TARGET TRACKING THROUGH IMAGE PROCESSING

Submitted by

CAPT ZAHEER HAIDER	TE-36
CAPT HAFEEZ ULLAH	TE-36
CAPT AFAQ ALI	TE-36
LT NAVEED IMTIAZ	TE-36
LT NADEEM SAEED	TE-36

Supervised by
LT COL TANVEER GOHAR

National University Of Sciences and Technology
Department of Electrical Engineering
Military College of Signals
RAWALPINDI

2000-2003

A project report submitted to the National University of Sciences and Technology Rawalpindi, in partial fulfillment of the requirements for the award of the degree of Bachelor of Electrical Engineering (TeleCom)

In the name of Allah,
Most Gracious,
Most Merciful

National University of Sciences and Technology
Rawalpindi

Final Approval

This is to certify that we have examined the report submitted by capt Zaheer Haider, Capt Hafeez Ullah, Capt Afaq Ali, Lt Naveed Imtiaz and Lt Nadeem Saeed in our judgment it is of sufficient standard to warrant its acceptance by theNational University of Sciences and Technology Rawalpindi, for the award of the degree of Bachelor of Electrical Engineering (TeleCom).

Committee**:**

Dr Nauman Jafri	____________________________
Wng Cdr Akhlaq Ali	____________________________
Lec Irtiza	____________________________

Project Supervisor
Lt Col Tanveer Gohar ____________________________

Project Brief

Project Title	Target Tracking through Image Processing
Objective:	Providing a working prototype of a locomotive that can see its target through Web Camera and can hit its target
Submitted by:	CAPT ZAHEER HAIDER TE-36 CAPT HAFEEZ ULLAH TE-36 CAPT AFAQ ALI TE-36 LT NAVEED IMTIAZ TE-36 LT NADEEM SAEED TE-36
Supervised by:	Lt Col Tanveer Gohar
Duration:	Dec to April 2003
Operating System:	Microsoft Windows 98/NT/2000/XP
Hardware:	Intel Celeron 500MHz RAM 192MB

CONTENTS AT A GLANCE

Table of Contents

List of Figures

List of Tables

Chapter 1 : INTRODUCTION

1 Introduction

The product is the core artefact of the software development process. In this section we will demonstrate the product. It includes product features, constraints, and operational domain.

1.1 Goals And Objectives

The main idea is to develop a control system, which is fault tolerant and self-healing. It seeks a target, one that it has recognized. It is based on the techniques for image processing.

The main objective is to develop a control-system that takes input from digital camera and then recognizes the object as its target and instructs to hit it by the car attached to it. The product is defined as:

"The end product is a software-intensive control system which can be viewed as an imaging based intrusion detector."

1.2 System Statement of Scope

1.2.1 General Requirements

The main requirement is to develop a software system that seeks a moving target. It is guided through a CCD camera that captures the image from camera. Then the controlling computer (Control Station) uses image recognition techniques to identify the target and then instructs the remotely controlled car (the locomotive) to hit that target.

.

.

1.3 Project Class

The project falls into the category of Concept Development Projects.

1.4 System Context

Multiple processes are under real-time (Kernel Mode) execution. So, concurrency must be built into system such that the overall response must be real-time or as close to it as possible.

1.5 Mjor Contstraints

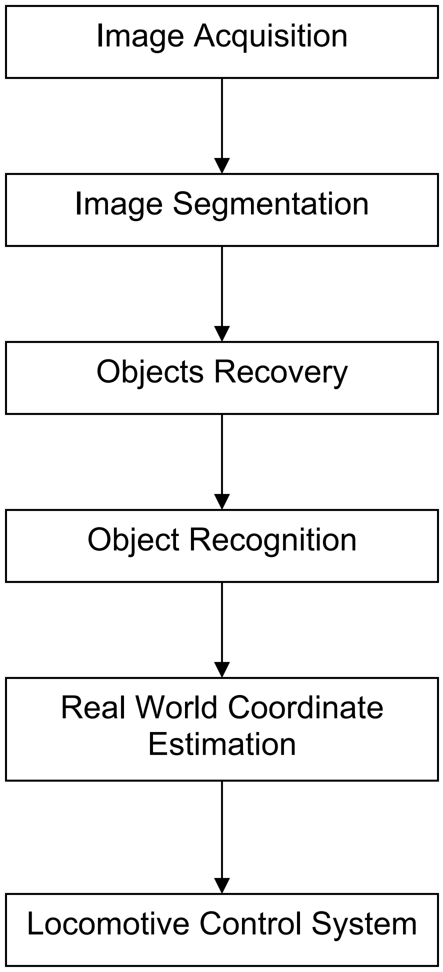
Time

The development team has completed the project in 5 months. It includes total time for developing and deploying the system.

Part I: Image Processing Module

The System Defined

The image processing pipeline is showing all the operations performed for locating the object of interest in the real world.



Problem Statement

The basic aim is to recognize the object of interest (i.e. target) and then estimating its real world coordinates. Estimated real world coordinates serves as an input to locomotive used in order to hit the recognized object.

Chapter 2: Image Segmentation

2.0 Image Segmentation

There are different methods for image segmentation in order to extract different objects. The extracted objects are further analyzed to get the desired results. In our domain the results might be extracting different objects those have some specific colour. Once the colour based extracted objects has been located, then they are further processed. The image is acquired in BMP format which uses the RGB colour space to represent the colour of each pixel. Then the colour space is converted to HSV colour model for colour based segmentation. The reasons for converting the colour space from RGB to HSV are described after understanding the basic concepts of Colours.

2.1 Color Fundamentals

Although human colour acquisition as a psychological phenomenon is not yet fully understood we know that it is based on the nature of the reflected light coming from environment objects.

When dealing with achromatic light the intensity is the only perceivable attribute, resulting in Gray level images.

Chromatic light wavelength spans the EM energy spectrum from roughly 400 to 700 nm. The quality of a chromatic light source can be expressed in terms of:

- Radiance, measured in watts (W) related to the energetic power of the light,
- Luminance, measured in lumens (lm) related to the perceived energy of the light and
- Brightness, a more ambiguous notion, impossible to measure.

A colour is defined from human eye capabilities. If you consider a normal human being, his vision of a colour will be the same as for another normal being. Of course, to show any collared information, you need a definition (or a model, to use the right word). There are two kinds of colour definitions:

- The device-dependent: These definitions are more or less accurate.

It means that when you display on such a device one color with the particular definition, you get a rendering but when you display on an other device the same colour, you get another rendering (more or less dramatically different).

- The device-independent: This means that the model is accurate and you must adjust your output device to get the same answer. This model is based on some institute works (curves of colours and associated values). From an absolute point of view, it means from a human visual sensation, a colour could be defined by:
- Hue: The perception of the nuance. It is the perception of what you see in a rainbow.
- Colourfulness: The perception of saturation, vividness, purity of a colour. You can go from a sky blue to a deep blue by changing this component.
- Luminance: The perception of an area to exhibit more or less light. It is also called brightness. You can blurry or enhance an image by modifying this component.

As you see above, I describe a colour with three parameters. All the students of math are quickly going to say that the easier representation of this stuff is a space, a tri-dimensional space with the previous presentation. And I totally agree with that. That is why we often call 'colour space' a particular model of colours.

2.1.1 Image Based on Color Lookup Tables

All of the pictures don't use the full colour space. That's why we often use another scheme to improve the encoding of the picture (especially to get a file which takes less space). To do so, you have two possibilities:

- You reduce the bits/sample. It means you use fewer bits for each component that describe the colour. The colours are described as direct colours, it means that all the pixels (or vectors, for vectorial descriptions) are directly stored with the full components. For example, with a RGB (see item 5.1 to know what RGB is) bitmapped image with a width of 5 pixels and a height of 8 pixels, you have:

(R11, G11, B11) (R12, G12, B12) (R13, G13, B13) (R14, G14, B14) (R15, G15, B15)
(R21, G21, B21) (R22, G22, B22) (R23, G23, B23) (R24, G24, B24) (R25, G25, B25)
(R31, G31, B31) (R32, G32, B32) (R33, G33, B33) (R34, G34, B34) (R35, G35, B35)
(R41, G41, B41) (R42, G42, B42) (R43, G43, B43) (R44, G44, B44) (R45, G45, B45)
(R51, G51, B51) (R52, G52, B52) (R53, G53, B53) (R54, G54, B54) (R55, G55, B55)
(R61, G61, B61) (R62, G62, B62) (R63, G63, B63) (R64, G64, B64) (R65, G65, B65)
(R71, G71, B71) (R72, G72, B72) (R73, G73, B73) (R74, G74, B74) (R75, G75, B75)
(R81, G81, B81) (R82, G82, B82) (R83, G83, B83) (R84, G84, B84) (R85, G85, B85)

Where Ryx, Gyx, Byx are respectively the Red, Green, and Blue components you need to render a colour for the pixel located at (x, y).

- You use a palette. In this case, all the colours are stored in a table called a palette and the components of all the colours for each pixel (or the vector data) are removed to be replaced with a number. This number is an index in the palette. It explains why we call the palette, a colour look-up table.

2.1.2 Color Spaces

A colour space is a model for representing colour in terms of intensity values; a colour space specifies how colour information is represented. It defines a one-, two-, three-, or four-dimensional space whose dimensions, or **components,** represent intensity values. A colour component is also referred to as a **colour channel**. For example, RGB space is a three-dimensional colour space whose components are the red, green, and blue intensities that make up a given colour.

Visually, these spaces are often represented by various solid shapes, such as cubes, cones, or polyhedron.

2.1.3 Uniform Color Space

A colour space in which equivalent numerical differences represent equivalent visual differences, regardless of location within the colour space. A truly uniform colour space has been the goal of colour scientists for many years. Most colour spaces, though not perfectly uniform, are referred to as uniform colour spaces, since they are more nearly uniform when compared to the chromaticity diagram.

2.1.4 RGB Color Model

In the RGB model, an image consists of three independent image planes, one in each of the primary colours: red, green and blue. (The standard wavelengths for the three primaries are as shown in figure 1). Specifying a particular colour is by specifying the amount of each of the primary components present. Figure 1 shows the geometry of the RGB colour model for specifying colours using a Cartesian coordinate system. The grayscale spectrum, i.e. those colours made from equal amounts of each primary, lies on the line joining the black and white vertices.

Figure 1: The RGB colour cube. The grayscale spectrum lies on the line joining the black and white vertices.

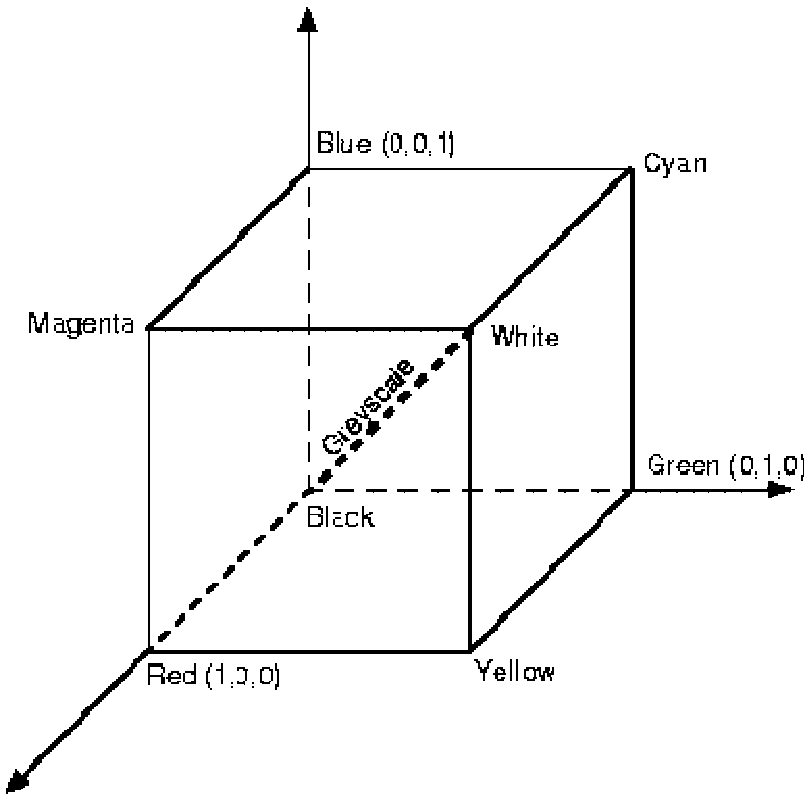
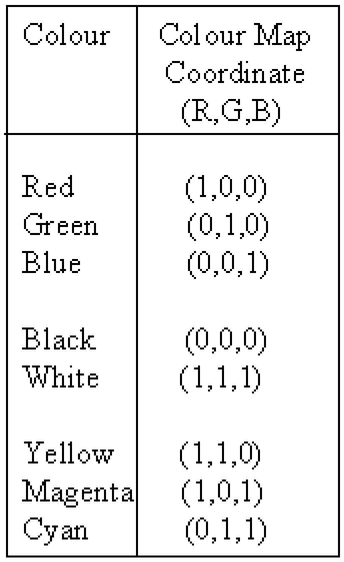


Figure 1 *RGB Color Model*

This is an *additive* model, i.e. the colours present in the light add to form new colours, and is appropriate for the mixing of coloured light for example. The RGB model is used for colour monitors and most video cameras.
The RGB representation model is based on the usage of the so-called “primary” colours:

- Red (700 nm)
- Green (546.1 nm)
- Blue (435.8 nm)

This does not mean that any colour can be obtained by varying the intensity of each
component.



