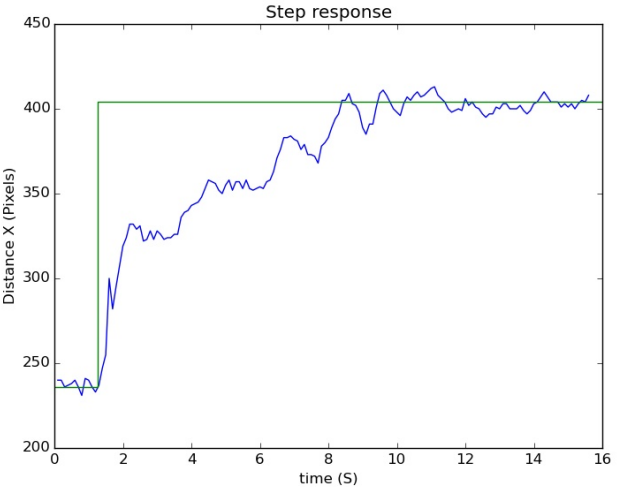
The values were obtained using manual tuning. The β value was set to 1 as the major disturbance to the system is a change in reference values. The low pass filtering (N) of the derivate part was given a rather high value since no large noise source exists in the position measurements.

4.3 System comparison

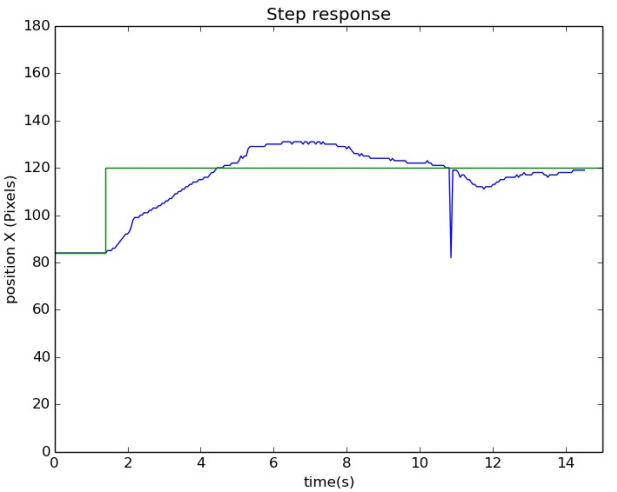
Step response tests were performed, on the original system with a camera mounted above the maze and with the camera mounted inside the maze, with an update frequency of both 20 and 30 Hz. The result is shown in Figure 4.1.

The step response of the 30 Hz system in Figure 4.1c is the fastest, even though the old system with a response in Figure 4.1a has a similar update frequency. The old system has a higher positioning variance and to counteract the larger variance a filter exists that reduces the speed of the control. The tuning of the PID parameters for the 30 hz system is not ideal, the derivate response is to high and need to be lowered for even better control.

The step response of the 20 Hz system is the only system showing an overshoot. This is mainly due to the fact that in order to avoid oscillations the proportional

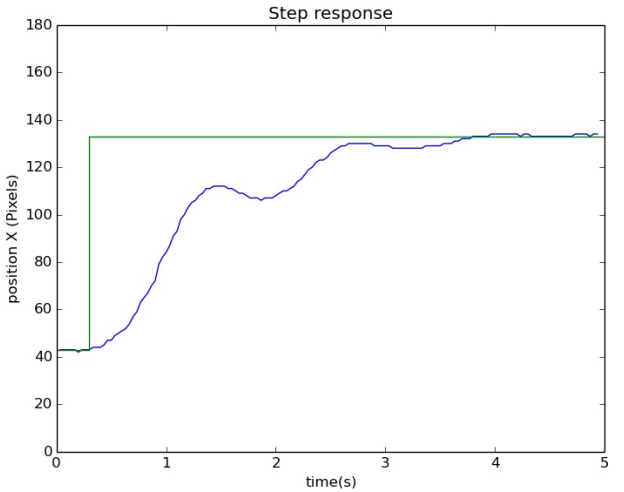


(a) *Above board*



(b) *Below board*@20Hz

Figure 4.1: Step response comparison between the current and previous system



(c) *Below board*@30Hz

Figure 4.1: Step response comparison between the current and previous system

value has to be so low that the main influence on the output signal at low error levels is the integral part.

