Sign Board Detection and Information extraction

for MAVI (Mobility Assistant for Visually Impaired) project

A thesis submitted in partial fulfilment of
the requirements for the degree of

MASTER OF TECHNOLOGY
in

VLSI Design Tools and Technology

by
PAPPIREDDY DEDEEPYA
Entry No. 2014JVL2705

Under the guidance of

Prof. M.BALAKRISHNAN

&

Dr.Chetan Arora(IITD)

Department of VLSI Design Tools & Technology,
Indian Institute of Technology Delhi.
June 2016.

Certificate

This is to certify that the thesis titled "Sign Board detection and information extraction for MAVI (Mobility Assistant for visually impaired) project" being submitted by **Pappireddy.Dedeepya** for the award of the Master of Technology in VLSI Design Tools and Technology is a record of bona fide work carried out by her under my guidance and supervision. The work presented in this thesis has not been submitted elsewhere in part or full for the award of any other degree or diploma.

Supervisor:

Prof.M.Balakrishnan

Department of Computer Science Engineering

Indian Institute of Technology,Delhi.

Co-supervisor

Dr.Chetan Arora

IIIT Delhi

ACKNOWLEDGEMENT

I would like to express my solemn and sincere gratitude to Prof.M.Balakrishnan for giving me this opportunity to work on this project. I am highly indebted for the support he has offered me during my stay at IIT Delhi. He has motivated me all the time and has been a pillar of support.

I would also like to thank my co-supervisor Dr. Chetan Arora for his valuable insights and assistance regarding the concepts of image processing and computer vision. I extend my thanks to Mr. Siddhartha Gupta of Assistech Group, for all his efforts to develop an OCR engine for Hindi language and also for offering his valuable suggestions on improving the algorithm. I thank Mr.Munib Fazal and Saurabh Agarwal for their help in cross compilation and profiling on Zed board. I would like to thank Mr.Rajesh Kedia for all his help and his encouragement throughout the thesis. Laslty I thank God for making this journey possible for me.

PAPPIREDDY DEDEEPYA

2014JVL2695

ABSTRACT

Signboard detection and information extraction aims to solve the hindered navigation in outdoor environment for a visually impaired user. Various image processing algorithms are used to extract sign board present in the input image. Extensive use of OpenCV functions is seen in the developed approach. The extracted signboard is further passed on to the information extraction stage where the text present on the sign board is identified with the help of an Optical Character Recognition Engine.

CONTENTS

1. Introduction.....1- 3

1.1. Introduction to MAVI SYSTEM.....1

1.2. Overview of signboards detection and info extraction.....2

1.3. Thesis outline.....3

2. Background.....4- 5

2.1. Introduction to opencv.....4

2.2. Object detection algorithms.....4

2.2.1. Colour based detection using opencv.....4

2.2.2. Contour based object detection 5

2.2.3. Conclusion.....5

3. Proposed algorithm for signboard detection.....6- 18

3.1. Overview of algorithm.....6

3.2. Stage 1:Blue detection stage.....7

3.3. Stage 2:White detection stage.....10

3.4. Modes of operation of signboard detection algorithm.....11

3.4.1. Default mode.....11

3.4.2. Low energy modes.....11

3.5. Experimental Results.....12

3.5.1. Accuracy of signboard detection.....12

3.5.2.Distance Vs accuracy of detection.....12

3.5.3. Accuracy in various modes of operation.....13

3.5.4. Performance estimation on Desktop.....13

3.5.5. Sampling Rate Vs Walking speed.....18

4.Information extraction.....19-25

4.1. Overview of Information extraction.....19

4.2. Stage 1:.....20

4.2.1. Preprocessing and noise removal.....20
4.2.2. Word extraction 21
4.2.3. Perspective transform.....21
4.3. Stage 2.....22
4.2.2. Detection and Removal of Header line.....22
4.2.3. Splitting the words into regions.....22
4.4. OCR stage.....23
4.4.1. Tesseract.....23
4.4.2. Hindi OCR Engine.....23
4.5. Results.....24
4.5.1. Accuracy.....24
4.5.2. Timing Estimates.....25
5.Software ONLY implementation on Zed board.....26-33
5.1. Introduction to Zed board.....26
5.2. Cross compilation of opencv and Linaro on Ubuntu.....26
5.3. Implementation of signboard detection on Linaro(Zed _board _implementation).....28
5.4.1. For NRS_BOTH mode.....28
5.4.2. For NRS_SINGLE mode.....29
5.4.3. For RS_BOTH mode.....30
5.4.3. For RS_SINGLE mode.....31
5.4. Comparison with desktop implementation.....32
6. Energy measurement for signboard detection.....34-40
6.1. Hardware setup for energy measurements.....34
6.2. Results for NRS_BOTH.....35
6.3. Results for RS_BOTH.....36
6.4. Results for NRS_SINGLE.....37
6.5. Results for RS_SINGLE.....39

7. Hardware Software CoDesign 41-44

7.1.Introduction.....41

7.2.Profiling.....42

7.3.Results.....43

8. Conclusion and Future Work 45

9.References.....46

List of figures

1.1.Overview of MAVI system.....1

1.2.Overview of sign board detection algorithm.....2

3.1.Signboard Detection algorithm.....6

3.2.1.Input image for signboard detection.....8

3.2.2.Thresholded image.....8

3.2.3.Bounding box drawn on the input image indicating sign board.....9

3.2.4.Extracted bounding box from the image.....9

3.3.1.Demonstration of White Detection Stage.....10

3.5.1.Performance estimates for NRS_BOTH mode True cases.....14

3.5.2.Performance estimates for NRS_BOTH mode False cases.....14

3.5.3.Performance estimates for NRS_SINGLE mode True cases.....15

3.5.4.Performance estimates for NRS_SINGLE mode True cases.....15

3.5.5.Performance estimates for RS_BOTH mode True cases.....16

3.5.6.Performance estimates for RS_BOTH mode False cases.....16

3.5.7.Performance estimates for RS_ SINGLE mode True cases.....17

3.5.8.Performance estimates for RS_SINGLE mode False cases.....17

4.1.Implementation of Information extraction.....20

4.2.1.Demonstartion of word extraction and Perspective Correction.....21

4.5.1 Demonstration of Tesseract engine(For English text) 25

5.2.1.Zync flow.....27

5.3.1.Performance estimates for NRS_BOTH mode True cases.....28

5.3.2.Performance estimates for NRS_BOTH mode False cases.....29

5.3.3.Performance estimates for NRS_SINGLE mode True cases.....29

5.3.4.Performance estimates for NRS_SINGLE mode False cases.....30

5.3.5.Performance estimates for RS_BOTH mode True cases.....30

5.3.6.Performance estimates for RS_BOTH mode False cases.....31

5.3.7.Performance estimates for RS_ SINGLE mode True cases.....31

5.3.8.Performance estimates for RS_SINGLE mode False cases.....32

6.1. Setup for Energy Estimation.....34

6.2.1.Voltage graph obtained for TRUE_NRS_BOTH mode.....35

6.2.2.Voltage graph obtained for FALSE_NRS_BOTH mode.....36

6.3.1. Voltage graph obtained for TRUE_NRS_SINGLE mode.....36

6.3.2.Voltage graph obtained for FALSE_NRS_SINGLE mode 37

6.4.1.Voltage graph obtained for TRUE_RS_BOTH mode.....38

6.4.2. Voltage graph obtained for FALSE_RS_BOTH mode.....38

6.5.1.Voltage graph obtained for TRUE_RS_SINGLE mode.....39

6.5.2.Voltage graph obtained for FALSE_RS_SINGLE mode.....40

7.1.Levels of Abstraction of HW/SW Codesign.....41

7.2 Hardware Software CoDesign flow.....42

7.3.1.Profiling results for NRS_BOTH mode.....43

7.3.2.Profiling results for NRS_SINGLE mode.....43

7.3.3.Profiling results for RS_BOTH mode.....44

7.3.4.Profiling results for RS_SINGLE mode.....44

List of Tables:

Table 3.5.1.Accuracy of Sign Board Detection.....12

Table 3.5.2.Analysis of Failure cases for distance greater than 5m.....13

Table 3.5.3.Comparision of Performance for various modes in Desktop implementation.....13

Table 3.5.4.Comparision of Performance for various modes in Desktop implementation.....18

Table 3.5.5 .Estimation of Sampling Rate Vs Walking Speed.....18

Table 5.4.1 Performance Estimates on Zed Board Implementation.....32

Table 5.4.1.Comparision of Performance on both the platforms.....33

Table 6.2.1.Energy measurements for NRS_BOTH mode for TRUE cases.....35

Table 6.2.2.Energy measurements for NRS_BOTH mode for FALSE cases.....35

Table 6.3.1.Energy measurements for RS_BOTH mode for TRUE cases.....36

Table 6.3.2.Energy measurements for RS_BOTH mode for TRUE cases.....37

Table 6.4.1.Energy measurements for NRS_SINGLE mode for TRUE cases.....37

Table 6.4.2.Energy measurements for NRS_SINGLE mode for FALSE cases.....38

Table 6.5.1.Energy measurements for RS_SINGLE mode for TRUE cases.....39

Table 6.5.2.Energy measurements for NRS_SINGLE mode for FALSE cases.....39

Table 6.6.Comparison of Energy measurements for all modes.....40

CHAPTER 1:

INTRODUCTION:

1.1.Introduction to MAVI:

The two important problems faced by the visually impaired include Obstacle Detection and Independent Navigation. With the advent of technology, cost effective solutions like Smart Cane [1] , have been developed to solve the obstacle detection problem. However independent Navigation especially in the Outdoor environment , is still an unresolved issue for the visually impaired. In general, Safety issues, Navigation problems in an unknown area and social inclusion are the major issues of concern . MAVI system attempts to solve these issues by integrating various modules which work to solve problems posed by outdoor navigation independently. MAVI stands for Mobility assistant for visually impaired . The overview of the MAVI system is shown in the below figure 1.1.

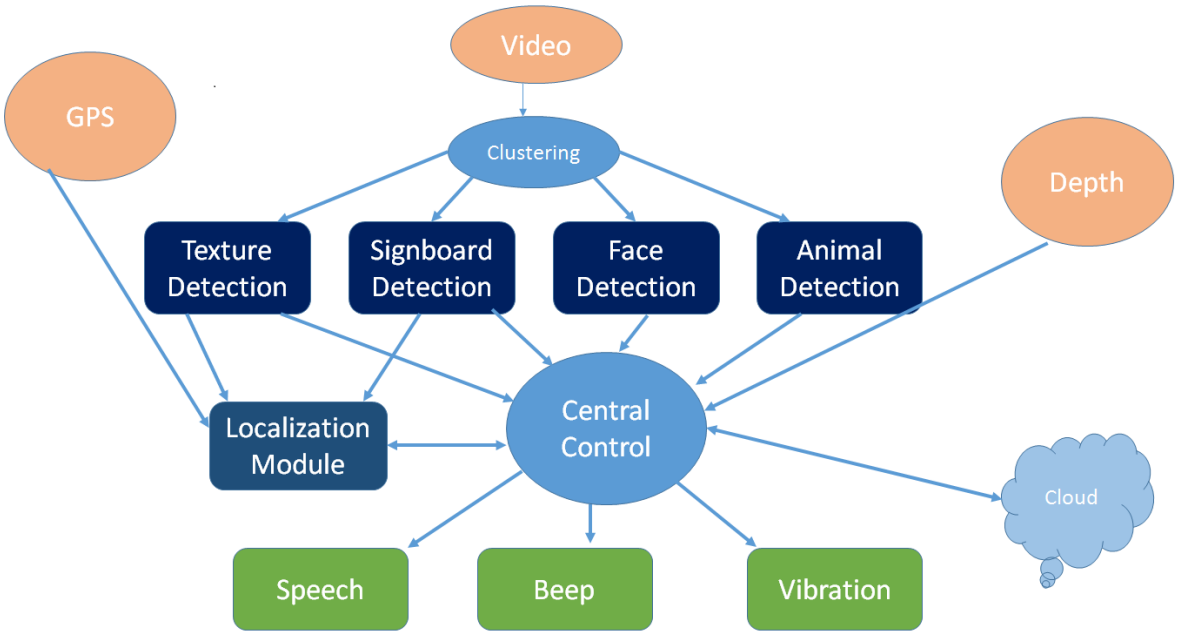


Fig.1.1 Overview of MAVI system

The MAVI system constitutes of a central controller that binds the four modules of MAVI system together. These modules are Texture detection,

Animal detection, signboard Detection ,Face detection and recognition. The controller also has the information from the GPS+IMU localisation module and depth sensor. It also interacts with the server and MAVI app to provide information to the user.