Colour	Colour Map Coordinate (R,G,B)
Red	(1,0,0)
Green	(0,1,0)
Blue	(0,0,1)
Black	(0,0,0)
White	(1,1,1)
Yellow	(1,1,0)
Magenta	(1,0,1)
Cyan	(0,1,1)

Table 1 *RGB Color Map Coordinates*

2.1.5 HSV Color Model

Hue, Saturation, and Value colour space was chosen because it is the only main non-linear transformation from RGB and because it is usually considered the easiest to use for colour based segmentation.

The HSV colour space, like RGB, is a device-dependent colour space, meaning the actual colour you see on your monitor depends on what kind of monitor you are using, and what its settings are.
The HSV Colour Model stands for the Hue, Saturation, and Value based on Tint, Shade, and Tone.

The coordinate system in a hexagon.

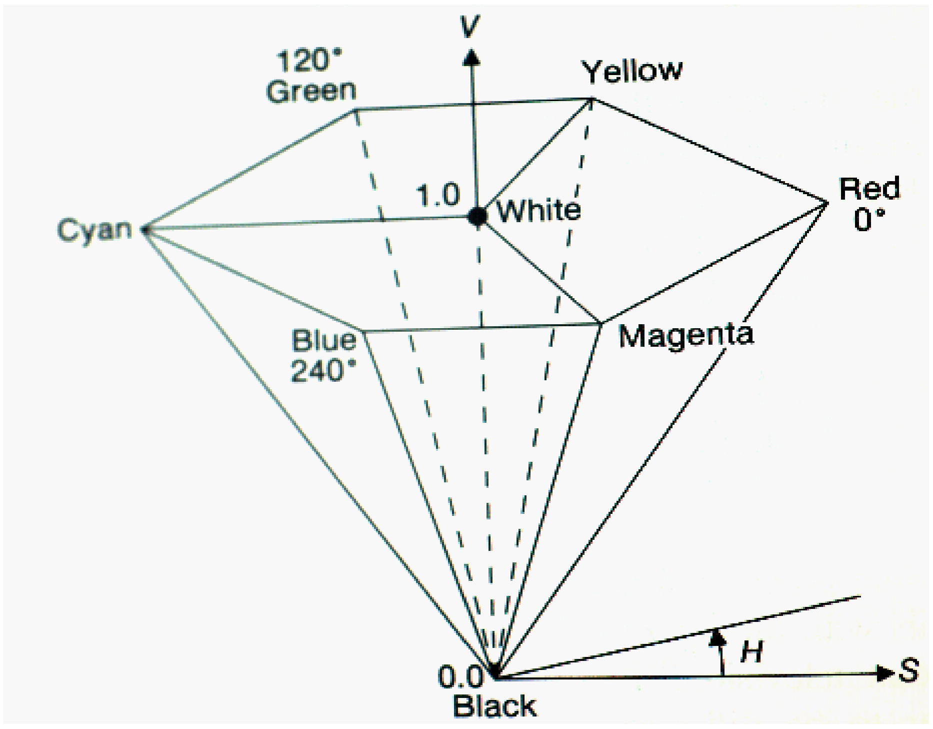


Figure 2 *HSV Color Model*

Here is a view of the HSV colour model from the side. The axis has been rotated 180 degrees so that the magenta is pointing to the left.

2.1.6 Some Interesting Things to Notice

The top of the HSV hexagon corresponds to the projection one would see by looking down the principal diagonal of the RGB cube (from white [1, 1, 1], towards black [0, 0, 0]). Thus, the main diagonal of the RGB cube corresponds to the V parameter of the HSV space. Therefore, you will notice if you choose S to be 0 (and H becomes undefined), you are on the V-axis of the HSV hexagon, which corresponds to the gray-scale main diagonal of the RGB cube, so changing V simply amounts to moving along this diagonal, producing all the gray-scales. (If you have the RGB applet running next to it, you will notice this even better.)

The pure colours are at S=1, V=1 (i.e. the very top slice of the hexagon). Decreasing S without changing V corresponds to adding white pigment to a colour, whereas keeping S=1 while changing V corresponds to creating a shade of a colour. (Again, if you have the RGB applet running next to it, you will notice this even better.)

2.1.7 Why Do We Need to Convert Color Space

It has been mentioned before that RGB colour space uses the three basic colours to represent any colour and its attributes (i.e. Brightness and saturation). Thus any change in the brightness or saturation is accomplished by changing the primary colours. While considering the HSV colour space; variations in brightness and saturation are accomplished by changing the respective components not colour itself.

2.2 Operations Performed for Image Segmentation

There are different binary operations performed for image segmentation. These binary operations are explained.

2.2.1 Input/Output of Segmentation

The input to this phase is the image acquired and the output is also an image but having those objects which have met the criteria as explained in the algorithm below.

Input Image (BMP format, RGB Colour Space)

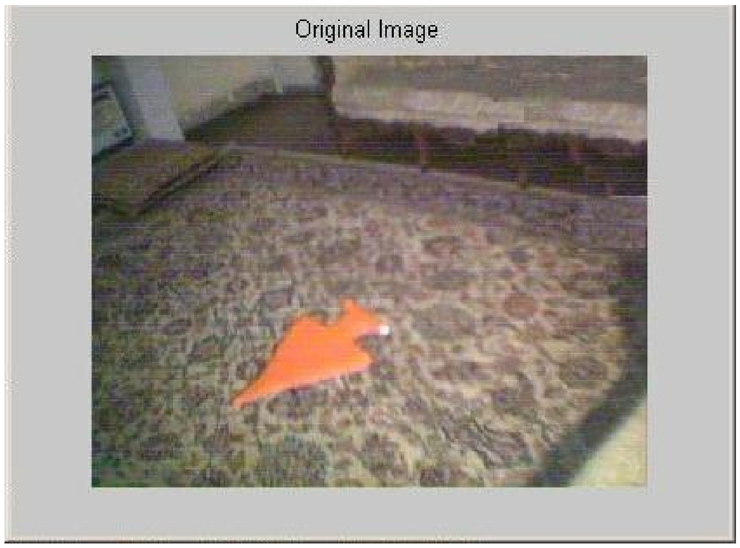


Figure 3 *Sample Image for Image Processing Operation*

2.2.2 RGB to HSV Color Space Conversion

$$
\boxed{
\begin{array}{l}
V = \max(R, G, B) \\
S = (I - \min(R, G, B)) / 255 \\
H = 0 + (G-B)/\Delta \quad \text{if max is R} \\
\quad = 1/3 + (B-R)/\Delta \quad \text{if max is G} \\
\quad = 2/3 + (R-G)/\Delta \quad \text{if max is B} \\
(\Delta \text{ is (max-min) of RGB}) \\
\\
I=(R+G+B) / 3
\end{array}
}
$$

The image is converted from RGB colour space to HSV. RGB is a linear colour space so it is converted to a uniform colour (i.e. HSV)

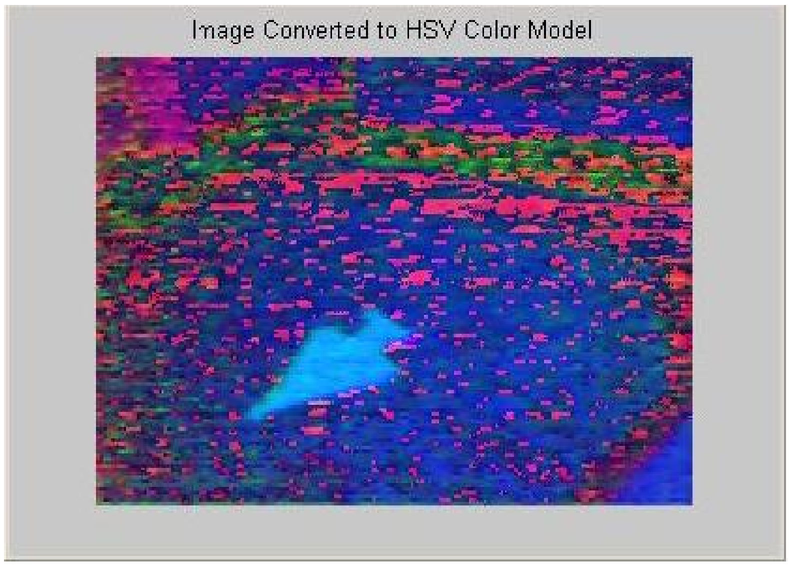


Figure 4 *Image after Coversion from RGB to HSV Color Model*

The image looks different from the original image because the monitors implement the RGB colour space while manufacturing.

2.2.3 Algorithm for Color Based Segmentation

The algorithm is

```
Scan each pixel in the image
    Check whether the pixel lies within a specific region of hue and saturation
    If yes
     then do not change the pixel value
    If no
        Then change the pixel into black.
```

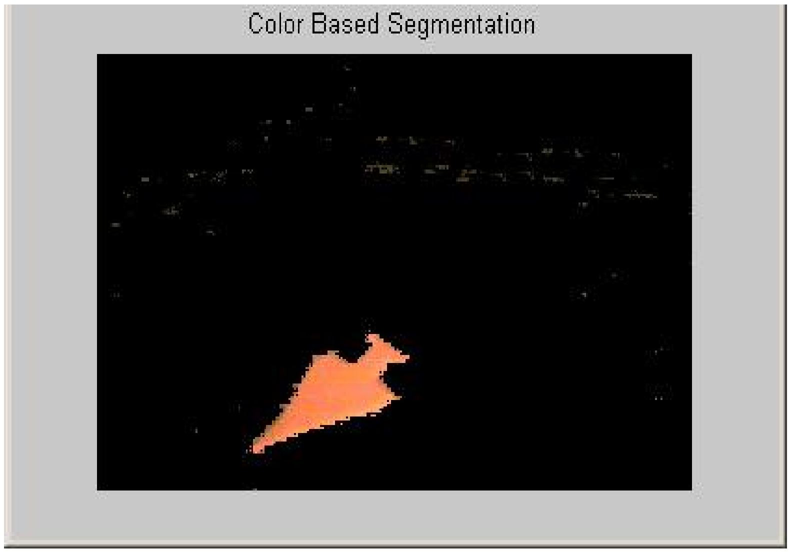


Figure 5 *Color Based Segmentation*

Objects are being segmented using colour. In our case the colour we have selected is close to red including its different variations. Figure 5 shows the objects with specific colours (i.e. red).

Chapter 3: Objects Recovery

3 Objects Recovery

There are different factors involved which distorts the original image and the image is not perceived properly as the camera should perceive them. So to recover the objects in their original shape or to connect the disconnected components of an object, we used Dilation and Erosion as explained below.

3.1 Object Recovery Using Connected Component

Since the whole task of image processing (starting from image acquisition down to finding the real world coordinates of the desired object) is in a pipeline. Thus the output of the previous process is input to the next. Thus the segmented image based on colour is input to the process under discussion.
While working in this phase different image processing techniques have been used. It is better to have a brief view of these techniques. Then I would like to explain their use in our context.

3.2 Binary Image Operations (An Overview)

A binary image is an image in which each pixel assumes one of only two discrete values. Essentially, these two values correspond to on and off. Looking at an image in this way makes it easier to distinguish structural features. For example, in a binary image, it is easy to distinguish objects from the background.
A binary image is stored as a two-dimensional matrix of 0's (which represent off pixels) and 1's (which represent on pixels). The on pixels are the foreground of the image, and the off pixels are the background.
Binary image operations return information about the form or structure of binary images only. To perform these operations on another type of image, you must first convert it to binary. Figure 6 shows the collared image converted to binary image.

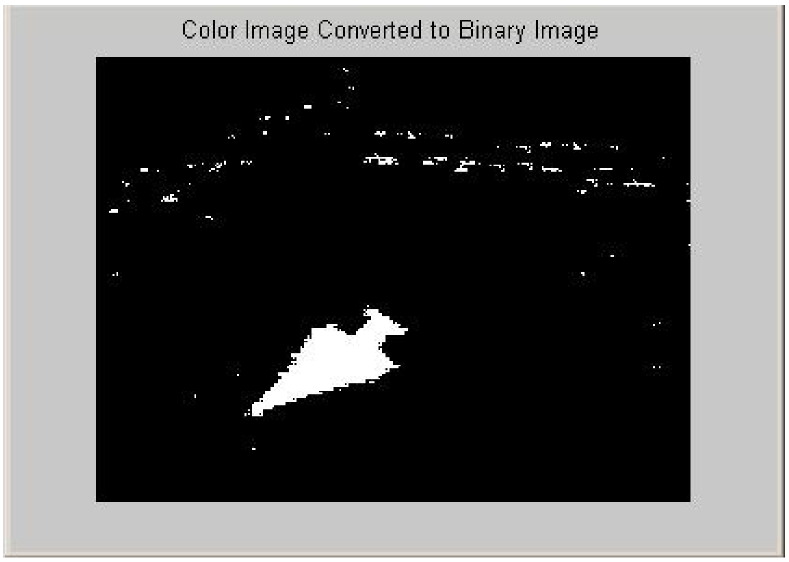


Figure 6 Color Image Converted to Binary Image

An overview of the following terms will help you to understand methods and algorithms presented in this report.