The low noise in the positioning and the 30 Hz update frequency is a significant improvement of the control performance. A single specle of noise can be seen in Figure 4.1b.

The graphs of the new system (Figure 4.1b and 4.1c) were created by recording the positioning information generated by the camera and as a result the graphs represent the exact poisitioning information generated by the algorithms. When recording the data from the old system, Figure 4.1a, similar instrumentation was not available and the ball positioning was generated by calculating the position from a previous camera recording of the step response. The scales between the two systems differ as a result of the different methods of gathering the data.

4.4 Ball detection

The Hough transform is very accurate in determining the ball position. The error was determined to be $\pm 1 pixel$ in the 200x200 pixel image. The positioning error was determined by recording the maximum and minimum position detected by the camera at different locations of the playfield and calculating the maximum error. The ball is reliably detected and picked out among other objects within the image. Problems in detecting the ball occurs if the ball is in front of a surface with similar

color or texture, as well as if round objects with the approximate same size are in the camera image. This will position the ball at the wrong place. One such glitch can be seen in Figure 4.1b. With a slight tuning of the threshold value for the ball color, no such errors have been detected after this when using the opaque plate.

The ball is allowed to be located everywhere in the playfield and it is thus possible to lift and place the ball at a different position. The first trials were performed with a transparent maze in order to show the internal electric components of the maze, but the problems where a bystander with the wrong color of clothes might disturb the detection resulted in the use of a white semi-transparent background of the maze, where the shadow of the ball was detected by the camera instead. The position of the ball depends on locating the maximum value in the accumulator space. By increasing the resolution to more than the used 200x200 pixels it is possible to get a higher value in the accumulator since more pixels along the edge of the ball contribute to the accumulator. This improvement makes it possible to detect the ball in an environmen where the edge gradients is weaker such as a transparent playfield. If instead a lower resolution is used the accumulator value of the ball will be lower and the difference to other objects in the playfield such as sharp corners will be to low to reliably find the ball in the playfield.

4.5 Path generation

The solution for the path generation works very well and provides a minimum number of waypoints. The algorithm always generates a traversable path when a connection between start and destination exists. The generation of new waypoints is not time critical since the physical control over the ball is at least an order of magnitude slower. An increased resolution will increase the pathfinding time but not to an extent where the pathfinding will be the performance bottleneck. The cost functionality in the algorithm (where the cost for entering the area close to the wall is high) enables the algorithm to generate the best path from start to end even if the selected destination (or ball position) is to close to the wall. In this case the ball will be resting against the wall as close as possible to the destination.

4.6 Future improvements

Camera

The performance of the camera is a major issue, with a tradeoff between capturing the full playfield area and good control of the ball. To improve the situation a camera with a higher frame-rate capable of capturing the entire playfield is needed. One interesting thing to notice is that the resolution of the camera is of almost no importance, considering the used resolution for the hough transform of 200x200 pixels and the perceived camera resolution of 320x240 pixels. It is possible to use a

camera with a low resolution, a high frame-rate and a good, exchangeable fish-eye lens system to capture the entire playfield. As an alternative route, more than one camera can be used cooperating by capturing parts of the playfield.

Ball control

The current control utilizes PID control to position the ball. Several alternative ways of controlling the ball exists like using a state feedback from an observer. A problem not investigated in this thesis is how the control algorithm should behave when the ball is stopped by a wall.

Ball detection

The detection of the ball works well and detects the ball at a correct and precise location. A future improvement is to enhance the detection in such a way that the ball is accurately detected even if the background is changing. This would enable the playfield to be totally transparent for ball detection. Improvements of the algorithms would increase the load on the camera processor and any substantial addition to the algorithms would require more processing power than the current camera has.

Map and path generation

The path generation works very well. The map is explicitly generated by the user, a future improvement may be to remove the manual step and generate a map continuously from the camera images. Some means of ignoring the ball during the map generation step is then needed in the software.