1.2.Introduction To Sign Board Detection and Information Extraction:

Signboards serve as navigation aids in an unknown location. The information present on them becomes extremely important for a person who is not the native of that place. For visually impaired people, this information remains unavailable unless a module is employed exclusively to provide the information. Hence signboard detection and Information extraction module focuses to assist the user in navigating seamlessly in an outdoor environment. The overall implementation of the Sign Board detection module is as shown below.

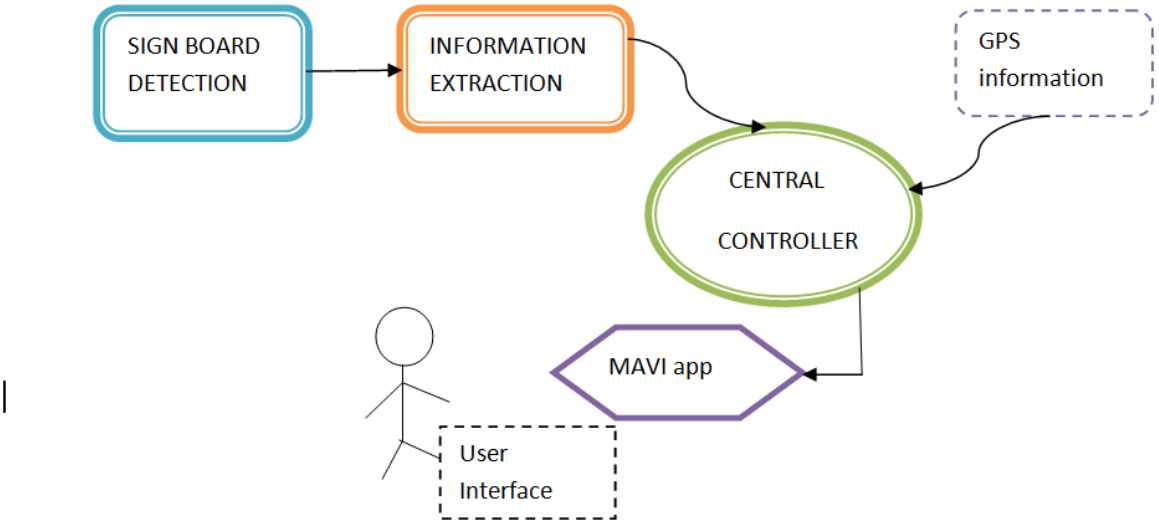


Fig.1.2 Overview of Sign Board Detection Module

1.3.Thesis Outline:

The body of the thesis is organised into 7 chapters out of which the first chapter is the introduction .The remainder of the thesis is organised as follows: The chapter 2 includes the introduction to OpenCV and briefly outlines the Detection Algorithms that are viable for the detection of sign boards. The

chapter 3 discusses the algorithm employed for signboard detection in MAVI system and its associated results, followed by chapter 4 which discusses the information extraction with the help of OCR Engine. The chapter5 describes software only implementation on Zed board and the performance estimation results for this implementation .Chapter6 is about the energy estimation for signboard detection module in various modes. Chapter 7 discusses the profiling results.Chapter8 is about the conclusion and future work.

CHAPTER 2[2]

BACKGROUND:

2.1.Introduction to OpenCV :

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real time computer vision, developed by Intel .It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing. The library was originally written in C and this C interface makes OpenCV portable to some specific platforms such as digital signal processors. Wrappers for languages such as C#, Python, Ruby and Java (using JavaCV) have also been developed. However, since version 2.0, OpenCV includes both its traditional C interface as well as a new C++ interface. This new interface seeks to reduce the number of lines of code necessary to code up computer vision functionality as well as reduce common programming errors such as memory leaks (through automatic data allocation and de-allocation) that can arise when using OpenCV in C. Most of the new developments and algorithms in OpenCV are now developed in the C++ interface .Unfortunately, it is much more difficult to provide wrappers in other languages to C++ code as opposed to C code; therefore the other language wrappers are generally lacking some of the newer OpenCV 2.0 features. A CUDA-based GPU interface has been in progress since September 2010.

2.2.Object Detection Algorithms:

2.2.1.**Template matching:**

Template matching is a technique in which a portion of the image or sub-image ,called template is searched in a given input image containing the sub-image to find its occurrence.

Algorithm:

1.It takes an source image and template image as input

2.The identification of the matching region the template is compared with the source image by sliding it.

2.The identification of the matching region the template is compared with the source image by sliding it.

3. Sliding, refers to moving one pixel at a time from left to right, up to down.At each location the measurement of match is done by calculating a metric.

4. For each location of **T** over **I**, the metric is calculated and stored in the *result matrix* **(R)**.Here the match metric **TM_CCORR_NORMED** is used. method=CV_TM_CCORR_NORMED .

2.2.2 Color Based Detection using OpenCV:

Detecting objects based on their colour is one of the prominent way of object detection. In this case the the colour of the object should significantly different from the colour of the background.[3]

Algorithm :

1.Creat an image window.
2.Creat track bars in the window for Hue,Saturation,Value.
3.Read the image in BGR/RGB format.
4.Convert the image from RGB to HSV format.
5.Threshold the Image
6.Doing morphological opening/closing(optional)
7.Display the thresholded and original image.

2.2.3 Contour based object detection:

OpenCV can be used for obtaining the vertices as a series of points of an object. So, a polygon can be determined by by getting the number of vertices present in it. For example a rectangle can be classified under an object with 4 vertices present in it. Also several features of the polygon such as convexity, flatness, concavity by comparing the position and distance between the vertices in an object

Algorithm:

1.Create an image window.
2.Read the image.
3.Convert the image(if colour image) from RGB to HSV format.
4.Threshold the Image to get a binary image i.e. black(0) and white(1).
5.Doing morphological opening/closing(optional)
6.Search for contours in the thresholded image using OpenCV functions.

2.3 CONCLUSION:

The proposed algorithm is a combination of the above techniques and is mentioned in chapter 3.

CHAPTER 3 :

ALGORITHM FOR SIGNBOARD DETECTION

The algorithm of signboard detection for MAVI is designed for the campus signboards of IIT Delhi as of now. The algorithm assumes that the boards are blue in colour with the text being written in white colour. The algorithm is constructed in two serial stages. The blue detection stage followed by the white identification stage. The algorithm is implemented in OpenCV 3.1.The input image is taken to be of vga resolution.(A 640x480 image is used.)

3.1.Overview of Algorithm:

The following figure 3.1 depicts the overview of algorithm:

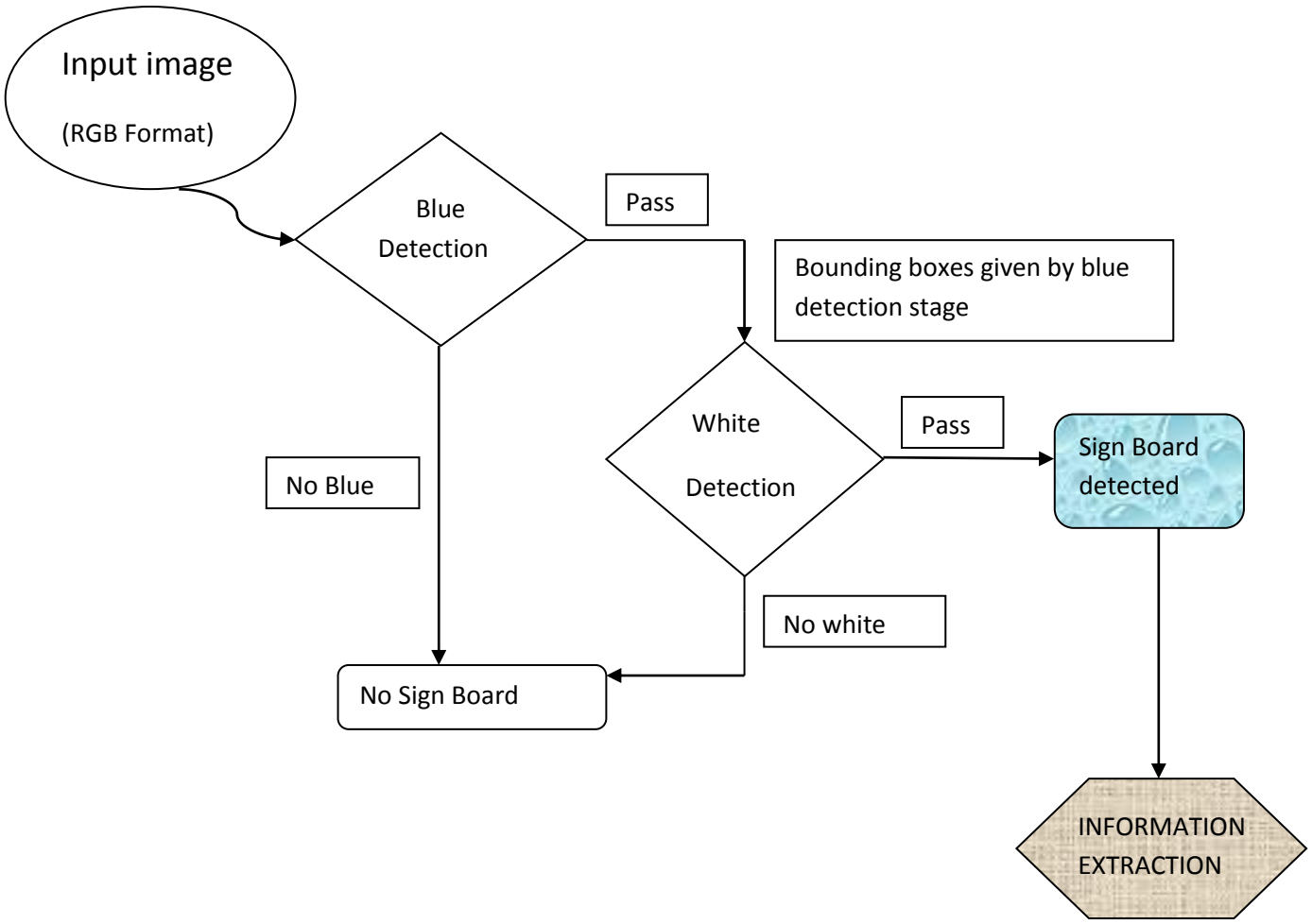


Fig. 3.1 Overview of Sign Board detection

3.2. Blue Detection Stage:

The steps performed in Blue detection stage are as follows:

1.Read the input image in RGB format

2.Transform the image from RGB colour space to HSV colour space using the OpenCV function CV_BGR2HSV.