Words	**Definitions**
Background	The set of black (or off) pixels in a binary object.
Binary image	An image containing only black and white pixels. A binary image is represented by a matrix containing 0's and 1's only.
Connected component	A set of white pixels that form a connected group. A connected component is "8-connected" if diagonally adjacent pixels are considered to be touching; otherwise, it is "4-connected."
Foreground	The set of white (or on) pixels in a binary object.
Morphology	A broad set of binary image operations that process images based on shapes. Morphological operations apply a structuring element to an input image, creating an output image of the same size. The most basic morphological operations are dilation and erosion.

Neighbourhood	A set of pixels that are defined by their locations relative to the pixel of interest. In binary image operations, a neighbourhood can be defined by a structuring element or by using the criterion for a 4- or 8-connected neighbourhood.
Object	A set of white pixels that form a connected group. In the context of this chapter, "object" and "connected component" are basically equivalent. See "Connected component" above.
Structuring element	A matrix used to define a neighbourhood shape and size for binary image operations, including dilation and erosion. It consists of only 0's and 1's and can have an arbitrary shape and size. The pixels with values of 1 define the neighbourhood. By choosing a proper structuring element shape, you can construct a morphological operation that is sensitive to specific shapes.

Table-2

3.2.1 Clean Operation

'Clean Operation' removes *isolated* pixels (individual 1's that are surrounded by 0's), such as the centre pixel in this pattern.

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

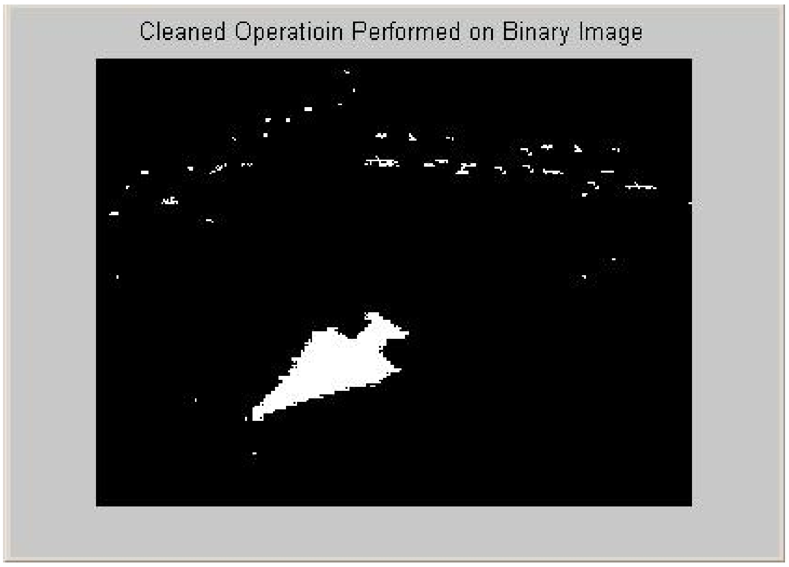


Figure 7 *Cleaned Operation Performed on Binary Image*

3.2.2 Dilation and Erosion

The main morphological operations are *dilation* and *erosion.* Dilation and erosion are related operations, although they produce very different results. Dilation adds pixels to the boundaries of objects (i.e., changes them from off to on), while erosion removes pixels on object boundaries (changes them from on to off).

Each dilation or erosion operation uses a specified neighbourhood. The state of any given pixel in the output image is determined by applying a rule to the neighbourhood of the corresponding pixel in the input image. The rule used defines the operation as dilation or erosion.

- For dilation, if *any* pixel in the input pixel's neighbourhood is on, the output pixel is on. Otherwise, the output pixel is off.
- For erosion, if *every* pixel in the input pixel's neighbourhood is on, the output pixel is on. Otherwise, the output pixel is off.

The neighbourhood for dilation or erosion operation can be of arbitrary shape and size. The neighbourhood is represented by a *structuring element*, which is a matrix consisting of only 0's and 1's. The *centre pixel* in the structuring element represents the pixel of interest, while the elements in the matrix that are on (i.e., = 1) define the neighbourhood.

The centre pixel is defined as floor ((size (SE) +1)/2), where SE is the structuring element. For example, in a 4-by-7 structuring element, the centre pixel is (2, 4). When you construct the structuring element, you should make sure that the pixel of interest is actually the centre pixel. You can do this by adding rows or columns of 0's, if necessary. For example, suppose you want the neighbourhood to consist of a 3-by-3 block of pixels, with the pixel of interest in the upper-left corner of the block. The structuring element would not be ones (3), because this matrix has the wrong centre pixel. Rather, you could use this matrix as the structuring element.

$$\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{array}$$

For erosion, the neighbourhood consists of the on pixels in the structuring element. For dilation, the neighbourhood consists of the on pixels in the structuring element rotated 180 degrees. (The centre pixel is still selected before the rotation.)

Suppose you want to perform an erosion operation. Figure shows a sample neighbourhood you might use. Each neighbourhood pixel is indicated by an x, and the centre pixel is the one with a circle.

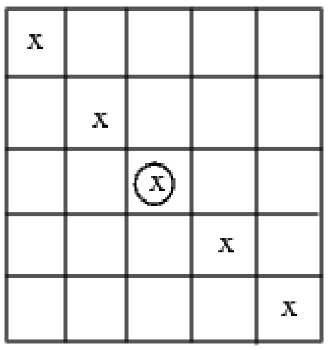


Figure 8: A Neighbourhood that will be represented as a structuring element

The structuring element is therefore

$$\begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{matrix}$$

The state (i.e., on or off) of any given pixel in the output image is determined by applying the erosion rule to the neighbourhood pixels for the corresponding pixel in the input image. For example, to determine the state of the pixel (4, 6) in the output image:

- Overlay the structuring element on the input image, with the centre pixel of the structuring element covering the pixel (4, 6).
- Look at the pixels in the neighbourhood of the input pixel. These are the five pixels covered by 1's in the structuring element. In this case the pixels are: (2,4), (3,5), (4,6), (5,7), (6,8). If all of these pixels are on, then set the pixel in the output image (4, 6) to on. If any of these pixels is off, then set the pixel (4, 6) in the output image to off.

You perform this procedure for each pixel in the input image to determine the state of each corresponding pixel in the output image.

3.2.3 Structuring Element Applied for Erosion

It is a one by one matrix for erosion. (i.e. 1 only). The output of the erosion operation has been shown in Figure 9.

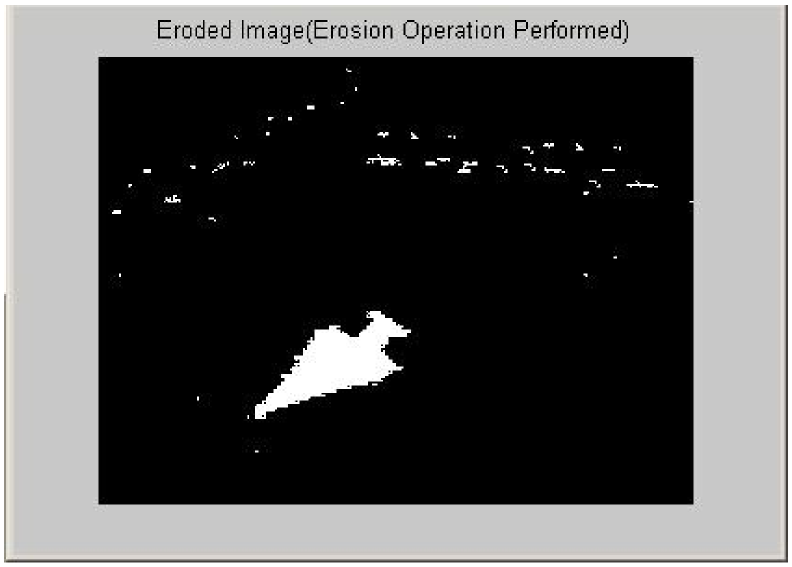


Figure 9 *Erosion Operation Performed on Binary Image*

3.2.4 Structuring Element Applied for Dilation

$$\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array}$$

Figure 10 is the result of the dilated image. The SE used in dilation is a 3*3 matrix shown above

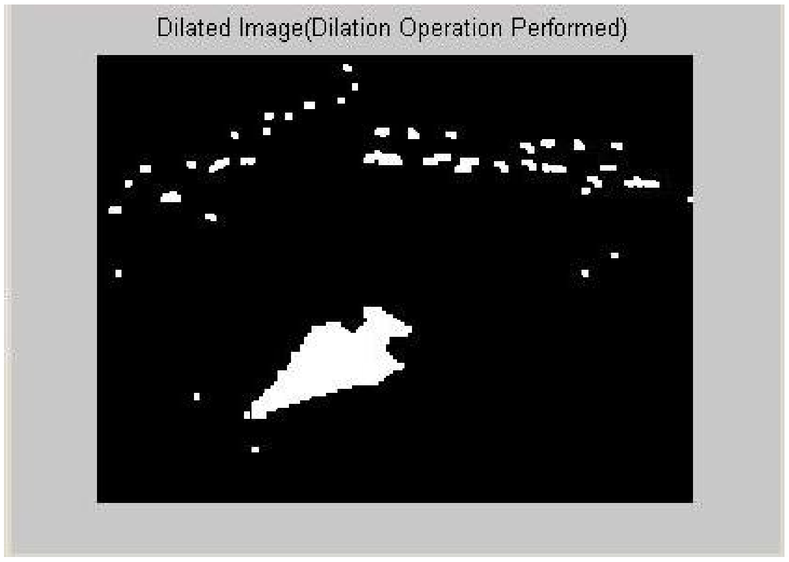


Figure 10 *Dilation Operation Performed on Binary Image*

3.3 Perimeter Determination

In order to find the skeleton of objects the perimeter is being determined which serves the purpose for identification in later stages.
In order to recognize different objects we do require the outline of the objects. To find the outline of the objects there are different approaches like connected components, Edge Detection algorithm (also used for image segmentation). There are different approaches for perimeter determination like edge detection (also used for image segmentation) and connected components. I have used the method of connected components for this purpose

3.3.1 4 and 8 Connected Neighbourhoods

For many operations, distinguishing objects depends on the convention used to decide whether pixels are connected. There are two different conventions typically used: 4-connected or 8-connected neighbourhoods.
In an 8-connected neighbourhood, all of the pixels that touch the pixel of interest are considered, including those on the diagonals. This means that if two adjoining pixels are on, they are part of the same object, regardless of whether they are connected along the horizontal, vertical, or diagonal direction.

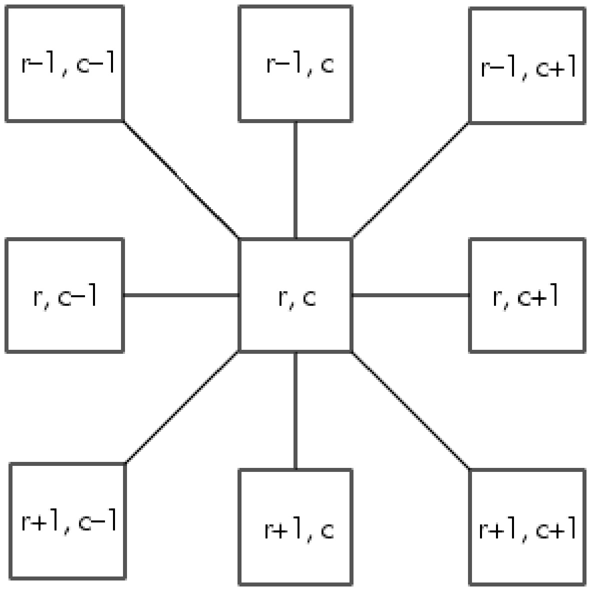


Figure 11 *8-Connected Neighbourhood*

In a 4-connected neighbourhood, the pixels along the diagonals are not considered. This means that a pair of adjoining pixels is part of the same object only if they are both on and are connected along the horizontal or vertical direction.