Tablet

The software in the tablet was not a primary goal of this thesis. The software is working, but in a demonstrator there always exists room for improvements in the look and feel of the software, with nice looking transitions between different play modes, animated graphics and other gizmos.

Bibliography

Bresenham, J. (1965). “Algorithm for computer control of a digital plotter”. *IBM systems Journal* **4**:6.

Brown, D. C. (1971). “Close-range camera calibration”. *Photogrammetric Engineering* **37**, pp. 855–866.

Canny, J. (1986). “A computational approach to edge detection”. *IEEE Transactions on pattern analysis and machine intelligence* **8**:6.

Chang, Y.-H., W.-S. Chan, and C.-W. Chang (2013). “T-s fuzzy model-based adaptive dynamic surface control for ball and beam system”. *IEEE Transactions on industrial electronics* **60**:6.

Daniel, K., A. Nash, S. Koenig, and A. Felner (2010). “Theta*: any-angle path planning on grids”. *Journal of Artificial Intelligence Research* **39**.

Dhane, P., K. Kutty, and S. Bangadkar (2012). “A generic non-linear method for fisheye correction”. *International Journal of Computer Applications (0975 – 8887)* **51**.

Espersson, M. (2010). *Vision Algorithms for ball on beam and plate*. MSc thesis. Department of automatic control, Lund university.

Galvan-Colmenares, S., M. A. Moreno-Armendáriz, J. de Jesús Rubio, F. Ortíz-Rodriguez, W. Yu, and C. F. Aguilar-Ibáñez (2014). “Dual pd control regulation with nonlinear compensation for a ball and plate system”. *Mathematical Problems in Engineering* Article id:894209.

Hart, P. E., N. J. Nilsson, and B. Raphael (1968). “A formal basis for the heuristic determination of minimal cost paths”. *IEEE Transactions of system science and cybernetics* **4**:2.

Kingston, J. H. (1995). *Algorithms and data structures Design, Correctness, Analysis*. Addison wesley.

Persson, D. and F. Wadman (2014). *Self-playing Labyrinth game Using camera and industrial control system*. MSc thesis. Department of automatic control, Lund university.

Shah, S. and J. K. Aggarwahl (1996). “Intrinsic parameter calibration procedure for a (high-distortion) fish-eye lens camera with distortion model and accuracy estimation*”. *Pattern Recognition* **29**:11, pp. 1775–1778.

Shih Frank Y.and Wu, Y.-T. (2004). “Close-range camera calibration”. *Computer Vision and Image Understanding* **93**, pp. 195–205.

Virseda, M. (2004). *Modeling and Control of the Ball and Beam Process*. MSc thesis. Department of automatic control, Lund university.

Weiss, J. M. (2009). “Real-time feature detection using the hough transform”. In: *Proceedings of Midwest Instruction and Computing symposium 2009 (MICS), Rapid city, USA*.

Årzén, K.-E. (2012). *Real-Time Control Systems*. Department of automatic control, Lund university.

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER´S THESIS

Date of issue
March 2015

Document Number
ISRN LUTFD2/TFRT--5964--SE

Author(s)
Stefan Olsson

Supervisor
Erik Lindmark, Alten AB
Anton Cervin, Dept. of Automatic Control, Lund University, Sweden
Karl-Erik Årzén, Dept. of Automatic Control, Lund University, Sweden (examiner)

Sponsoring organization

Title and subtitle

Pathfinding and positioning in a labyrinth game using a wide-angle camera

Abstract

Alten AB has a technology demonstator in the form of a motorized and camera equipped large scale labyrinth game. The ball position is controlled by a ABB industrial PLC connected with Android tablets for user interface and a camera as a sensor for the ball position.

This thesis demonstrates the ability to place a wide angle camera inside the cabinet, correcting the lens distortion caused by the wide angle lens and detect the ball with the use of a circular Hough transform. A path is also generated from the ball position to any position of the maze by capturing an image from the camera, generating a map for subsequent pathfinding, using an improvement of the Dijkstra's pathfinding algorithm named Theta*. It further demonstrates the feasibility of using the computing power of the camera for both pathfinding and ball positioning.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-49

Recipient's notes

Security classification