3.Generate the mask for the Blue colour present in the image using the InRanges function of OpenCV and also performing BitWiseAND operation to remove any residual blue.

4.On the result obtained find the contours using FindContours function of OpenCV.

- While using this function , the contour retrieval mode is set to the CV_RETR_TREE so as to retrieve all of the contours and reconstruct a full hierarchy of nested contours.[]
- The contour approximation method used is CV_CHAIN_APPROX_SIMPLE which compresses Horizontal, Vertical and diagonal segments and leaves only end points.

5.The contours obtained are sorted in the descending order of their areas.

6.These sorted contours are further approximated into closed closest regular polygon .This is accomplished by using the approxPolyDp function of OpenCV.

- The approxpolyDp function is used to approximate a curve or polygon with another curve or polygon with less vertices so that the distance between them is less than or equal to the specified precision.[]
- It uses Douglas-Peucker algorithm.

7. Calculate the Up Right Rectangle on the obtained contour point set i.e on the approximated polygon try to fit a Rectangle.

8.Group the rectangles which lie inside each other(if present) as a single rectangle.

9.Return the two largest possible rectangles in the image.

These rectangular boxes are further passed on to the white detection stage for presence of text verification.

The following figures illustrate the implementation of the algorithm:

Fig.3.2.1.Input image for Blue detection

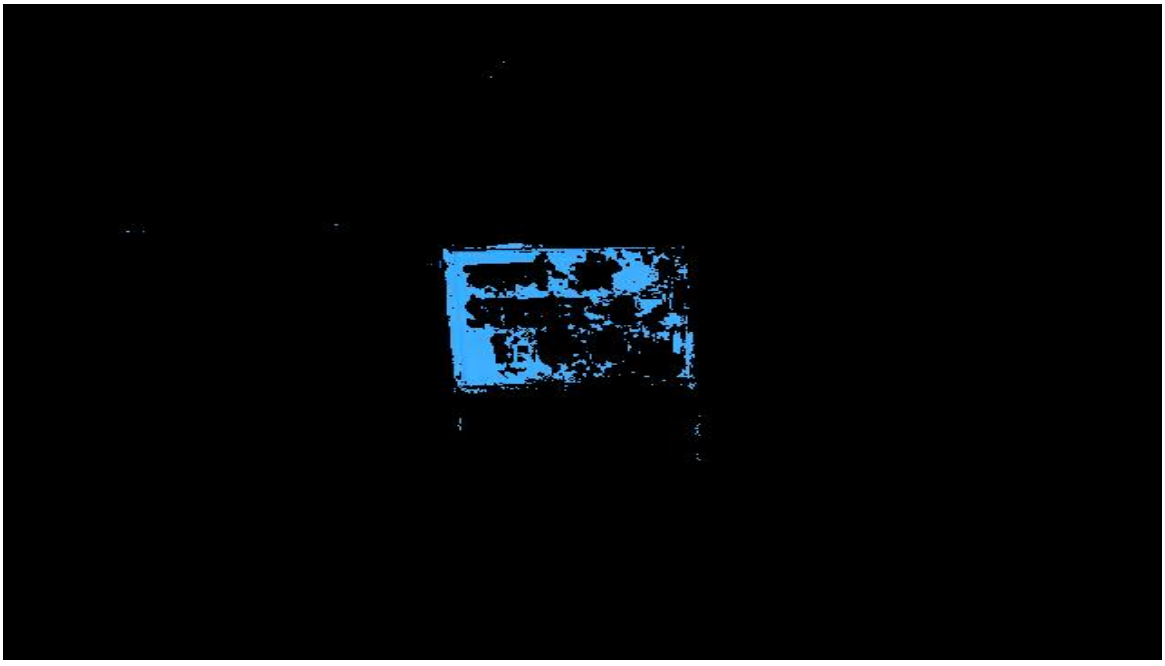


Fig.3.2.2.Thresholded Image

Fig.3.2.3.Bounding Box obtained after blue detection (Drawn on input image)

Fig.3.2.4. Extracted Bounding Box (Input to White Detect stage)

3.3.White Detection stage:

1.The bounding boxes returned by the blue detect are passed on to this stage as input.

2.Here we apply the flood fill technique so as to remove the boundary noise assuming that the noise is to the exterior of the board and at the border os the bounding box.

3.On the resultant image we calculate the amount of white by counting the number of white pixels.

4.Then a ratio of this amount of white to the total number of pixels in the image is obtained and compared against a predefined threshold.

5.If the amount of the white in the image exceeds this preset threshold then the bounding box is passed out as a Possible Sign board

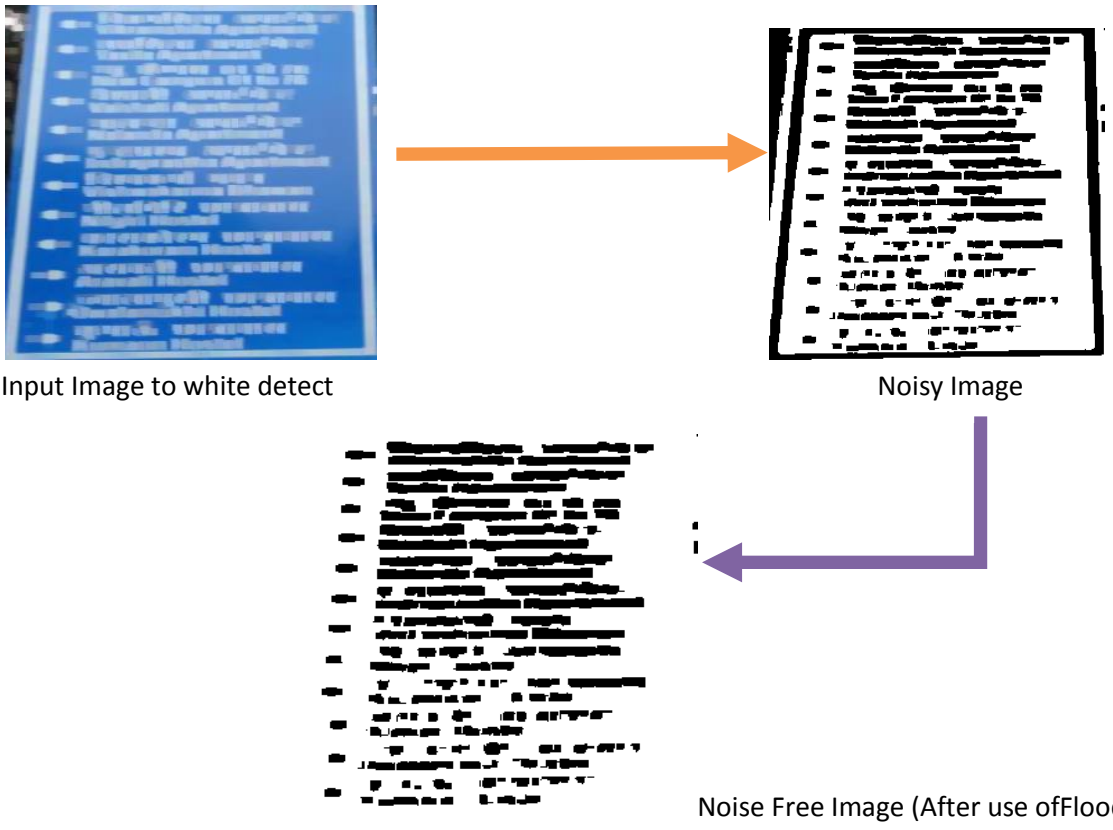


Fig.3.3.1.Demonstration of White Detection Stage

3.4.Modes of Operation of Signboard Detection Algorithm :

3.4.1.Default mode:

Here the algorithm is implemented without any change from the discussed one i.e. , the Blue detection is performed on the input image and both the output boxes returned by the blue detect are passed on as input for white detection stage. And the information extraction stage by default employs the use of OCR engine. This is referred as NRS_BOTH mode.

3.4.2.Low Energy mode:

This mode is further classified into 3 modes , each of them having a difference in processing in either blue detection or white detection or both of them.

- Mode a: [NRS_SINGLE]

  This NRS_SINGLE stands for No Resizing in Blue detect and Single input to white detect. Here the blue detection stage is the same as mentioned in the default mode. However in this mode ,if the blue detect returns two bounding boxes to be possible sign boards ,still the white detect is directed to work only on the First Maximum box (The box with maximum area).This is based on the observation that the smaller boxes are at farther distance from the user and the information loss is insignificant. Thus as the processing time comes down ,eventually the energy is saved. If the sign boards of similar sizes are present close to each other then there is information loss at significant amount.
- Mode b: [RS_BOTH]

  Here the input image is resized before passing on to the Blue detection stage. From the results of profiling at software level it is observed that the blue detect takes significant amount of time .Also the accuracy measurements have shown that the algorithm works well even when the image is resized to 320x240 . The white detection stage processes the outputs returned by blue detect as in the default mode.

  Thus in scenarios where the location is well mapped and the controller decision excludes the use of OCR engine for information extraction, this mode can be employed. The processing time drastically reduces in this mode.

- Mode C: [RS_SINGLE]

  In this mode the input to blue detect stage is resized and also the white detection stage is directed to work on the single box even if two boxes are passed on from blue detect as output to the white detect stage. This mode suffers from loss of information as in case of mode a . Also this mode results in more False Negatives.

3.5.Experimental Results :

3.5.1.Accuracy of Sign Board Detection:

Data Set employed : Data Set1

Total No. of Images = 2957

No. of Images with sign boards=1724

No. of images without sign boards= 1233

Type	Number	Percentage
True Positives	1658	96.1721
True Negatives	1168	94.7283
False Positives	65	5.27169
False Negative	66	3.82830

Table 3.5.1. Accuracy of Sign Board Detection

3.5.2. Distance Vs Accuracy of Sign Board Detection:

Data set employed :Data set 2

For distance between 1m and 4m :

Total No. of Images(with sign board) = 1804

Detected = 1694

However for distance less than 1m (typically from 0.6m) the accuracy diminishes very much. For distance greater than 5m also the algorithm fails.

Failure Analysis:(from 4m to 5m)

Images with sign board = 240

1.	No. of images detected	27
2.	No .of False cases	213
	2.a)False_contours	35
	2.b)False _blue_input	68
	2.c)False_White_input	23
	2.d)False_white_area	45

Table 3.5.2. Analysis of Failure cases for distance greater than 5m

3.5.3.Accuracy in various modes of operation:

Dataset used : Dataset 3

MODE	**TOTAL TRUE CASES** 1698		**TOTAL FALSE CASES** 1143	
	TRUE	**FALSE**	**FALSE**	**TRUE**
NRS_BOTH	1484	214	1066	77
NRS_SINGLE	1488	210	1066	77
RS_BOTH	1459	239	1108	35
RS_SNGLE	1461	237	1108	33

Table 3.5.3. Comparison of accuracy for various modes in Desktop implementation

3.5.4. Performance Estimation on Desktop:

Implementation On Desktop :

- System Configuration used: i5 processor @ 3.3GHz Clock
- Tool employed: perf
  All graphs below have the X-axis as time elapsed, measured in milliseconds.

3.5.4.1. For NRS_BOTH mode:

This is the default mode of operation.