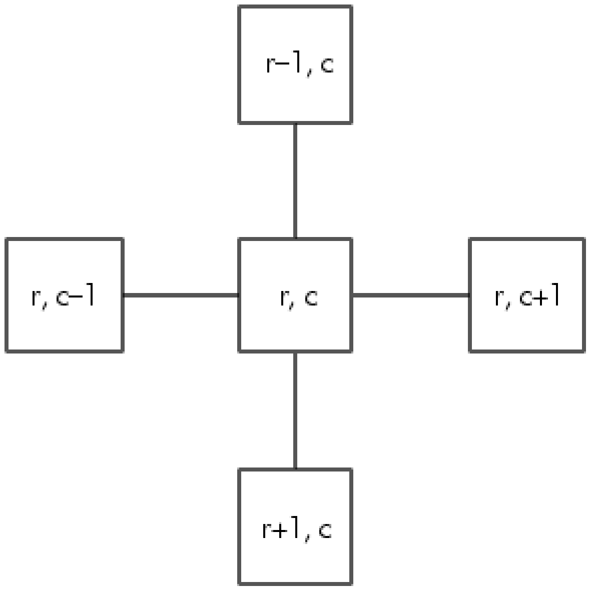


Figure 12 *4-Connected Neighbourhood*

The type of neighbourhood you choose affects the number of objects found in an image and the boundaries of those objects. Therefore, the results of the object-based operations often differ for the two types of neighbourhoods.

For example, this matrix represents a binary image that has one 8-connected object or two 4-connected objects.

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

3.3.2 Algorithm for Perimeter Determination

We have used 8-connected neighbourhood for perimeter determination. A pixel is considered a perimeter pixel if it satisfies both of these criteria:

- It is an on pixel.
- One (or more) of the pixels in its neighbourhood is off.

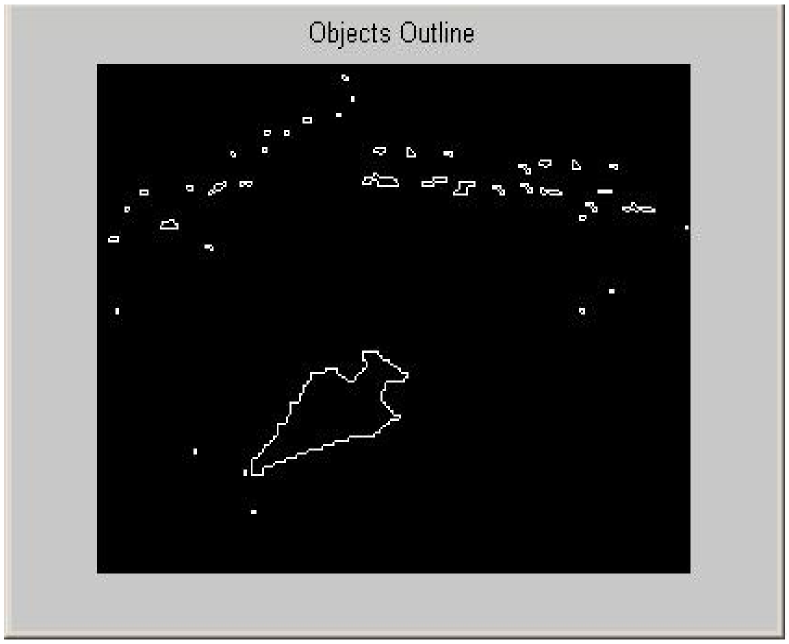


Figure 13 *Objects Perimeter*

Figure 13 shows the different objects with perimeter only. These objects are further analyzed for object recognition

Chapter 4:

Object Recognition and Real World Coordinates Estimation

4.1 Object Recognition

There are different techniques for object recognition :-

4.2 Input/Output of Object Recognition Phase

The input to this phase is an image with objects outline and the output is a single recognized object. The ANN makes a bounding box enclosing the recognized target.

This is the most critical phase of image processing module and involves much more effort compared to other phases in image processing.

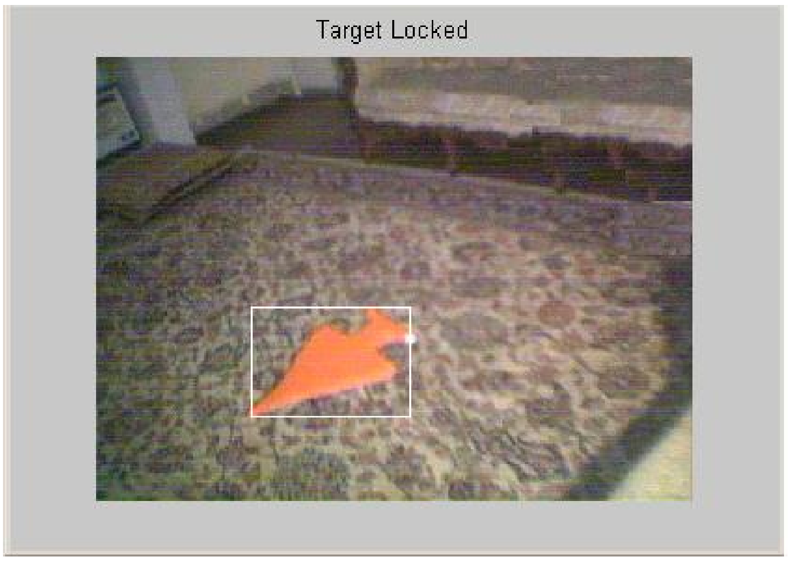


Figure 14 *Target Locked after Object Recognition*

4.4 Estimation of Real World Coordinates of Target

Camera calibration has been achieved by keeping the position of the camera fixed.

Different samples values have been measured and then making a look up table as shown below.Total 80 samples images were taken for camera calibration.

The centroid of the target in images have been found and then the corresponding closest match is searched for the Real World Coordinate System.

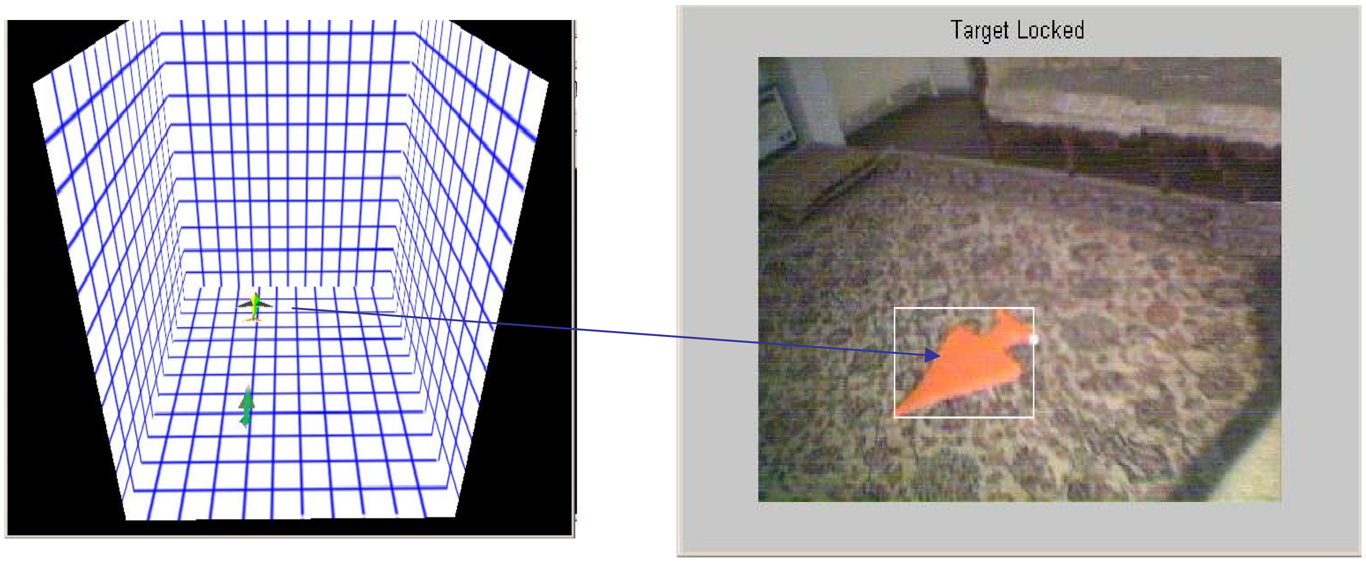


Figure 15 *Real World Coordinates Estimation*

Coordinates of Target in Image	Real World Coordinates of Target
0,41	160,162,0
0,61	158,133,0
-90,181	29.45,0
90,221	27,10,0
• • • •	• • •

Table 3 *Real World Coordiante Estimation*

Once the rea lworld coordinates has been found then the locomotion control system controls the locomotive to hit the target.

Appendix A

Implementation Notes

The Image Processing Module implemented using Matlab 6

Digital Image Processing toolbox was used for all image operations.

APPENDIX B

```
function ok();
rw_cent_y(1)=41;%x is zero on this line
rw_cent_y(2)=221;
%image coordinate on rw_center
% x is 160 in the image center
im_cent_y(1)=162.35;
im_cent_y(2)=133.22;
im_cent_y(3)=108.43;
im_cent_y(4)=87.04;
im_cent_y(5)=72.02;
im_cent_y(6)=57.07;
im_cent_y(7)=44.54;
im_cent_y(8)=34.4;
im_cent_y(9)=23.77;
im_cent_y(10)=13.0;

% Left side of the data from centeral axis(y) in real world 30 cm from left
rw_n_1y(1)=21;
rw_n_1y(2)=221;

% coordinates in image in x-axis
im_n_1x(1)=83.65;
im_n_1x(2)=89.28;
im_n_1x(3)=90.43;
im_n_1x(4)=93.46;
im_n_1x(5)=101.27;
im_n_1x(6)=103.812;
im_n_1x(7)=107.65;
im_n_1x(8)=108.72;
```

```
im_n_1x(9)=115;
im_n_1x(10)=116.62;
im_n_1x(11)=118.30;

% coordinates in image in y-axis
im_n_1y(1)=188.68;
im_n_1y(2)=157.24;
im_n_1y(3)=134.24;
im_n_1y(4)=111;
im_n_1y(5)=88.87;
im_n_1y(6)=69.95;
im_n_1y(7)=59.03;
im_n_1y(8)=43;
im_n_1y(9)=27;
im_n_1y(10)=24;
im_n_1y(11)=16;

%% coordinates in real world  in y-axis and deviated 60 cm to the left
rw_n_2y(1)=81;
rw_n_2y(2)=221;

% coordinates in image in x-axis
im_n_2x(1)=38.56;
im_n_2x(2)=46.1;
im_n_2x(3)=53.22;
im_n_2x(4)=60.53;
im_n_2x(5)=69.15;
im_n_2x(6)=73.206;
im_n_2x(7)=78.39;
im_n_2x(8)=88.67;
```

```
% coordinates in image in y-axis
im_n_2y(1)=112.86;
im_n_2y(2)=93.1;
im_n_2y(3)=76.69;
im_n_2y(4)=64.39;
im_n_2y(5)=50.59;
im_n_2y(6)=39.45;
im_n_2y(7)=28.82;
im_n_2y(8)=17.07;

%% coordinates in real world  in y-axis and deviated 90 cm to the left
rw_n_3y(1)=181;
rw_n_3y(2)=221;

% coordinates in image in x-axis
im_n_3x(1)=29.13;
im_n_3x(2)=34.18;
im_n_3x(3)=43.92;

% coordinates in image in y-axis
im_n_3y(1)=45.50;
im_n_3y(2)=35.49;
im_n_3y(3)=25.29;

% here we go in left quadrant

% right side of the data from centeral axis(y) in real world 30 cm from right
rw_p_1y(1)=21;
rw_p_1y(2)=221;

% coordinates in image in x-axis
```

```
im_p_1x(1)=236.84;
im_p_1x(2)=236.53;
im_p_1x(3)=227.43;
im_p_1x(4)=219.29;
im_p_1x(5)=213.85;
im_p_1x(6)=21.69;
im_p_1x(7)=206.28;
im_p_1x(8)=202.77;
im_p_1x(9)=201.47;
im_p_1x(10)=199.91;
im_p_1x(11)=202;

% coordinates in image in y-axis
im_p_1y(1)=180;
im_p_1y(2)=158.32;
im_p_1y(3)=129.56;
im_p_1y(4)=105.41;
im_p_1y(5)=84.74;
im_p_1y(6)=68.32;
im_p_1y(7)=54.36;
im_p_1y(8)=42.45;
im_p_1y(9)=31.43;
im_p_1y(10)=21.47;
im_p_1y(11)=20.67;

%% coordinates in real world  in y-axis and deviated 60 cm to the left
rw_p_2y(1)=81;
rw_p_2y(2)=221;

% coordinates in image in x-axis
im_p_2x(1)=279;
```

```
im_p_2x(2)=269.91;
im_p_2x(3)=263.47;
im_p_2x(4)=256.3;
im_p_2x(5)=248.68;
im_p_2x(6)=240.85;
im_p_2x(7)=237.84;
im_p_2x(8)=238.19;

% coordinates in image in y-axis
im_p_2y(1)=100.6;
im_p_2y(2)=82.79;
im_p_2y(3)=63.16;
im_p_2y(4)=51.3;
im_p_2y(5)=40.41;
im_p_2y(6)=30.52;
im_p_2y(7)=18.939;
im_p_2y(8)=19.13;

%% coordinates in real world  in y-axis and deviated 90 cm to the left
rw_p_3y(1)=181;
rw_p_3y(2)=221;

% coordinates in image in x-axis
im_p_3x(1)=286.23;
im_p_3x(2)=276.60;
im_p_3x(3)=274;

% coordinates in image in y-axis
im_p_3y(1)=25.85;
im_p_3y(2)=15.64;
im_p_3y(3)=10.42;
```

```
I = imread('c:\pic','bmp');
P=I;
figure,imshow(I)
title('Original Image')
s=size(I);
hsv=rgb2hsv(I);
figure,imshow(hsv)
title('RGB Image Converted to HSV Color Model')

%imhist(hsv);
d=size(hsv);
h=hsv(:,:,1);
s=size(h);
for j=1:s(1,2)
  for l=1:s(1,1)
    if   hsv(l,j,1)<0.5 & hsv(l,j,2)>0.8
           segmented(l,j,1)=I(l,j,1);
                                  segmented(l,j,2)=I(l,j,2);
                                  segmented(l,j,3)=I(l,j,3);
        I(l,j,1)=255;
         I(l,j,2)=255;
         I(l,j,3)=255;
        else
         I(l,j,1)=0;
         I(l,j,2)=0;
         I(l,j,3)=0;
      end
    end
  end

  figure,imshow(segmented);
  title('Color Based Segmentation')
  binary=im2bw(I);
  figure,imshow(binary);
  title('Color Image Converted to Binary Image')
  cleaned=bwmorph(binary,'clean',10);
  figure,imshow(cleaned);
  title('Cleaned Operatioin Performed on Binary Image')
  matrix=ones(1,1);
  eroded=erode(cleaned,matrix);
  figure,imshow(eroded);
  title('Eroded Image(Erosion Operation Performed)')
  matrix=ones(3,3);
  dilated=dilate(eroded,matrix);
  figure,imshow(dilated);
  title('Dilated Image(Dilation Operation Performed)')
  filled=bwfill(dilated,3,3,4);
  matrix=ones(2,2);
  eroded=erode(dilated,matrix);
  figure,imshow(eroded);
  title('Eroded again Image')
  boundary=bwperim(eroded);
  figure,imshow(boundary);
  title('Objects Outline')

%   contour=imcontour(eroded);

%   figure,imshow(contour)

%   title('Contour Image')
  area=bwarea(eroded);
  Label=bwlabel(eroded,8);
```

```
stats=imfeature(Label,'Area');
idx=find([stats.Area]>990);
object=ismember(Label,idx);

%figure,imshow(object);

%title('Object image')
boundary=bwperim(object);

%figure,imshow(boundary);

%title('Boundary of Objects')
stats=imfeature(object,'Area','Centroid','BoundingBox');
x1=ceil(stats.BoundingBox(1));
y1=ceil(stats.BoundingBox(2));
      x2=ceil(x1+stats.BoundingBox(3));
y2=ceil(y1+stats.BoundingBox(4));
for m=x1:x2
  for n=y1:y2
    if(m==x1 | m==x2 | n==y1 | n==y2)
    P(n,m,1)=255;
    P(n,m,2)=255;
    P(n,m,3)=255;
  end
  end
end
figure,imshow(P)
title('Target Locked');
x1=stats.Centroid(1);
y1=stats.Centroid(2);
```

```
 %uint rw_x;

%uint rw_y;

%rw_x=uint(rw_x);

%rw_y=uint(rw_y);
indices=-1;
   if x1>=160
       x=x1-160;
       x=x*x;
       for i=1:10
          y=y1-im_cent_y(i);
          y=y*y;
          distance(i)=sqrt(x+y);
       end
     j=i+1;
    for i=1:11
       x=x1-im_p_1x(i);
       y=y1-im_p_1y(i);
       x=x*x;
       y=y*y;
       distance(j)=sqrt(x+y);
       j=j+1;
    end
     for i=1:8
       x=x1-im_p_2x(i);
       y=y1-im_p_2y(i);
       x=x*x;
       y=y*y;
       distance(j)=sqrt(x+y);
```

```
      j=j+1;
    end
    for i=1:3
      x=x1-im_p_3x(i);
      y=y1-im_p_3y(i);
      x=x*x;
      y=y*y;
      distance(j)=sqrt(x+y);
      j=j+1;
    end
  [min_dist,indices]=min(distance);
  end
    if indices>=1 & indices<=10;
      indices=indices-1;
      rw_x=0;
      rw_y=41+20*(indices);
    end
   if indices>=11 & indices<=21;
      indices=indices-10;
      indices=indices-1;
      rw_x=30.5;
      y_shift=200*(indices);
      rw_y= 21 + y_shift;
    end
    if indices>=22 & indices<=29


      indices=indices-21;
      indices=indices-1;
      rw_x=61;
      y_shift=5*(indices);
      rw_y= 81 + y_shift;
```

```
        end

        if indices>=30 & indices<=42
          indices=indices-29;
          indices=indices-1;
          rw_x=91.5;
          y_shift=20*(indices);
          rw_y= 181 + y_shift;
        end

% here we tackle the plane in left plane
        if x1<160
          x=x1-160;
          x=x*x;
          for i=1:10
            y=y1-im_cent_y(i);
            y=y*y;
            distance(i)=sqrt(x+y);
          end
        j=i+1;
        for i=1:11
          x=x1-im_n_1x(i);
          y=y1-im_n_1y(i);
          x=x*x;
          y=y*y;
          distance(j)=sqrt(x+y);
          j=j+1;
        end
        for i=1:8
          x=x1-im_n_2x(i);
          y=y1-im_n_2y(i);
        x=x*x;
        y=y*y;
        distance(j)=sqrt(x+y);
        j=j+1;
    end
    for i=1:3
        x=x1-im_n_3x(i);
        y=y1-im_n_3y(i);
        x=x*x;
        y=y*y;
        distance(j)=sqrt(x+y);
        j=j+1;
    end
[min_dist,indices]=min(distance);
end

    if indices>=1 & indices<=10;
        indices=indices-1;
        rw_x=0;
        rw_y=41+20*(indices);
    end
    if indices>=11 & indices<=21;
        indices=indices-10;
        indices=indices-1;
        rw_x=-30.5;
        y_shift=20*(indices);
        rw_y= 21 + y_shift;
    end
    if indices>=22 & indices<=29
        indices=indices-21;
        indices=indices-1;
```

```
   rw_x=-61;
   y_shift=20*(indices);
   rw_y= 81 + y_shift;
end
if indices>=30 & indices<=32
   indices=indices-29;
   indices=indices-1;
   rw_x=-91.5;
   y_shift=20*(indices);
   rw_y= 181 + y_shift;
end
fid=fopen('c:\new.txt','w');
   fprintf(fid,'%6.2f %12.8f\n',rw_x);
   fprintf(fid,'%6.2f %12.8f\n',rw_y);
fclose(fid);
```

Appendix C

Recommendatoins

Since it is the first iteration of the project, so basic idea has been implemented successfully and the foundation has bee laid. Following are the recommendations for further enhancements provided that you have more resources to continue.

1. WebCam can be replaced by a customized camera used in Robotics whose intrinsic parameters are known(i.e provided by the Vendor)
2. A camera can be mounted on stepper motor so that camera can lock the target even when the target is moving.
3. Algorithm for Moving Target can be implemented in the vicinity of the camera.

Appendix D

References

1. Digital Image Processing by AK Jain, for Morphological image operations.
2. Zarit, B.D., Super, B.J., Quek, F.K.H., Dept. of Electr. Eng. & Comput. Sci., Illinois Univ., Chicago, IL, USA, Comparison of five colour models, (Proceedings International Workshop on Recognition, Analysis, and Tracking (RATFG-RTS'99), Corfu, Greece, 26-27 Sept. 1999.) Los Alamitos, CA, USA: IEEE Comput. Soc, 1999. P.58-63.
3. www.cs.rit.edu/~ncs/color/t_convert.html, for color space conversión
4. Pauwels, E.J., Frederix, G., Dept. of Electr. Eng., Katholieke Univ., Leuven, Belgium, Colour segmentation and shape extraction for image interpretation
5. Pitas, Fundamentals of colour image processing. In: *Digital Image Processing Algorithms* Prentice-Hall, Englewood Cliffs, NJ (1993), pp. 23–40.

Part II: Locomotive Control System

Defines the services, protocols, and the configurations of hardware and software.

EXECUTIVE SUMMARY

This document describes a sub-system of TTIP called as locomotion control system. It also elaborates the problem (the client’s view) along with its essence and inherent properties coupled with the analysis (the developer’s view) of the problem and the process. Next we see further details of the process with the object oriented analysis and design. After that, system is ready for coding. Configurations, tests, bug-fixes are also written in this document.

Acknowledgements

Without others support it would not have been possible to accomplish this project. The cooperation of my teachers is not forgettable.

Chapter 5 : Introduction

5.1.1 The System Defined

The locomotion control system is part of the TTIP. It is responsible for the control of the Locomotive car in the field. It will take input from Master Control to go for a position in the field.

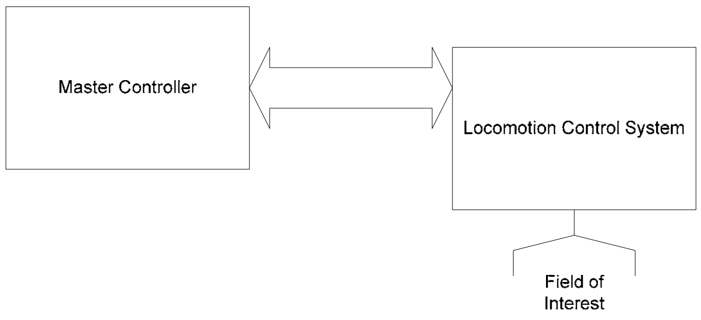


Figure 16 Logical Relationships between Master Controller and The Locomotion Control System

The scope of the Locomotion Control System (LCS) must be clearly understood. It means that whether some service or protocol is part of the system or not. Once such a crisp boundary is found, we can express the system's view as the use-case.

5.2 The Problem Statement

The desired system must be able to control a locomotive through a coordinate system *(O, x, y)* from any initial point *P(x, y)* to final point *Q(x, y).*

To achieve this effect, there must be a mathematical model describing the paths of the locomotive. In this case, the locomotive is a toy cat that can move ahead, backwards, left, and right. Different constants for this car must be calibrated as per the mathematical model.

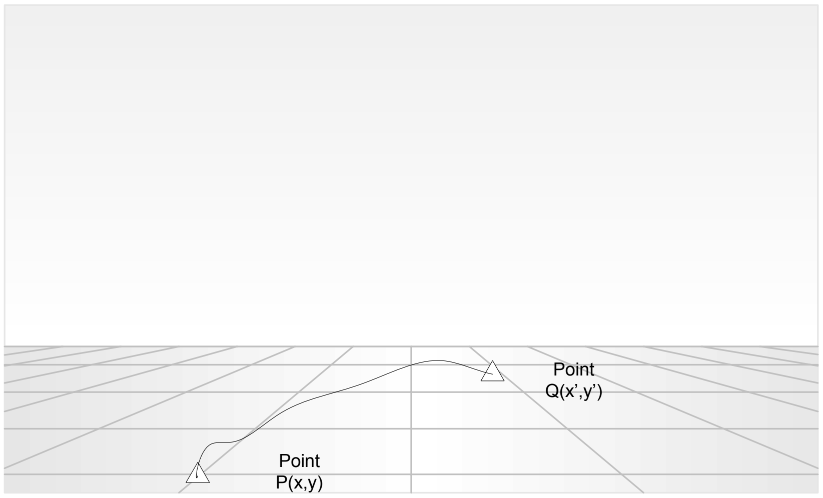


Figure 17 The representation of a scenario when Locomotive L seeks point Q from P

The patterns for the paths must be figured out and modelled as linear equations or splines functions etc. Then real time processing must be employed to track the locomotive and make the decisions on time. Note that the locomotive's constants must be classified such that there is no dependency of the configuration upon the binaries.

The locomotive is a small car. Its paths are well defined through the combination of linear as well as elliptic curves. These paths must be worked out accurately and the car will be instructed to move. Note that the car moves in the world coordinate system. Either you can fix a unit of the mesh as an inch, foot, meter, etc. Or you can use world-to-machine coordinate system transformations.

The life span of car (which is playing the role of a locomotive) is very small. The life span is defined as the time to reach from point P to Q. It can be assumed that the life span of the locomotive is 1~10 seconds.

Different constants, as felt necessary, shall be calibrated from the given locomotive. That car is navigable in the field through its remote controller working at 27 MHz.

5.2.1 Problem Analysis

This problem is clearly related to hardware interfacing. The control circuit is analog. It must be interfaced with the communication port of the controlling computer. For parallel transmission of several signals and their combinations, we have used the parallel port of the computer which is capable of sending data from all the data pins, simultaneously.

The controller of the system is the entity which will handle all the application logic to hit the target. It must have nothing to do with the underlying port or protocol. Its only requirement is to encapsulate the functionality of the controller.