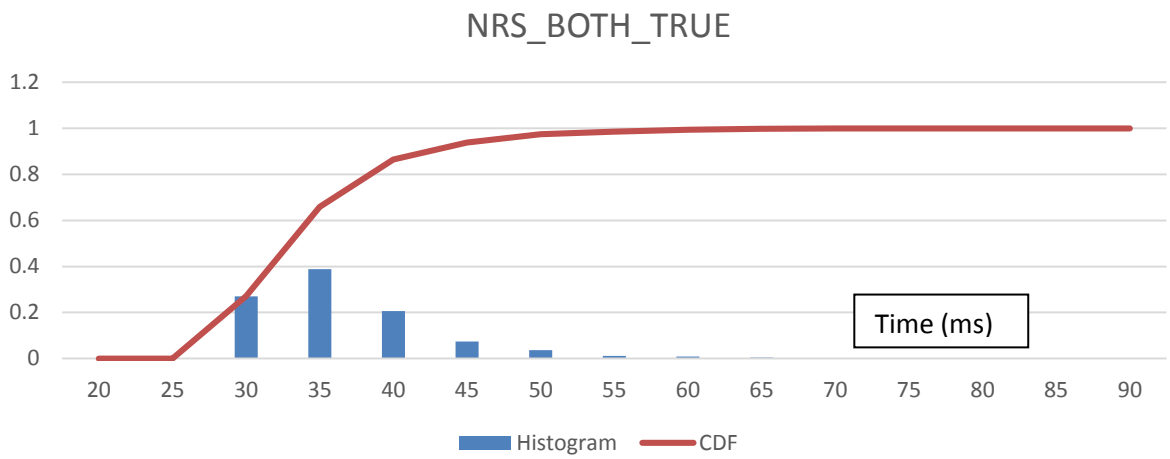


Fig.3.5.1 .Performance (Time estimation) for Default mode true cases

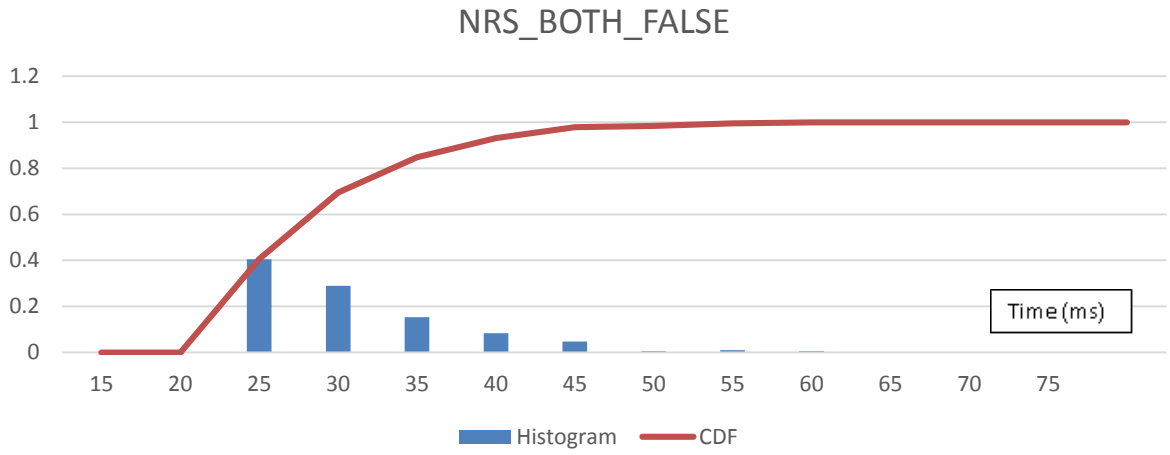


Fig.3.5.2.Performance (Time estimation) for Default mode false cases

3.5.4.2. For NRS_SINGLE mode:

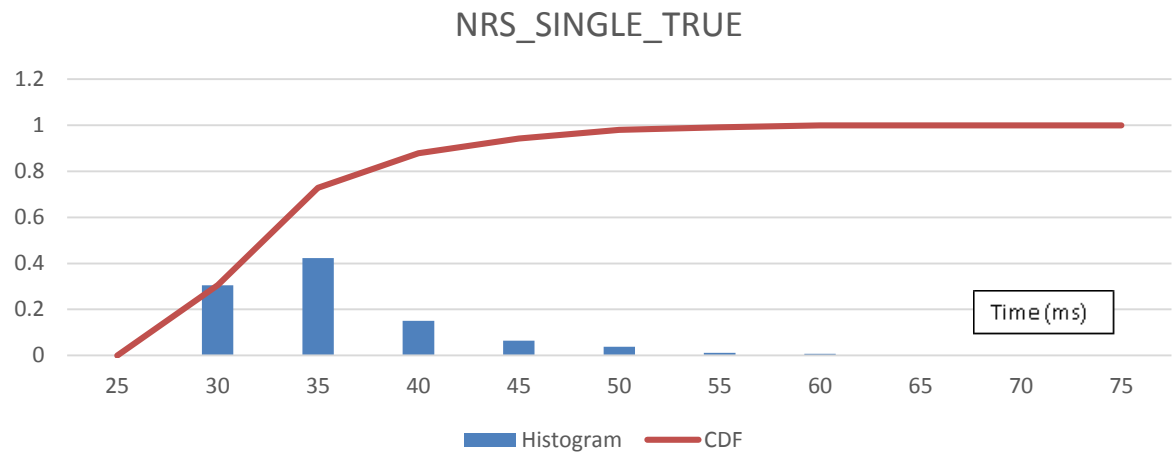


Fig.3.5.3.Performance (Time estimation) for NRS_SINGLE mode True cases

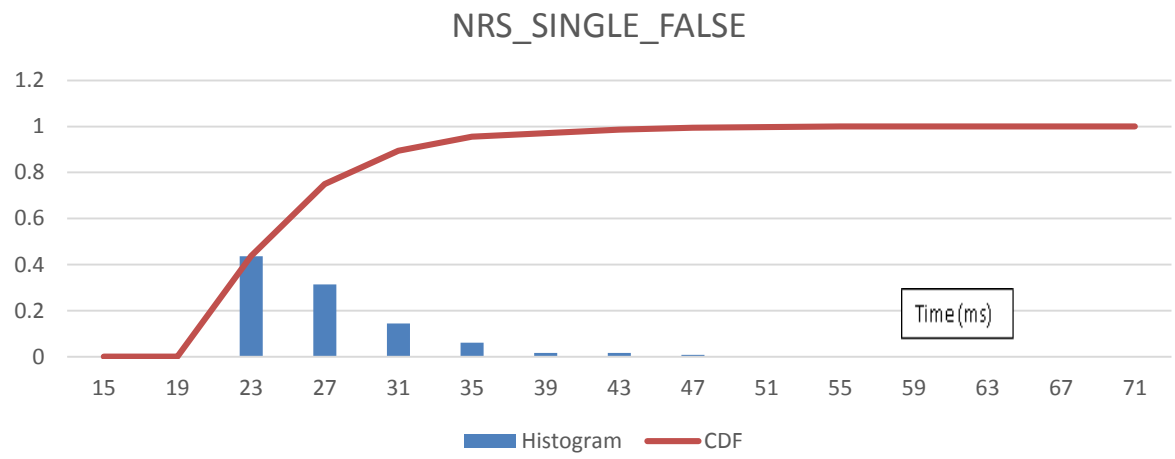


Fig.3.5.4.Performance (Time estimation) for NRS_SINGLE mode False cases

3.5.4.3. For RS_BOTH mode:

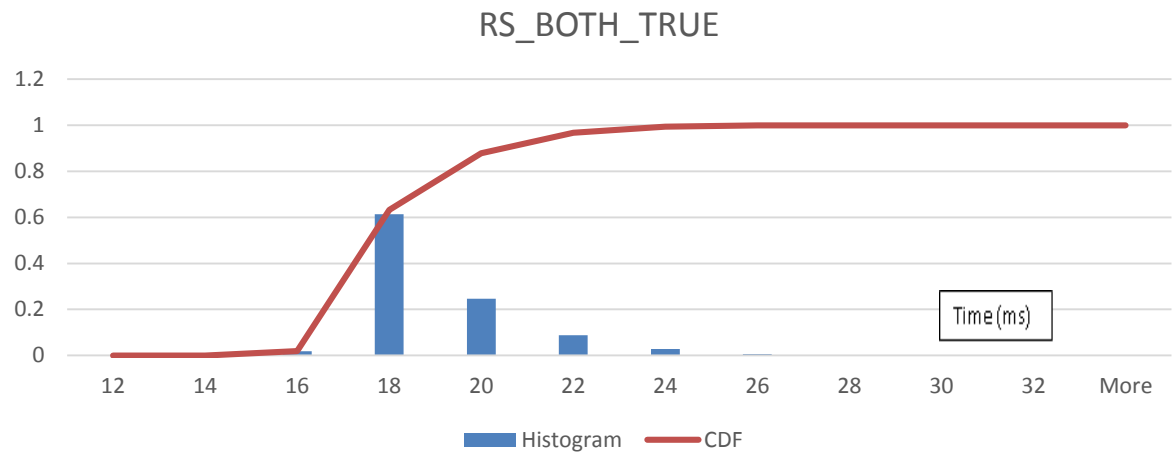


Fig.3.5.5.Performance (Time estimation) for RS_BOTH mode True cases

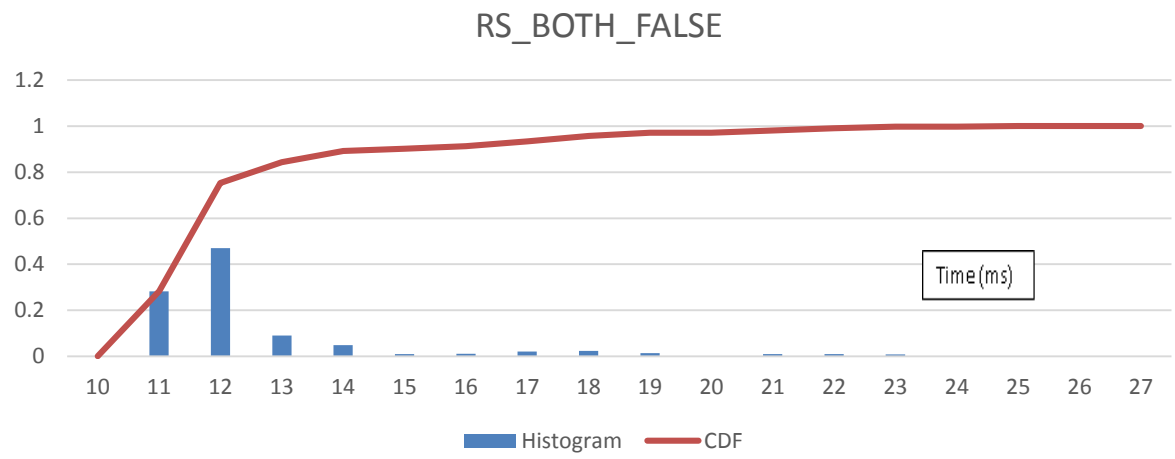


Fig.3.5.6. Performance (Time estimation) for RS_BOTH mode False cases

3.5.4.4. For RS_SINGLE mode:

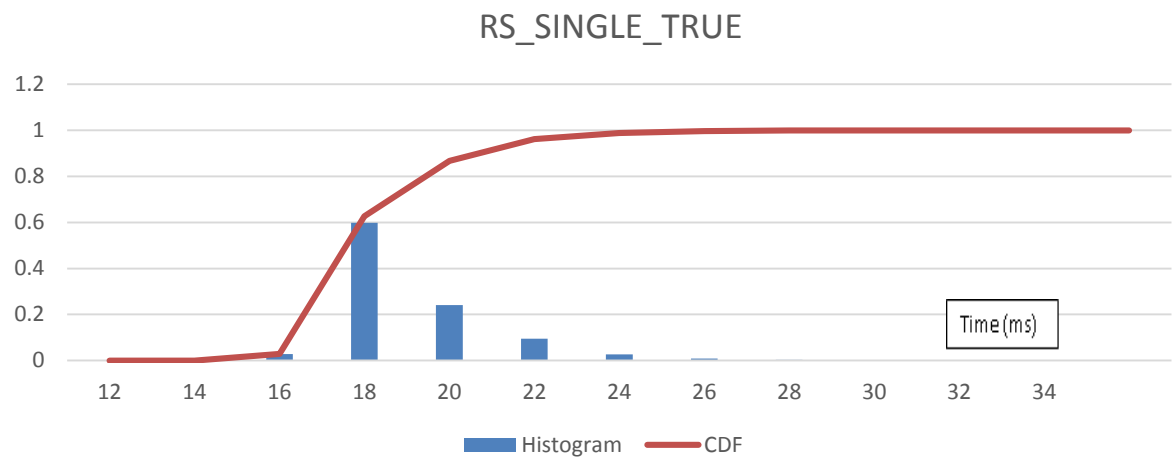


Fig.3.5.7.Performance (Time estimation) for RS_SINGLE mode True cases

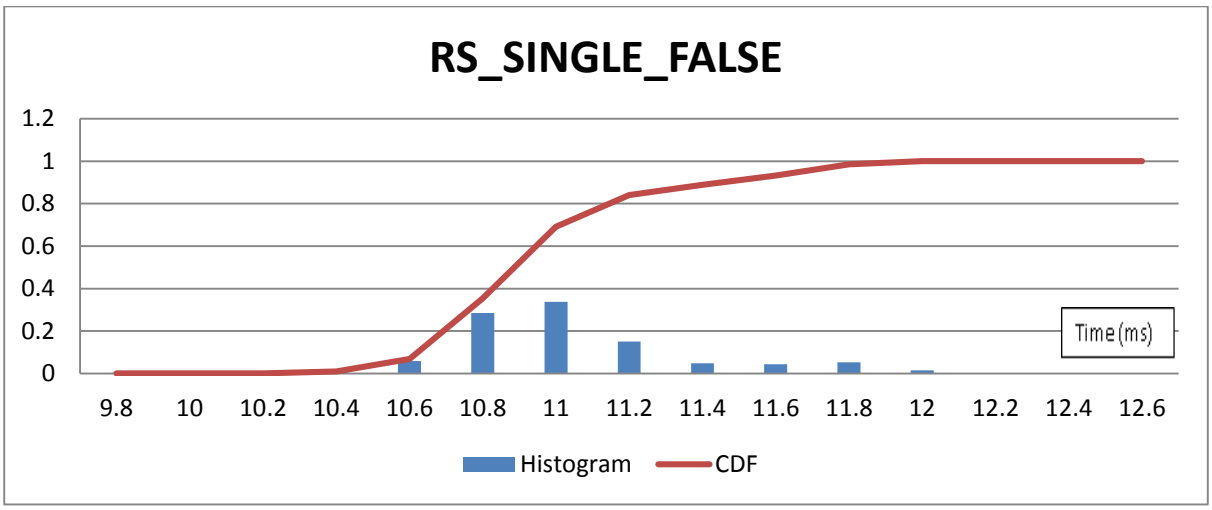


Fig.3.5.8. Performance (Time estimation) for RS_SINGLE mode False cases

Summary of Performance estimates in various modes of operation:

Data set employed :Data set 3

MODE		Desktop Implementation time (ms)				
		Minimum	Maximum	Average	90% point	95% point
NRS_ both	True	27.3116	70.3676	34.2544	43.4589	45.3751
	False	22.0576	57.2628	28.6954	37.4591	42.8546
NRS_ Single	True	27.1469	59.9157	32.4221	41.5014	45.3128
	False	21.4493	54.4690	25.3344	31.8567	33.1587
RS_ Both	True	15.7827	30.7485	17.9652	20.4575	22.8675
	False	10.1683	24.3883	12.1194	16.1458	17.1247
Rs_ Single	True	15.5850	26.8265	17.9232	20.1287	22.2398
	False	10.3443	11.9982	10.9461	11.4258	11.6257

Table 3.5.4.Comparision of Performance for various modes in Desktop implementation

3.5.5. Sampling Rate Vs Walking Speed:

The experiment is conducted in the default mode. Here the Normally populated sign boards refer to the place where a signboard occurs for a distance at least greater than 8 metres. If the signboards are separated by a distance less than 8 metres, they are taken as densely populated .The following table 5.3 estimates the maximum sampling rate for various walking speeds. If the sampling rate exceeds the tabulated value then the accuracy of Sign Board Detection falls drastically.

Walking Speed(Metres/Second)	**Frame Rate(Samples/Second)**	
	Normally populated Sign Boards	**Densely Populated Sign Boards**