The mind map diagram of the system at this stage can be visualized as in figure 18. The important characteristics of the system are its real-time or almost-real-time execution. At the moment, when the system is signalled to move backward and turn left, then it must be able to do it in real-time.

Moreover, the eventual goal is to implement the application logic i.e. when to move and where to move, and controlling mechanism separately. Object-Oriented technology must be used to separate both of the areas by providing one or more user-defined data-types for each.

The reason for the implementation through objects is its power to reuse and provide controlled concurrency over the port. The port class should be developed for Intel 386 class of processors running in protected mode. This needs to gain access to the port which is not allowed for a user-level process. This issue needs to be considered and kernel-mode code must be needed to accomplish this.

Chapter 6 : Mathematical Modelling

6.1 Abstract

Mathematics is the language of nature. It is the abstraction for the harmony found in nature. Planetary motion to growth of plants is understood by the human-beings through abstraction of mathematical symbols. Thus, any physical phenomenon can be abstracted in the language of mathematics. Within the constraints of our system, we need to model a similar technique using mathematics. Here is what we found.

The Problem

Locomotive used in our implementation is capable of turning left or right and it can move backwards and forward. Finite-State automation of this scenario is:

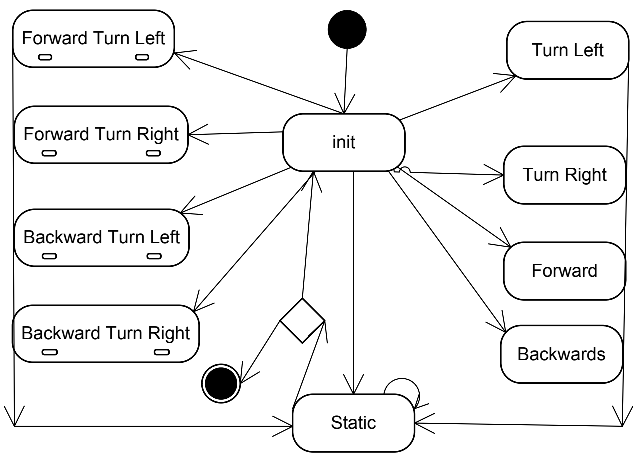


Figure 18 *The FSA diagram of the locomotive controller*

6.2 The Problem

The locomotive can move in circular paths or in straight lines this is very easy to model using 2D analytical geometry.

6.3 Mathematical Model

Intersection of a line and a circle

To prove that the straight line y = m x + c cuts the circle x2+y2=a2 in two points also to find the condition that the line may touch this circle.

The coordinates of the point in which the straight line

$y = m x + c$ (1)

and the circle

$x^2+y^2 = a^2$ (2)

intersect, satisfies both the equation and thus they can be obtained by solving eq (1) and (2) simultaneously substituting for y from eq (1) in eq (2), the abscissae of the points of intersection are given by the eq :

$x^2 + (mx+c)^2 = a^2$

$(1+m^2) x^2 + 2mcx + (c^2 - a^2) = 0$ (3)

this is quadratic in x and thus gives two values of x

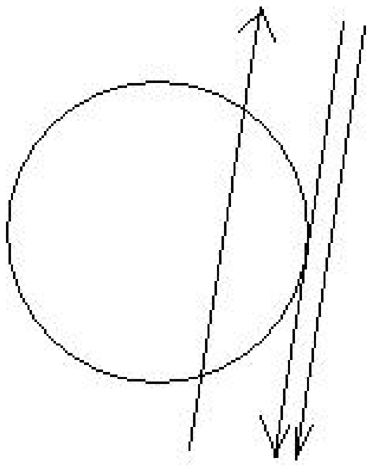


Figure - 19

These two values of x are nothing but abscissae of the two points of intersection corresponding two abscissae x1, x2. we can find two coordinates y1 and y2 from eq (1) and thus the coordinates of the two points of intersection are determined the two points of intersection are real and distinct, real and coincident or imaginary according as the roots of eq (3) are real and distinct, real and equal or imaginary

the line will touch the circle if two points of intersection coincide or roots of eq (3) are equal

$(2mc)^2 - 4(1+m)^2 (c^2 - a^2) = 0$

$4m^2 c^2 - 4(c^2 - a^2 + m^2c^2 - m^2a^2) = 0$

$c^2 = a^2(1 + m^2)$

$c = \pm a \sqrt{1 + m^2}$

Which is the required condition that the line (1) may touch the circle (2) or in other words may become tangent to the circle

Hence $y = mx \pm a\sqrt{1+m^2}$

are the eqs of the tangent to the circle (2) for all values of m.

Length of a tangent segment to a circle from a given point

To find the length of the tangent from an external point p (x1, y1) to the circle

$x^2 + y^2 + 2gx + 2fy + c = 0$

let PT and PT' be the tangents from p (x1, y1) to the given circle whose centre is

C (-g, -f).

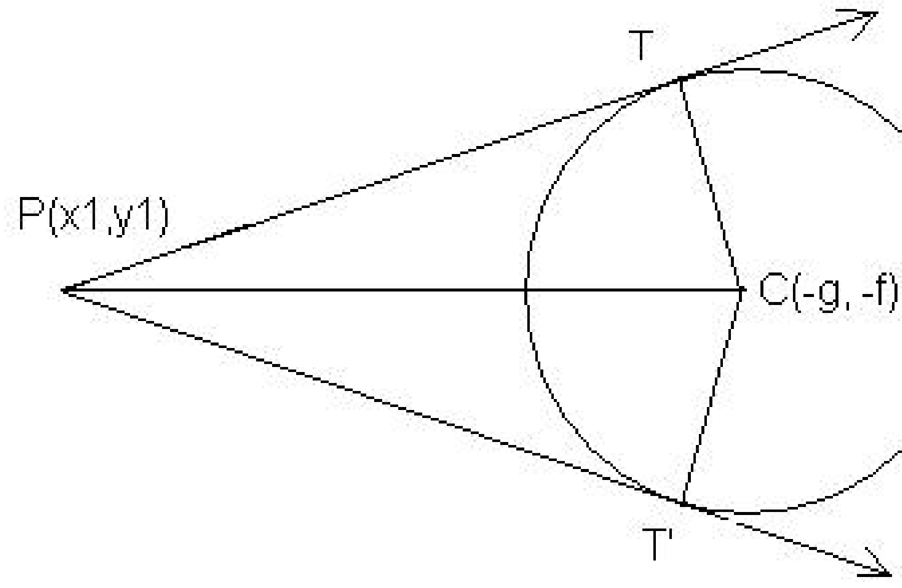


Figure - 20

Join PC , CT and CT'

Now $PC^2 = (x1 + g)^2 + (y1 + f)^2$

$= x1^2 + y1^2 + 2gx1 + 2fy1 + g^2 + f^2$

$TC^2= g^2 + f^2 – c$ (required radius)

m < PTC = 90 deg

from right side , Δ PTC we get

$PT^2 = PC^2 – CT^2$

$= (x1^22 + y1^2 + 2gx1 + 2fy1 + g^2 + f^2) – (g^2 + f2 – c)$

$= x1^2 + y1^2 + 2gx1 + 2fy1 + c$

Which is the square of the length of the tangent drawn from (x1, y1) to the given circle.

Equation of tangent circle $x^2 + y^2 = r^2$

$y = mx + c$

$y = mx + r \sqrt{} I + m^2$

through point (a, b)

$b = ma + r \sqrt{} 1 + m^2$

$(b – ma)^2 = (r \sqrt{} 1 + m^2)^2$

$b^2 – 2abm + m^2a^2 = r^2 (1+ m^2)$

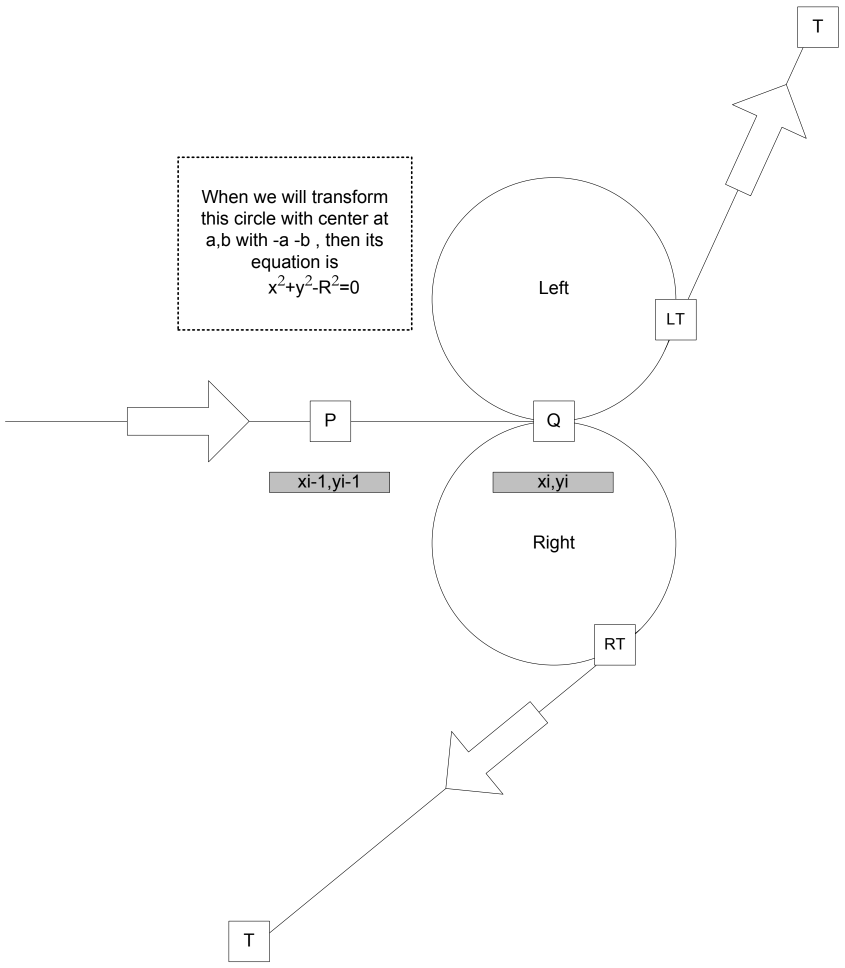
$b^2 – 2ab (m) + (m^2) a^2 – r^2 –r^2m^2 = 0$

$a^2m^2 - r^2m^2 – 2abm + (b^2– r^2) = 0$

$m^2 (a^2 – r^2) – (m) 2ab + (b^2 – r^2) = 0$

$m = +2ab \pm \sqrt{} (2ab)^2 – 4 (a^2 – r^2) (b^2 – r^2) / 2 (a^2 – r^2)$

$m = - 2 (a^2– b^2) (2ab) \pm \sqrt{} (2ab)^2 – 4 (a^2 – r^2) (b^2 – r^2) / 2 (a^2 – r^2)$



T
When we will transform this circle with center at a,b with -a -b , then its equation is $x^2+y^2-R^2=0$
Left
LT
P
Q
xi-1,yi-1
xi,yi
Right
RT
T

Figure 21 Basic Model Diagram

Basic Model diagram showing that the locomotive has two circles at any position. When it starts moving in the circular path, its motion is defined by equation of circle and then when it leaves the circle, its motion is defined by the equation of the line.

We begin our analysis by looking at the type of path adopted by the locomotive to hit the target. The problem was:

"*Find a point P on the circle with centre at C(x, y) and with Radius R such that the tangent drawn at point P and point T are collinear*"

To achieve this effect we derived it like this:

Let the circle be on the origin and let its radius be equal to R. Then equation of the tangent to the circle $x^2+y^2-R^2=0$ from a point T (a, b) is given as:

$b=ma\pm\sqrt{(1+m^2)}$ (1)

where *a* and *b* coordinates of point T

Equation of the tangent through any point P(x, y) is

$y= mx \pm r\sqrt{(1+m^2)}$

Solving equation (1) for m yields

$m= (2ab\pm\sqrt{((2ab)^2 -4(a^2-r^2)(b^2-r^2))})/2(a^2-r^2)$ (2)

Now let the equation of the line be

$y-y_1=m(x-x_1)$

Solving it simultaneously with equation of the circle yields

$x_{ab}=2x\pm\sqrt{(4x^2-4(1+m^2)(x^2+y^2-R^2))}/2(1+m^2)$

Where x1, y1 are the coordinates of Target point T.

$Y_a= m(x_a-x_1)+y_1$ (A)

$Y_b=m(x_b-x_1)+y_1$ (B)

Then points (x_a, y_a) and (x_b,y_b) are two tangent points.

Then select one of the points which occur first in counter clockwise direction.