Highest walking speed **(0.5mps)**	1 frame in 3 seconds 0.333 fps	1 frame in 3 seconds 0.333 fps
Normal walking speed **(0.25mps)**	1 frame in 4 seconds 0. 25 fps	1 frame in 2seconds 0.50 fps
Lowest walking speed **(0.1mps)**	1 frame in 6 seconds 0.15 fps	1 frame in 6 seconds 0.15 fps

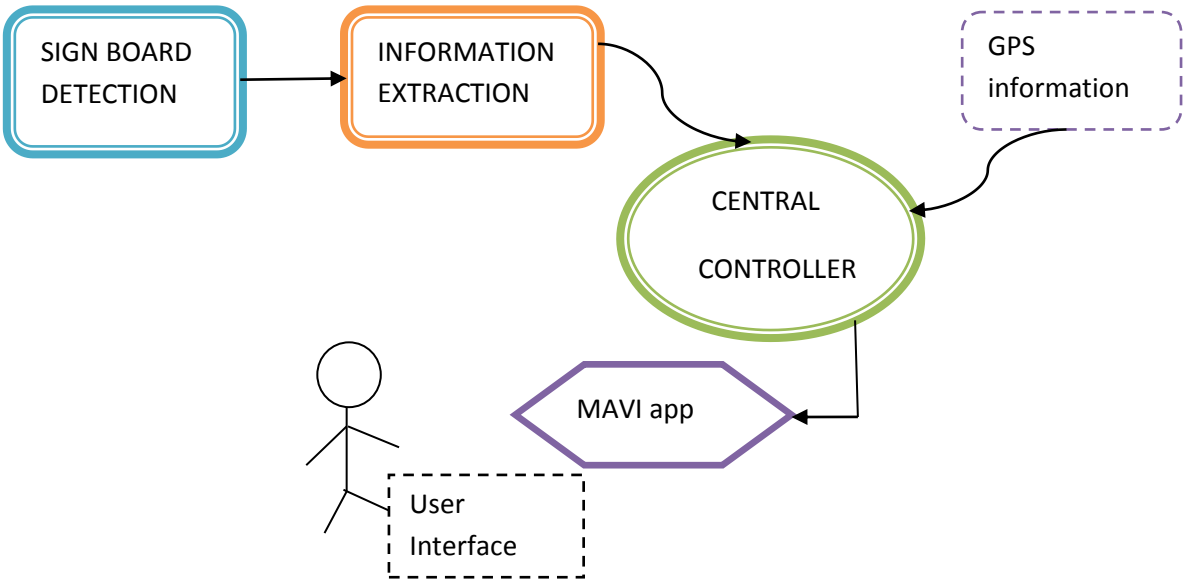
Table 3.5.5 Trade off between Sampling Rate Vs Walking Speed

CHAPTER 4 :

INFORMATION EXTRACTION:

After the signboard is detected by the sign board detection algorithm , it needs to be processed for the information present on it. This stage has to pass the text on the sign board to the controller and the controller gives this information to the user in voice format via MAVI app.

The overview of the entire processing (figure 1.2 replicated) is depicted below :



4.1.Overview of Information extraction:

The Information extraction stage is organised into various steps like Noise removal , ,word extraction, Rectification of words ,character extraction and passing it to the OCR engine. The sign boards are assumed to be bilingual and depending on the language(Hindi or English) we choose the OCR engine. The entire Information extraction is implemented in MATLAB. The Tesseract engine is employed for the English text identification where as the Hindi OCR has been developed .The implementation is shown in Fig.4.1.

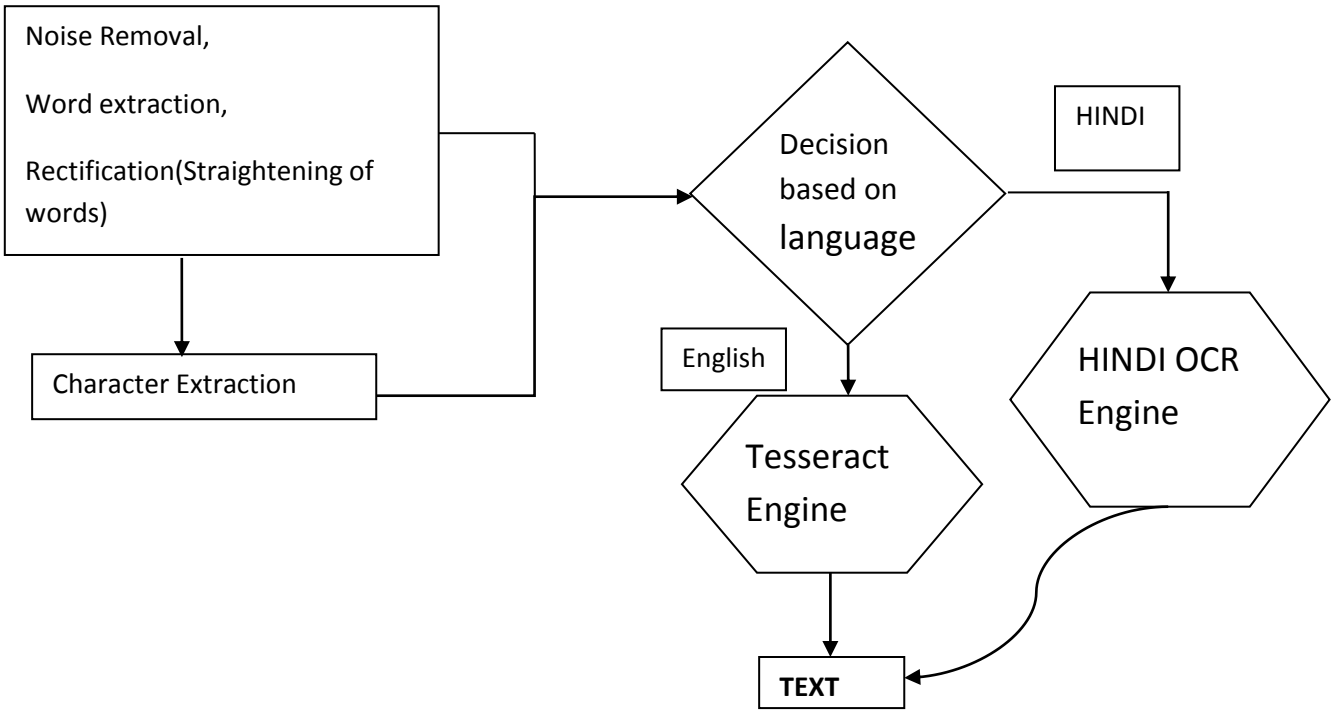


Fig.4.1.Implementation of Information extraction

4.2. Stage1:

4.2.1.Pre-processing and Noise Removal:

- The input image present in the RGB Domain is converted into HSV domain.
- Obtain the saturated image using the HSV image.
- Convert the saturated image into a Black and White image using the global threshold level of the saturated image.
- The obtained black and white image has text in white colour and the board in black colour.(The background noise is also white in colour)
- Negate the Black and White image so as to obtain the text in Black colour and Board in white colour.
- On the resultant image perform the morphological operations and obtain the connected components.
    - These connected components include both the words from the text of the sign board and noise components
- Using the major axis and minor axis lengths of the connected components ,we discard some of the Noise components from the image.

- Also, the area of the noise components is observed to be very high compared to that of the words .Hence using the area parameter we can remove large noisy components.
- The granular noises are also removed by Area Opening operation.

4.2.2.Word extraction:

- The Black and white image is now left with the words constituting the text of the sign board as we have possibly eliminated all the noise components.
- These words are stored in a collection along with their Height and aspect ratios.
- Any extracted component which is has height more than the average height is split based on the average height.

4.2.3.Perspective Transform:

- The word which has the maximum aspect ratio is identified.
- By using this word as reference, the orientation matrix for the rectification is calculated using Hough line and Hough Peaks .
- The obtained transformation matrix is applied to all the words.
- These straightened words are sent for character extraction.

Fig.4.2.1.Demonstartion of word extraction and Perspective Correction

4.3. Stage2:

Character Extraction :

This stage takes logical image of each connected component corresponding to text in the signboard image and returns the individual characters. Stage 1 returns Words in case of Hindi text and Letters(characters) in case of English text. For a Hindi word, the character extraction is carried out as follows :

4.3.1. Detection and removal of header line:

This step is based on the assumption that header line in Hindi word has the maximum horizontal projection profile. Further, for the entire width of header line, the projection profile varies within 10% of maximum value. Using this approach, we localized the header line.

4.3.2. Splitting the word into regions:

Given the location of header line, word can be divided into 2 regions- above and below the line. Depending upon width of region and number of white pixels in the upper region, presence of upper modifiers is determined. If upper modifiers are present, they are extracted using connected components algorithm. In the region below the header line, connected components are extracted and each such component is a character.

Now, based on the height of characters in the region below header line, presence of lower modifiers is determined. The maximum height among all characters is calculated and two groups of characters are formed: one with the characters having height within 80% of maximum and the other containing characters having height less than that. If the former group has more members than the latter, no lower modifier is present else the former group contains characters with lower modifiers. These characters are split at maximum value of height in the other group.

4.4.OCR Stage:

Optical character recognition (optical character reader) (**OCR**) is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example from a television broadcast). [4]It is widely used as a form of data entry from printed paper data records, whether passport documents, invoices, bank statements, computerised receipts, business cards, mail, printouts of static-data, or any suitable documentation[5]. It is a common method of digitising printed texts so that it can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as cognitive computing ,machine translation, (extracted) text to speech , key data and text mining .[4] .OCR is a field of research in pattern recognition, artificial intelligence, and computer vision [5].

In this work we are employing TESSERACT[6] for recognising the English text and for Hindi text we are have implemented our own OCR.

4.4.1.Teeseract:

Tesseract is an OCR engine for various operating systems.[6]. It is a free software, released under the Apache License, Version 2.0,[6] and development has been sponsored by Google since 2006.[6] . It was developed at Hewlett Packard Laboratories betwee-n 1985 and 1995.Tesseract OCR Engine works well for English text written in digitised format. However in case of the hand painted signboards the text identification is very inaccurate. For Hindi text recognition use of Tesseract is not a viable option.

4.4.2.Hindi OCR Engine:

(I).Data Acquisition :

A data set of Hindi characters has been made by extracting characters from sign boards and also by manually creating Hindi characters using Adobe Photoshop. This data set is split into a 70:30 ratio for training and testing

purposes. The training data set is made of 1022 digitised characters .However the test data base as of now has 1002 characters.

(II)Algorithm for Training and testing:

- Read the black and white single character image
- Then it is resized to a 16x16 image.
- Extract the features of connected components like centroid , Eccentricity, Solidity, EquivDiameter, Euler number.
- Apply two dimensional FFT to the black and white image and extract the orientation and size invariant inverse moments.
- Generate the overall feature vector for this each image and represent it in matrix form.
- Train the linear SVM from vlfeat library.

III).SVM FEATURES:

Linear SVM is employed with 55 character classes .The weight is chosen to be 50 and bias is chosen to be 1. The features employed are centroid , eccentricity, solidity, equivDiameter, Euler number ,extent.

Maximum number of iterations : 10 million

SVM converges between 9000 to 10000 iterations.

The implementation of SVM from vlfeat library is used in this work.

4.5.Results:

4.5.1.Accuracy:

English Text :

The Tesseract engine works only for the digital Sign board images. Even in this case the accuracy of detection depends on various factors , Font size used , Amount of spacing between the characters , distance from which the image is taken. For a distance of greater than 1.2metre, the accuracy is very poor. The figure 4.5.1 shows the output of Tesseract engine(For English text) .

The data base currently has 13 unique digitised sign boards out of which 9 were correctly identified by the Tesseract with less than 28% of error. Here

error is defined as 1-(Total Number of characters identified/Total number of characters present in the text).

Fig.4.5.1 Demonstration of Tesseract engine(For English text) .

Hindi Text:

The data base for Hindi OCR is relatively very small and the results are 98% accurate. This accuracy is only for digitised Sign boards and also the accuracy may fall when trained and tested on large data set.

4.5.2.Timing Estimates:

Specifications: Intel i7 8GB RAM 3.4GHz

Text Identification Stage :239.367ms(Includes feature extraction and SVM prediction)

For word and character extraction the time elapsed varies with the image, amount of text written (number of characters)and number of modifiers also in case of Hindi text.

Average time elapsed for

1.Word extraction = 260.81

2.Character extraction = 301.59

CHAPTER 5:

SOFTWARE IMPLEMENTATION ONLY ON ZED BOARD

5.1.Introduction To Zed Board:

Zed Board (Zynq Evaluation and Development Board) is an excellent development kit based on the Xilinx ZynqTM-7000 All Programmable SoC (AP SoC). It is a collaboration of three vendors Xilinx (Zynq AP SoC), Digilent (board manufacturer)and Avnet (distributor). This product integrates Xilinx Series7 programmable logic(PL) and a feature rich dual-core ARMCortex A9 MPCore based processing system (PS) in a single device having high performance and low power process technology [12].The Zed Board has a robust mix of on-board peripherals and also provides expansion capabilities which makes it an excellent platform for designing. The features provided by the Zed Board consist of:

- Xilinx Zynq XC7Z020-1CSG484 EPP (extensible processor platform) containing Dual-core ARM Cortex-A9 (PS) and Artix-7 FPGA (PL).
- Memory: 512 MB DDR3 and 256 Mb QSPI Flash
- On-board oscillators: 33.333 MHz (PS), 100 MHz (PL).
- Interfaces like USB-JTAG Programming, Ethernet, USB OTG, USB-UART bridge,SD Card, Pmod compatible headers, Eight Dip/slide switches, Nine User LEDs (1 PS, 8 PL) , Seven Push buttons etc.
- Display: HDMI output, VGA (12-bit colour) ,128x32 OLED Display Audio Line-in, Line-out, headphone, microphone
- Power: 12 V @ 5A AC-DC regulator

5.2.Cross compilation of OpenCV & Linaro Ubuntu:

Zed Board has a dual-core ARM Cortex A9 MPCore processor. Hence the object file that is supported by the arm processor is to be generated. An SD card of 8GB size is used to boot the Linux Operating system. The booting requires the SD card size to be of 8GB .The partitioning of SD card into two sections is done as follows:

1. FAT file system which contains boot image, kernel image and device tree file.
2. Ext4 file system which contain Linaro distribution. Zed Board boots over a number of stages.

The booting stages of Zed Board are shown in figure 6:1. Some of important files in the SD card are FSBL, U-Boot, Device tree, Kernel Image etc.

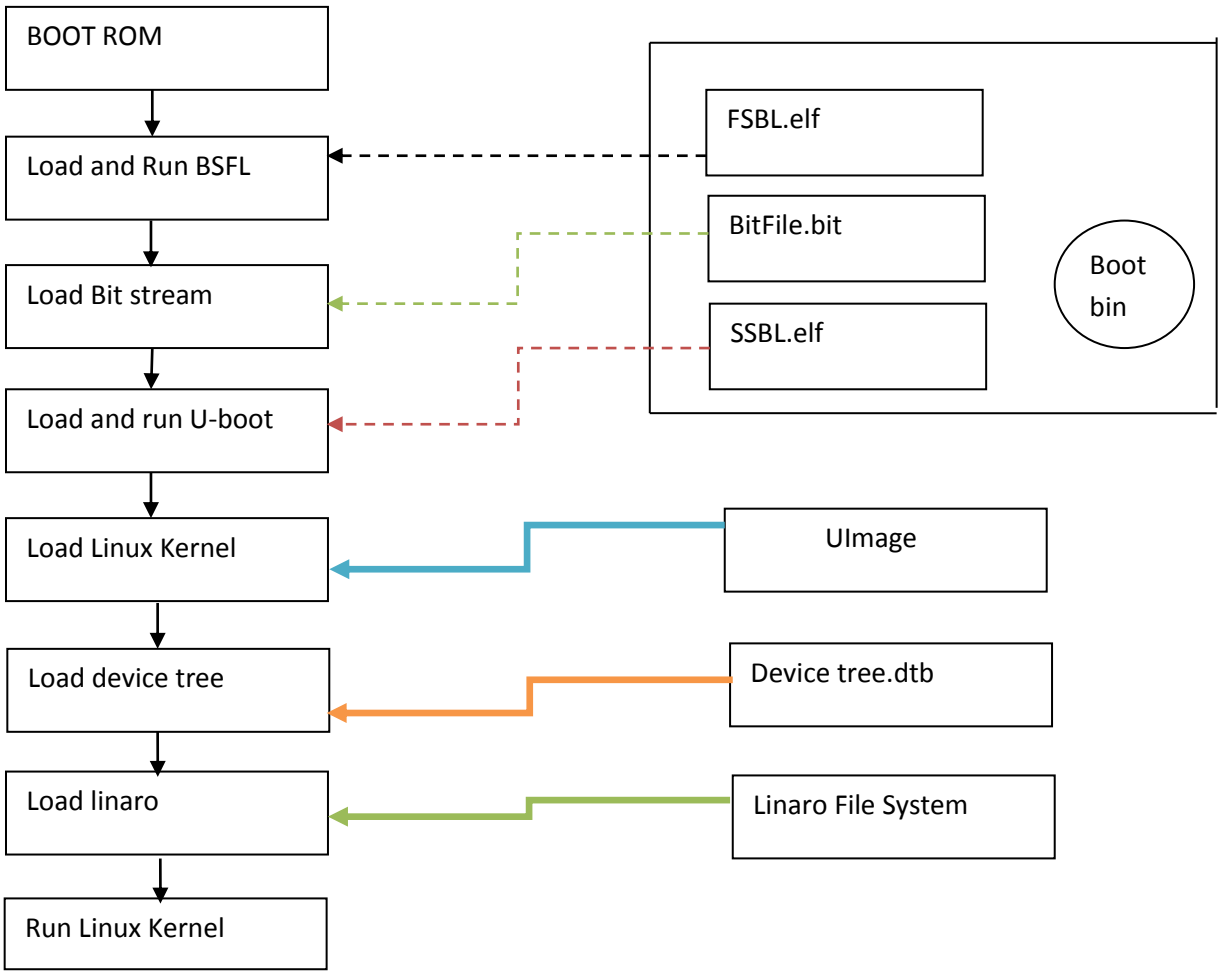


Fig.5.2.1. Zynq Boot Flow.