If is the point T is inside the circle the problem became:

"Find a centre of the circle that fits the point T, or find an equation of the circle that satisfies T (a, b)"

Our derivation is explained from the figure as:

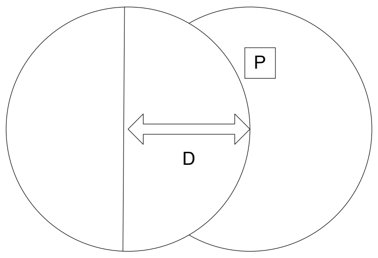


Figure - 22

There are two sectors:

If the target point is in the RHS sector

New centre is C $(-(R^2-y1^2-x_1), 0)$

If the target point is in the LHS sector

New centre is C $(-(R+ (R^2-y_1^2-x_1)), 0)$

Where centre C is assumed at origin of the coordinate system.

Chapter 7 : Circuit Designing

7.1 The Circuit

The actual locomotive was equipped with a remote control and the car. The remote control circuit is made up of four switches and a radio wave emitter. It generates radio waves signals which were identified as a radio wave whose frequency is 27 MHz. The car is also equipped with receiver at same frequency. We interfaced that circuit with its switches attached to a Darlington array called ULN 2003. When the pulse from the parallel port is generated by our program, it switches on the open path. Or diagrammatically as:

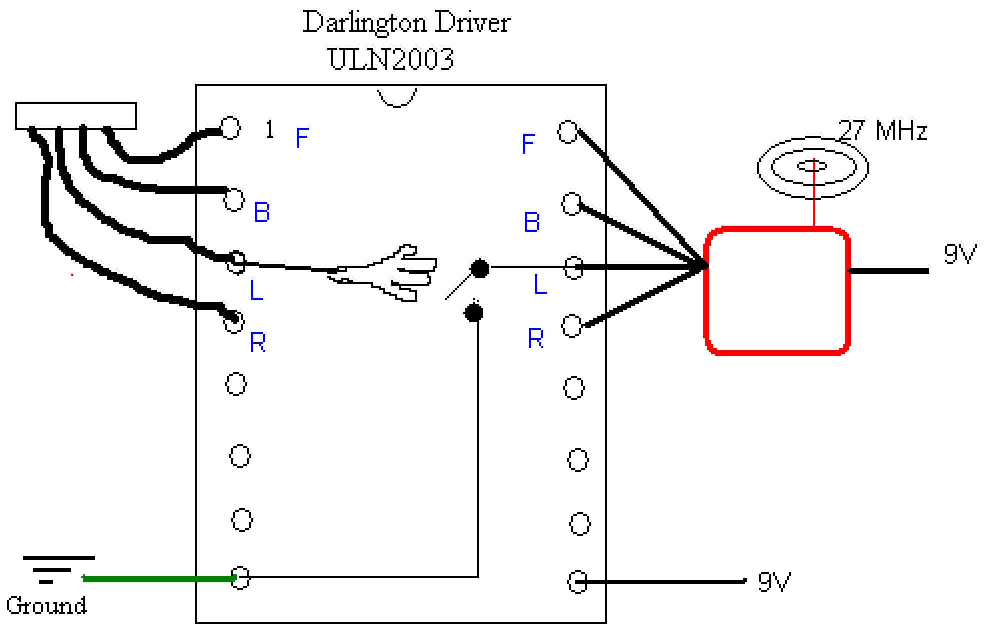
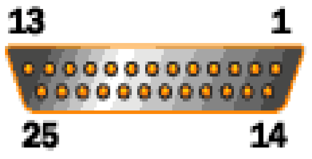


Figure 23 *Schematic representation of the interfacing circuit*

7.2 Parallel Port

When a PC sends data to a printer or other device using a parallel port, it sends 8 bits of data (1 byte) at a time. These 8 bits are transmitted parallel to each other, as opposed to the same eight bits being transmitted serially (all in a single row) through a serial port. The standard parallel port is capable of sending 50 to 100 kilobytes of data per second



(25 pin D-SUB female at the PC)

PIN	PURPOSE
Pin 1	-Strobe
Pin 2	+Data Bit 0
Pin 3	+Data Bit 1
Pin 4	+Data Bit 2
Pin 5	+Data Bit 3
Pin 6	+Data Bit 4
Pin 7	+Data Bit 5
Pin 8	+Data Bit 6
Pin 9	+Data Bit 7
Pin 10	-Acknowledge
Pin 11	+Busy
Pin 12	+Paper End
Pin 13	+Select
Pin 14	-Auto Feed
Pin 15	-Error
Pin 16	-Initialize Printer
Pin 17	-Select Input
Pin 18	-Data Bit 0 Return (GND)
Pin 19	-Data Bit 1 Return (GND)
Pin 20	-Data Bit 2 Return (GND)
Pin 21	-Data Bit 3 Return (GND)
Pin 22	-Data Bit 4 Return (GND)
Pin 23	-Data Bit 5 Return (GND)
Pin 24	-Data Bit 6 Return (GND)
Pin 25	-Data Bit 7 Return (GND)

Table – 4 Pin Configuration of Parallel Port

The following is an explanation of each of the above purposes.

Pin1 = Data acknowledgement when the signal is low.
Pin 2 - 9 = Data transfer pins.
Pin 10 = Acknowledge that the data has finished processing and when the signal is high indicates ready for more.
Pin 11 = When the signal goes high indicate that the printer has accepted the data and is processing it. Once this signal goes low and Pin 10 goes high will accept additional data.
Pin 12 = Printer paper jam when signal is high or no signal if printer jam.
Pin 13 = When high signal printer is indicating that it is on-line and ready to print.
Pin 14 = When low signal PC has indicated that the printer inset a line feed after each line.
Pin 15 = Printer sends data to the computer telling it that an error has occurred.
Pin 16 = When low signal PC has requested that the printer initiate a internal reset.
Pin 17 = When low signal the PC has selected the printer and should in return prepare for data being sent.
Pin 18 - 25 = Ground.

7.2 Radio Control

The exact frequency used is 27 MHz. Here's the sequence of events that take place when you use the RC transmitter:

- You press a trigger to make the truck go forward.
- The trigger causes a pair of electrical contacts to touch, completing a circuit connected to a specific pin of an integrated circuit (IC).
- The completed circuit causes the transmitter to transmit a set sequence of electrical pulses.
- Each sequence contains a short group of synchronization pulses, followed by the pulse sequence. For our truck, the synchronization segment -- which alerts the receiver to incoming information -- is four pulses that are

2.1 milliseconds (thousandths of a second) long, with 700-microsecond (millionths of a second) intervals. The pulse segment, which tells the antenna what the new information is, uses 700-microsecond pulses with 700-microsecond intervals.

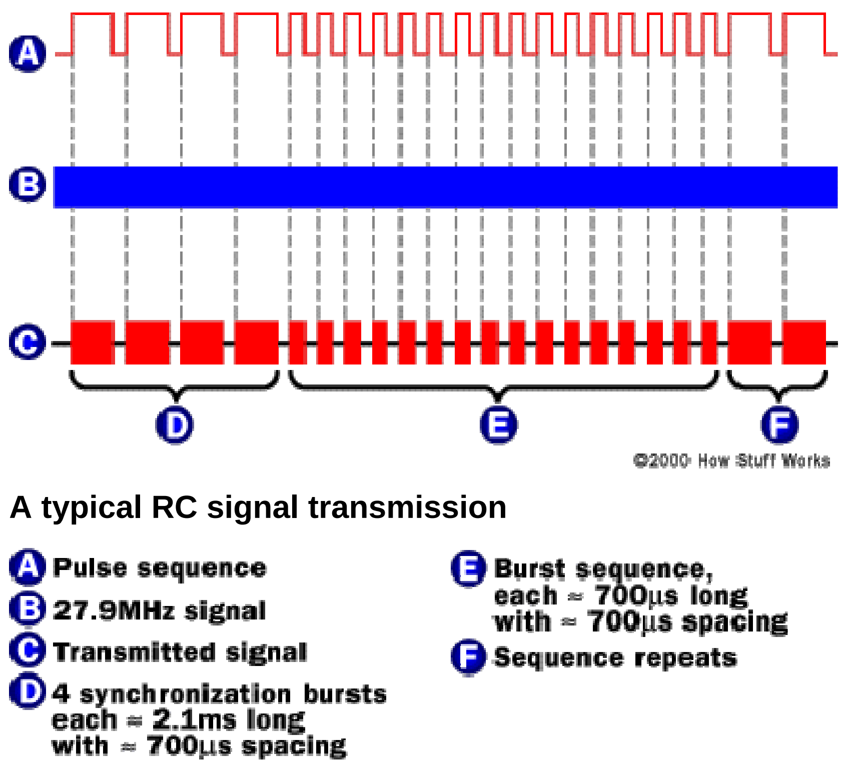
- 
A
B
C
D
E
F
©2000 How Stuff Works
A typical RC signal transmission
A Pulse sequence
B 27.9MHz signal
C Transmitted signal
D 4 synchronization bursts each ≈ 2.1ms long with ≈ 700µs spacing
E Burst sequence, each ≈ 700µs long with ≈ 700µs spacing
F Sequence repeats
-

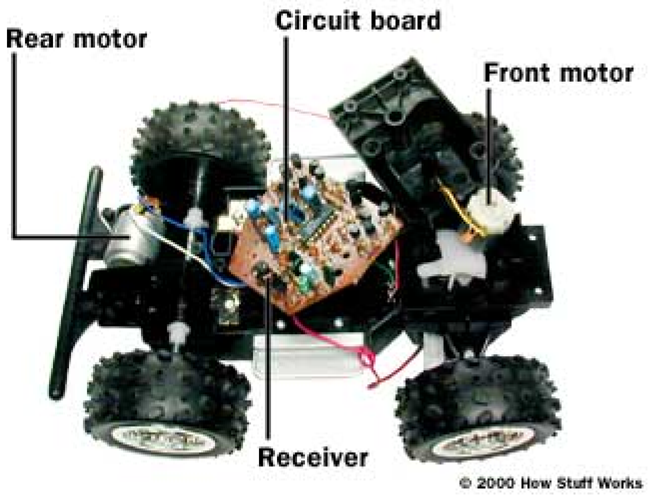
Figure – 24 The Radio Signal

Here are the pulse sequences used in the pulse segment:

- Forward: 16 pulses
- Reverse: 40 pulses
- Forward/Left: 28 pulses
- Forward/Right: 34 pulses
- Reverse/Left: 52 pulses
- Reverse/Right: 46 pulses

- The transmitter sends bursts of radio waves that oscillate with a frequency of 27,000,000 cycles per second (27 MHz).
- The truck is constantly monitoring the assigned frequency (27 MHz) for a signal. When the receiver receives the radio bursts from the transmitter, it sends the signal to a filter that blocks out any signals picked up by the antenna other than 27 MHz. The remaining signal is converted back into an electrical pulse sequence.
- The pulse sequence is sent to the IC in the truck, which decodes the sequence and starts the appropriate motor. For our example, the pulse sequence is 16 pulses (forward), which means that the IC sends positive current to the motor running the wheels. If the next pulse sequence were 40 pulses (reverse), the IC would invert the current to the same motor to make it spin in the opposite direction.
- The motor's shaft actually has a gear on the end of it, instead of connecting directly to the axle. This decreases the motor's speed but increases the torque, giving the truck adequate power through the use of a small electric motor!
- The truck moves forward.

If you look inside the RC truck, you will see that it is very simple: two electric motors, an antenna, a battery pack and a circuit board!



A look at the inside of the truck

Figure – 25 The Toy Car

One motor turns the front wheel right or left, while the other motor turns the rear wheels to go forward or backward. The circuit board contains the IC chip, amplifier and radio receiver. A few simple gears connect the motors to the wheels. It is really amazing how versatile the range of movement is with so few components.

Chapter 8 : Object Oriented Design

8.1 Use Case Model

The high level representation of the system is modelled with use-cases. Two actors are found interacting with this module:

1. Instructor
2. Locomotive

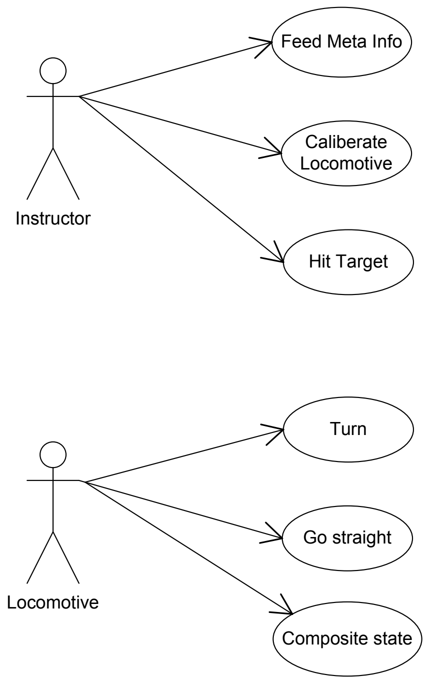


Figure 26 *It is the use-case view of the locomotion control module*

8.2 Class Diagram

.In depth analysis of the use-case model leads us to class diagram.