The stages are explained as below :

Stage-0: After a power-on reset (POR), the hardware samples the boot strap pins to determine the Boot mode: JTAG mode, NAND/NOR flash mode, SD card mode etc. and optionally enables PS PLLs. Then hard-coded BootROM is executed in primary CPU-0. Boot ROM configures the PS to access the boot device. It validates and reads boot header to determine the boot flow and then load the FSBL in OCM [11].

Stage-1: In stage-1 FSBL loaded and executed from OCM. It is responsible for several initialization functions including CPU initialization with PS7 Init configuration data, programming the PL using Bit stream (if available), loading SSBL into DDR memory, handoff control to SSBL execution.

Stage-2: For Linux booting, It is SSBL, such as U-Boot, open course universal boot loader for Zynq. It is responsible for loading the Linux kernel image, device-tree file and Linux file system. It also initializes hardware that is not done by kernel like serial port, DDR memory.

For the software-only implementation, Linaro Ubuntu distribution was used. Linaro is a open-source complete Linux distribution based on Ubuntu. It

supports graphical desktop through on-board HDMI port. For booting from SD card, Linaro file system has to be placed in different partition other than the partition where Kernel image, device tree reside. It is a persistent OS, i.e. all changes are written to memory and it saves files after reboot or shut down. Open CV(version 3.1) was built on top of it. The features that are not necessary like Qt-support, CUDA, video examples were disabled to reduce memory usage.

5.3. Implementation of Sign Board Detection on Linaro(Zed board Implementation)

Performance Estimation in Various modes:

Data set employed :Data set 3

5.3.1.NRS_BOTH mode:

This is the default mode of operation

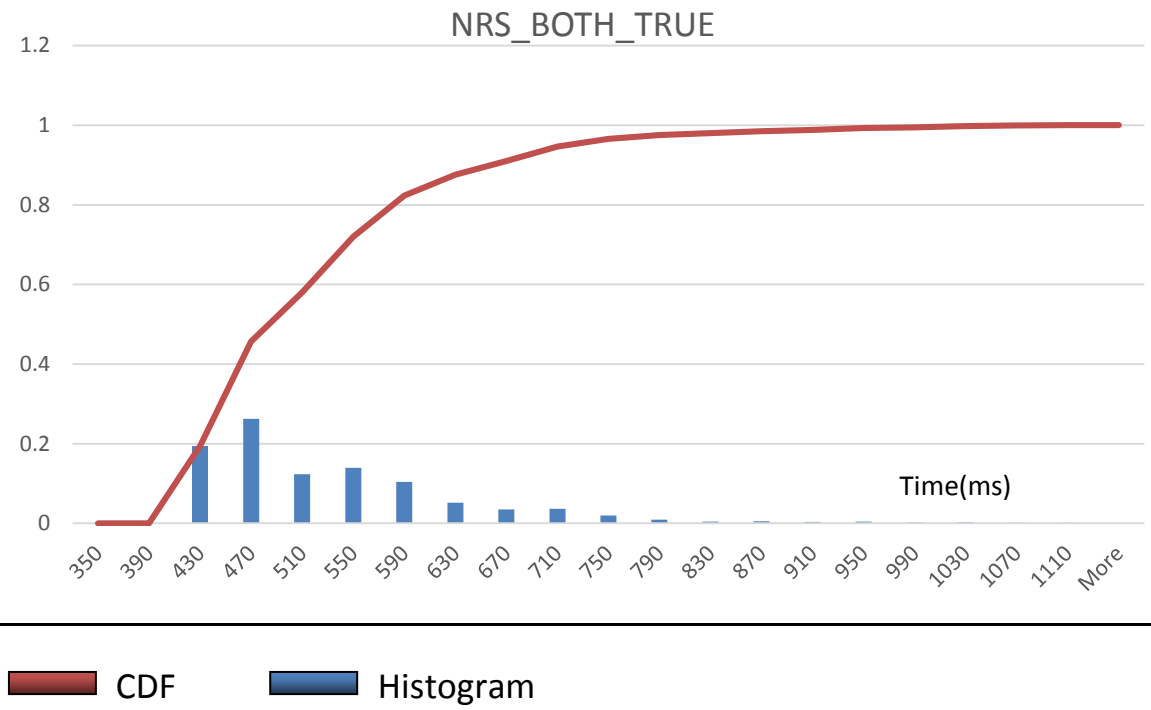


Fig.5.3.1. Performance (Time estimation) for Default mode true cases

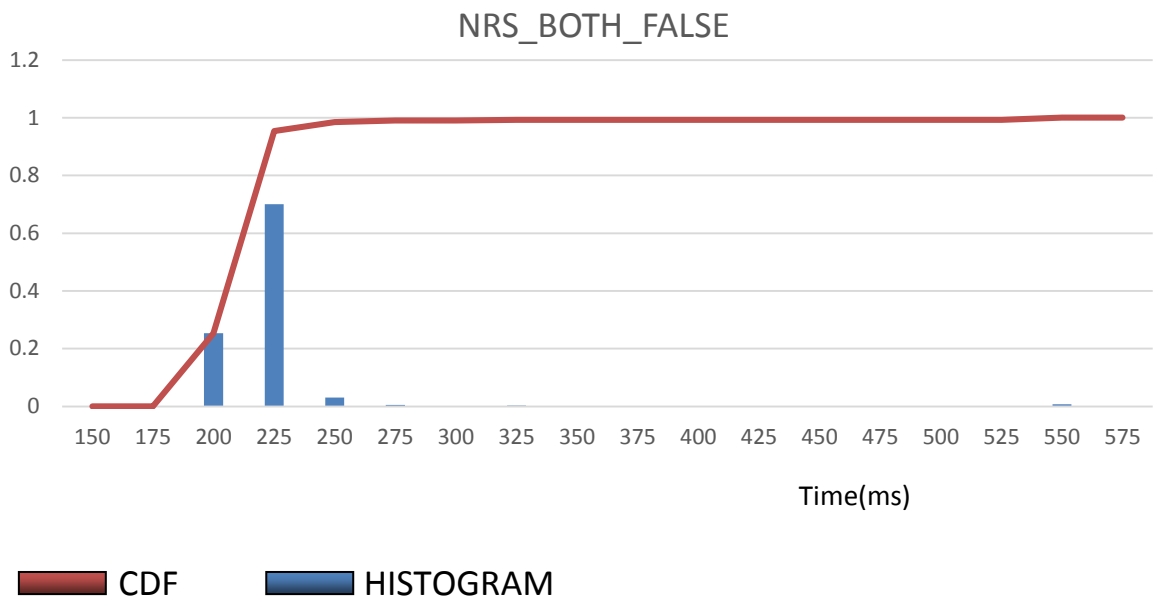


Fig.5.3.2 .Performance (Time estimation) for NRS_BOTH mode false cases.

5.3.2.For NRS_SINGLE mode:

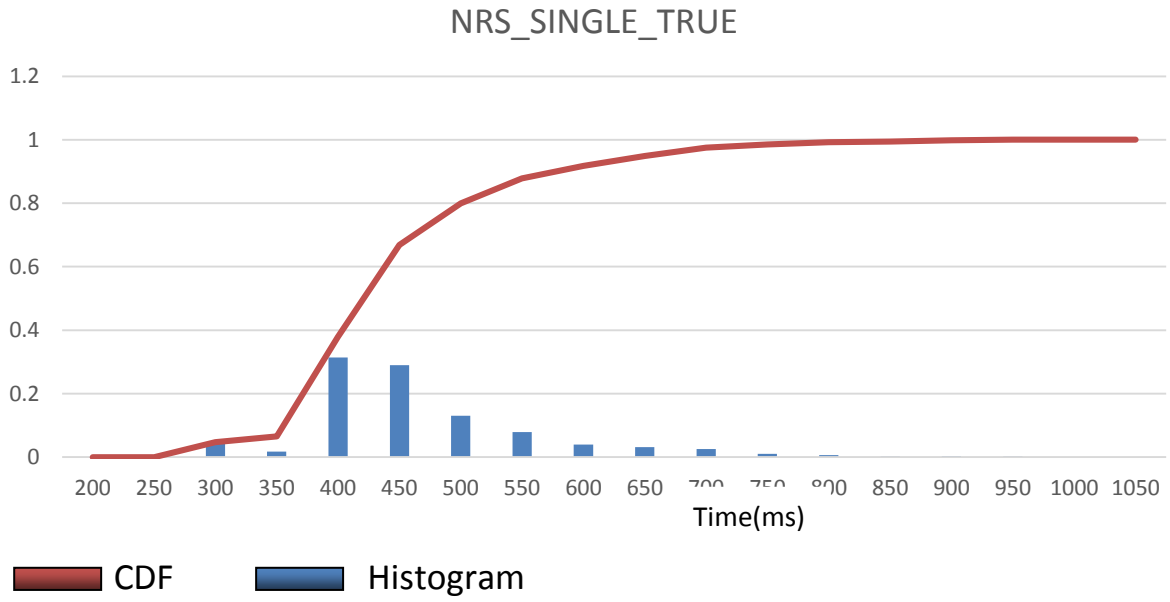


Fig.5.3.3. Performance (Time estimation) for NRS_SINGLE mode true cases.

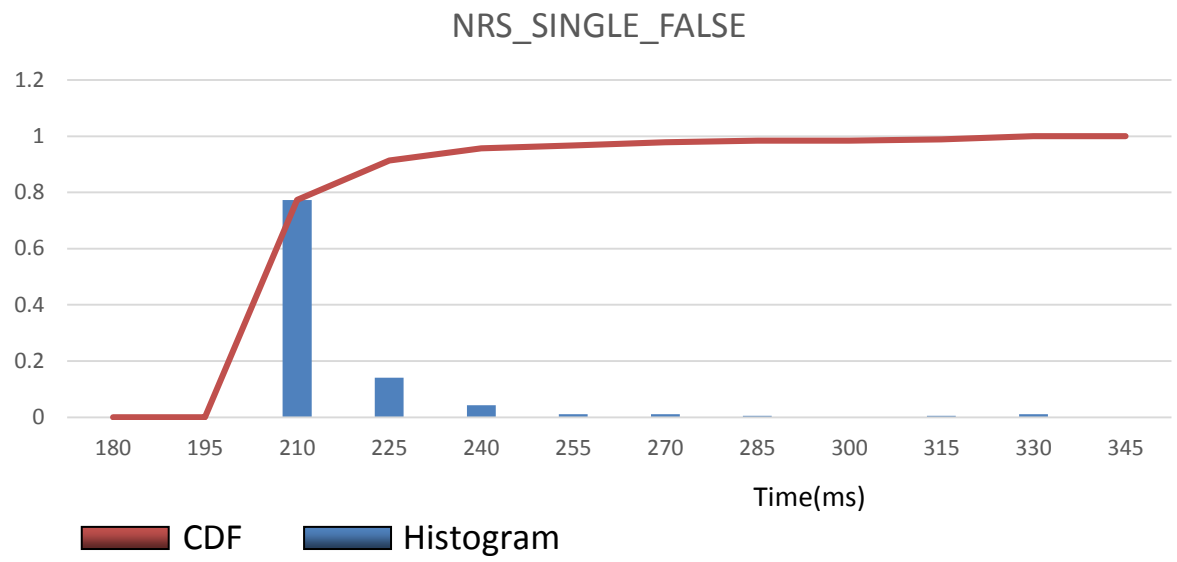


Fig.5.3.4.Performance (Time estimation) for NRS_SINGLE mode false cases.