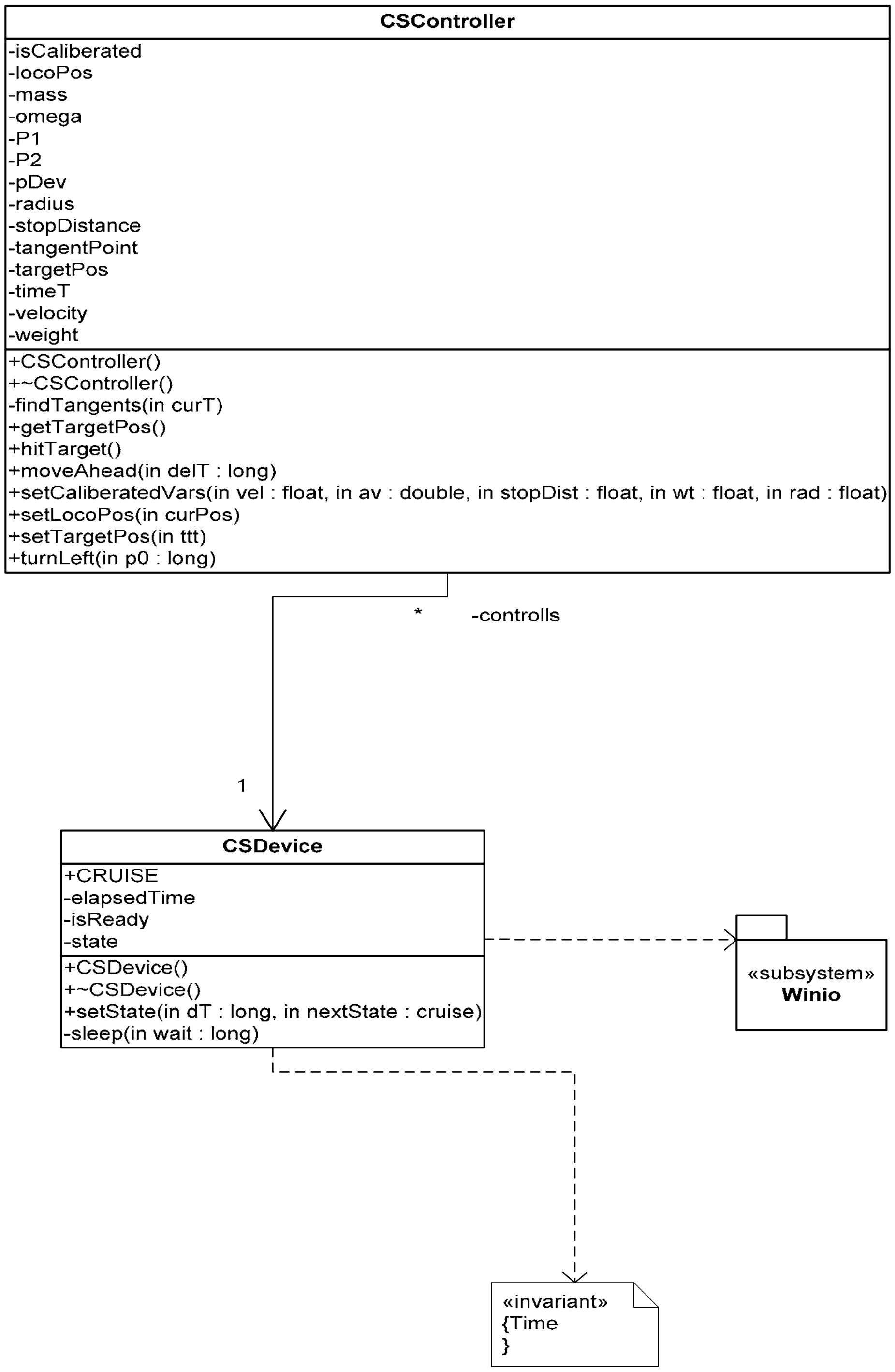


Figure 27 *Class Diagram*

8.3 CS Device Object State Chart

The following diagram represents the scenario when an object of the class CSDevice is instantiated. When its constructor is called, we enter into the initial state of this diagram and then the device enters into idle state. When the controller object instructs the device to send a signal, the object enters into a specific state for a finite time interval and then releases to idle state until its destructor is called and the object dies, we enter into exit state. Figure 7 illustrates the idea.

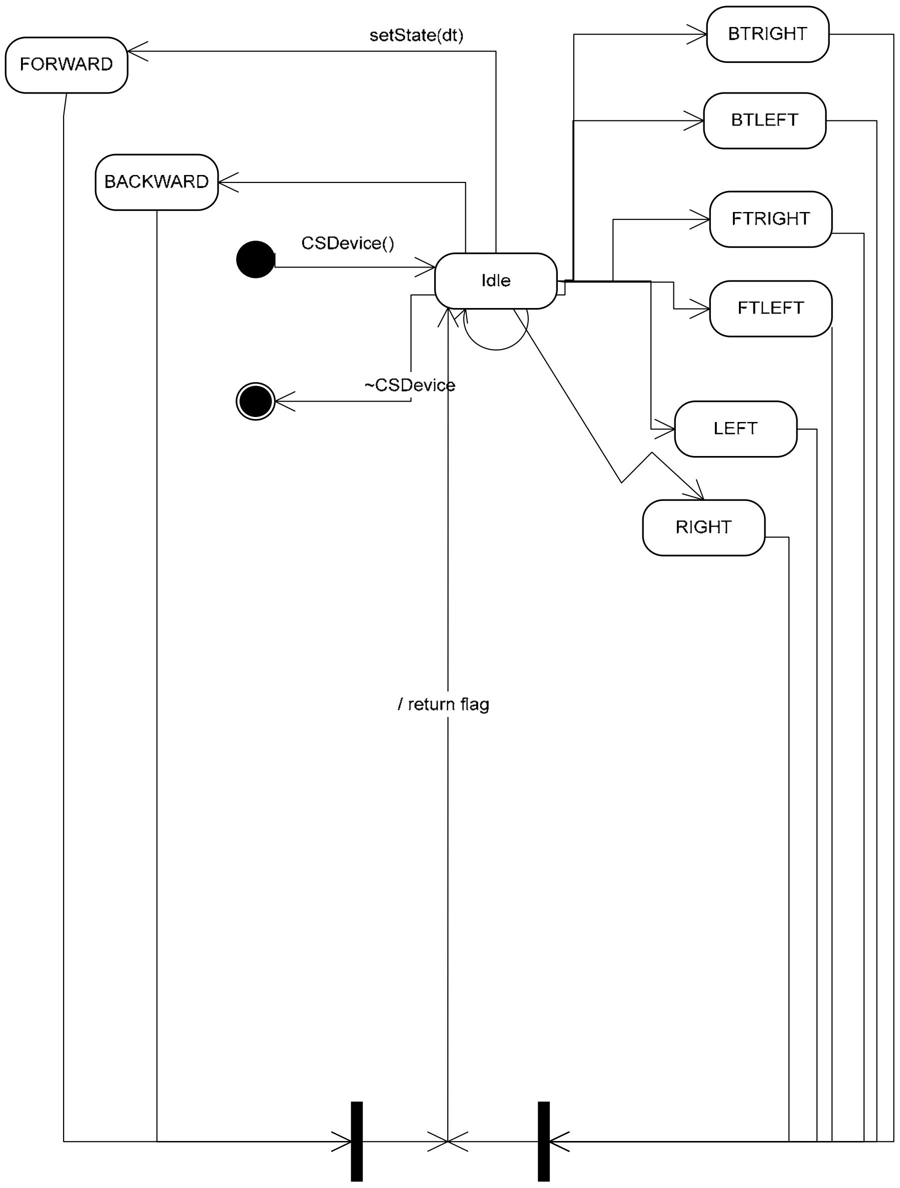


Figure 28 *State Chart for Class CSDevice Object*

Chapter 9 : System Calibration

9.1 Calibration

As our locomotive is free to move in the field and our controller class is going to control it, the system needs to know the locomotive-specific values. These values are constant with a surface and the locomotive. These are:

- Linear Velocity
- Angular Velocity
- Radius of turn circle
- Battery life etc.

For each of these, we developed a test; either manual or computer controlled. Some of the values are found out and the mean and variance are calculated and the values are fed to the system. There is a provision in the program that can adjust the 89values anytime you want. The calibration methods are:

9.2 Linear Velocity Calibration

The following procedure is adopted:

Procedure 1

- Mark the current position of the locomotive
- Make a small program that instructs the locomotive to move forward for exactly *x* milliseconds
- Find current position of the locomotive
- Measure the distance Ď between the two points
- Approximate errors due to inertia by taking many readings and subtract from Ď
- Linear velocity *v* = Ď/*x*
- Repeat the process for other values of time *x*

9.3 Turn Radius Calibration

The following procedure is adopted:

Procedure 2

- Mark the current position of the locomotive
- Turn it right or left and move forward until it comes back to its original position
- Note the position which is perpendicular to the starting point
- Measure the distance between both points
- It is diameter D. Find Radius r=D/2
- Iterate until values are close to each other

9.4 Angular Velocity Calibration

The following procedure is adopted:

Procedure 3

- Mark the current position of the locomotive
- Turn it right or left and move forward until it comes back to its original position
- Find the time *t* to complete this cycle through the program (by approximation)
- Then angular velocity ώ=2́/*t* measured in radians per millisecond

After calibrating these values for a plane surface Π, feed these values to the controller object. Then the controller object is able to control the locomotive effectively.

Chapter 10 : Implementation

10.1 Implementation Notes

- The system is implemented using
- C++ Programming Language
- Facilitation IDE used through out is Visual C++ 6.0

Other important facts:

The implementation is sticky to the platform for which it is developed i.e. if you take the implementation to Sun Spark, it will not run.

No container or algorithm from Standard Template Library (STL) has been employed.

10.2 Testing and Deployment

Testing of the system is quite simple. We attach the devices and then run the program to move the car, let's say twenty centimetres. Then we realize the results and if some problem has occurred, we analyzed the problem and solve it. The "bugzilla" chart is contained here in Appendix C.

As far as deployment of the system is concerned, we just need an x86 machine running NT class operating system. For DOS and Windows 9x, rebuild with changes is required. The current binaries are produced for Windows XP. The control module exposes its interface to the environment when it is running. Other modules can utilize this interface to do what they want. During deployment, it must be taken into account that there is a sheer dependency with WinIo; a kernel mode program used to gain access to the ports in I386 protected mode operating systems. Whenever the system is taken to another node, SYS and DLL files of this driver must be in the same directory where program executable is placed. The same is expressed using UML as:

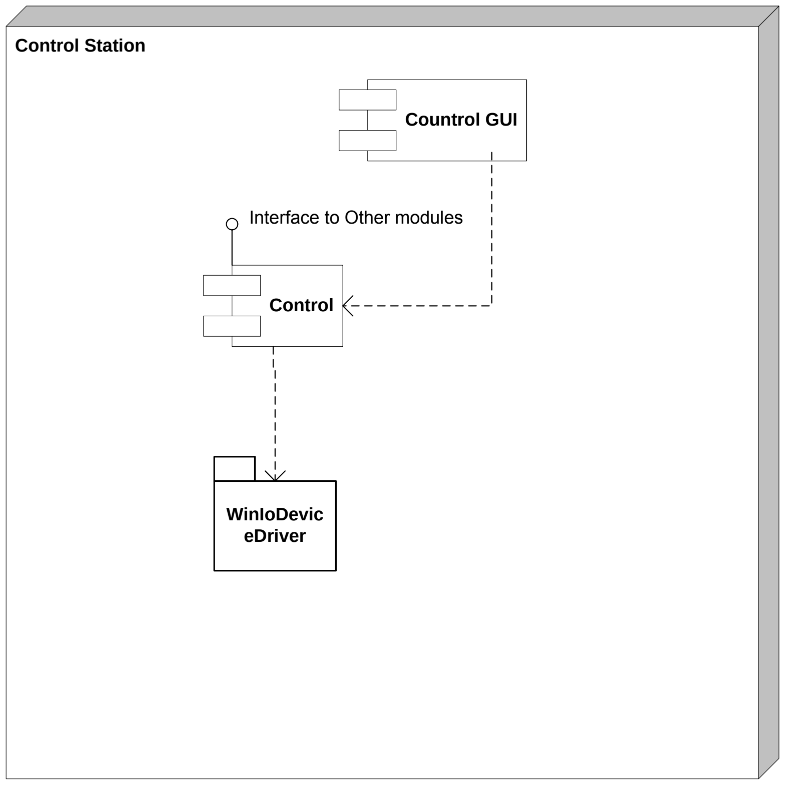


Figure 29 *The deployment diagram when program and the SYS file of driver are deployed on a NT class protected mode OS*

Appendix A

References

- Grady Booch
- www.beyondlogic.org
- Bjarne Stroustrup
- Mr. Schweller

Appendix B

All Supported Configurations

To date, the only supported configuration of the software is:
Running binaries on Windows XP/2000 with these shared libraries installed
WINIO device driver with winio.sys in the current directory
Nevertheless, it is required that the signalling device is attached to the port and supports the protocol given in figure 4, otherwise, no error will be generated because post state has been modified accordingly.
Runs in emulated 16-bit environment by NTVDM