5.3.3.For RS_BOTH mode:

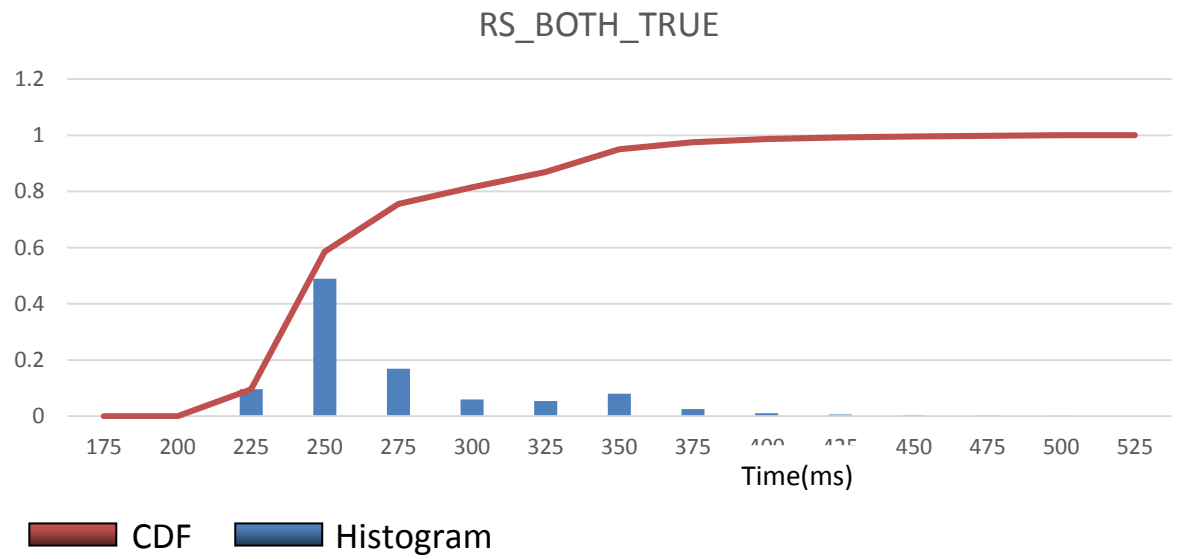


Fig.5.3.5.Performance (Time estimation) for RS_BOTH mode true cases.

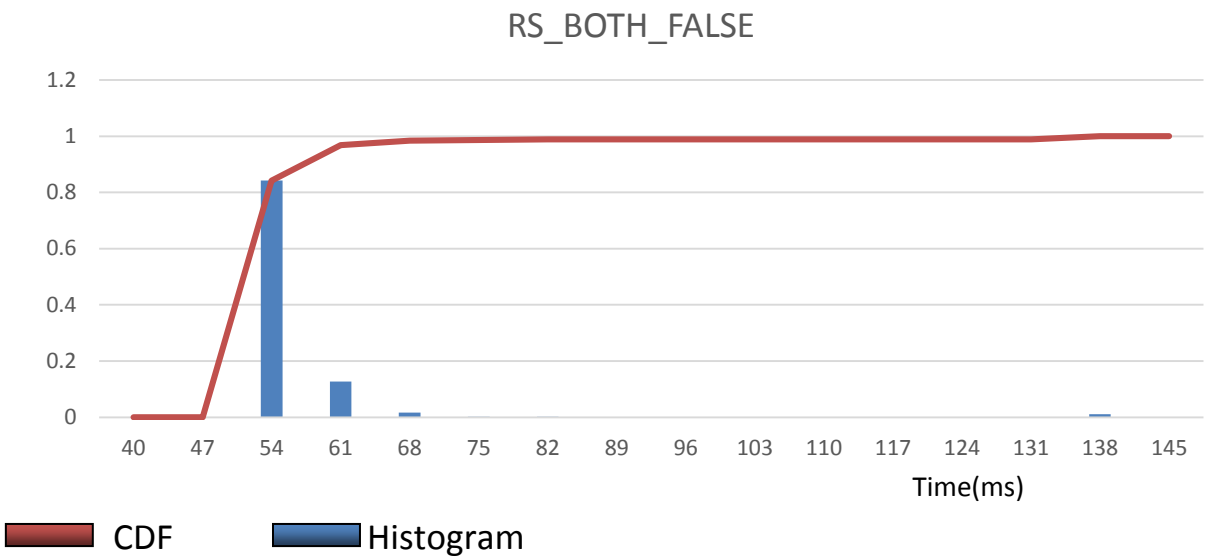


Fig.5.3.6.Performance (Time estimation) for RS_BOTH mode false cases.

5.3.4.For RS_SINGLE mode:

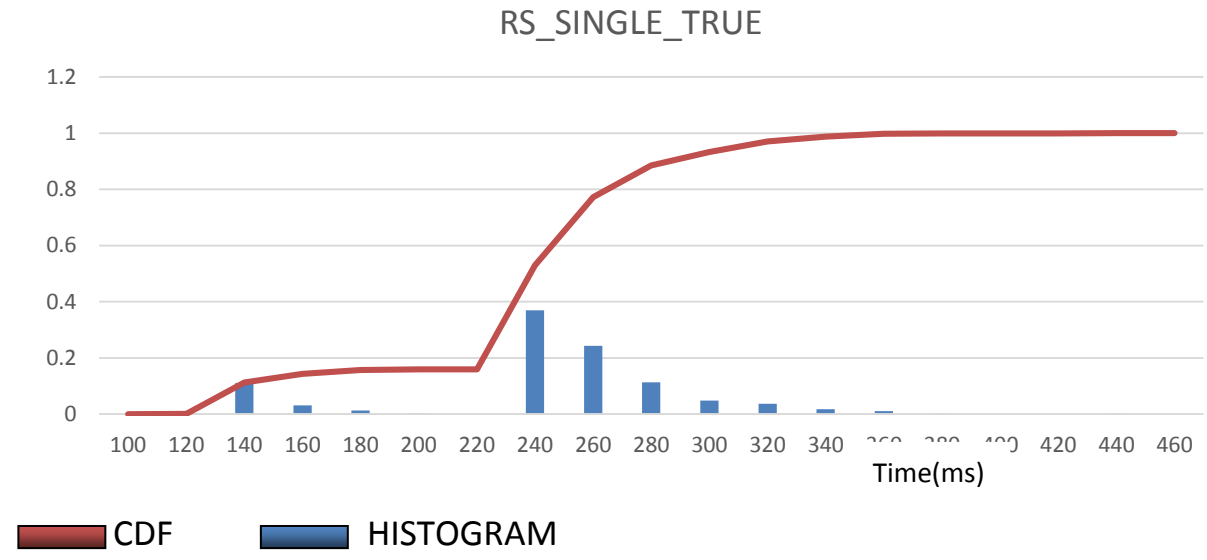


Fig.5.3.7.Performance (Time estimation) for RS_SINGLE mode true cases.

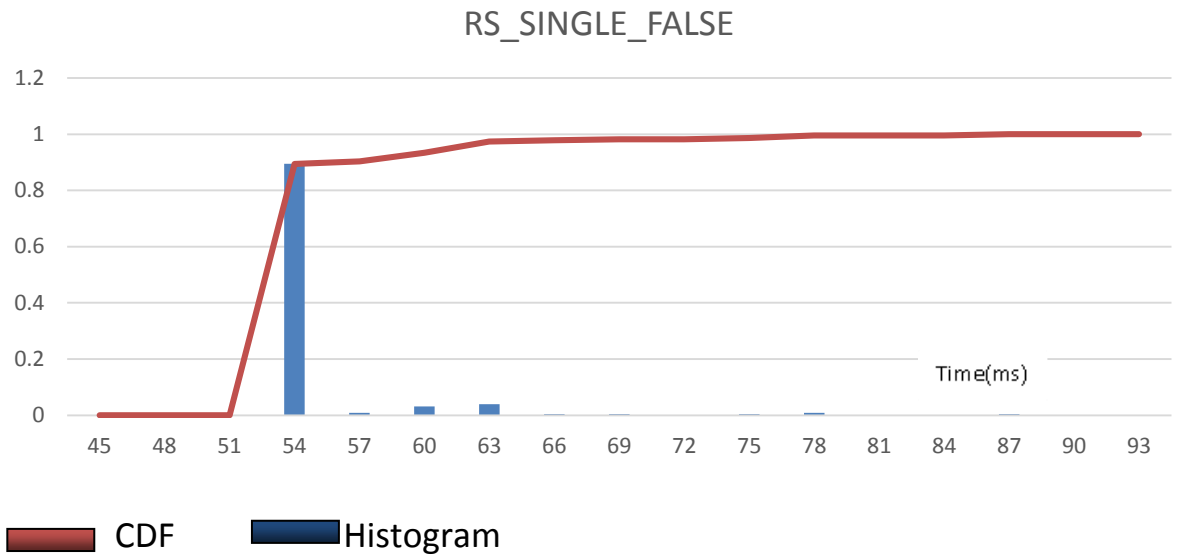


Fig.5.3.8.Performance (Time estimation) for RS_SINGLE mode FALSE cases

5.4.Comparison with Desktop Implementation

Data set employed :Data set 3

The following table summarises the time consumed by the Sign Board detection Algorithm in zed board Implementation.

MODE		Zed board Implementation(ms)				
		Minimum	Maximum	Average	90% point	95% point
NRS_BOTH	TRUE	369.546	1041.82	457.1337	670.98567	710.586942
	FALSE	197.316	540.873	209.8506	225. 97481	227.857621
NRS_SINGLE	TRUE	261.681	935.226	444.5515	575.44368	650.298365
	FALSE	197.519	321.051	206.9383	225.023446	226.124789
RS_BOTH	TRUE	219.006	485.853	261.2473	330.128765	355.867230
	FALSE	52.3473	135.9	54.51582	57.978945	61.2894872
RS_SINGLE	TRUE	118.076	557.683	234.5172	280.856912	310.128731
	FALSE	52.7528	86.7602	53.72602	57.1234851	61.1237891

Table 5.4.1. Performance Estimates on Zed Board Implementation

The following table 5.2 depicts the difference in the time consumed by the implementation of Sign Board detection Algorithm in both platforms.

MODE		Desktop Implementation(ms)			Zed Board Implementation(ms)		
		Average	Minimum	Maximum	Average	Minimum	Maximum
NRS_BOTH	TRUE	34.25442	27.3116	70.36763	457.1337	369.546	1041.82
	FALSE	28.6954	22.0576	57.26287	209.8506	197.316	540.873
NRS_SINGLE	TRUE	32.42210	27.1469	59.91574	444.5515	261.681	935.226
	FALSE	25.33440	21.4493	54.46904	206.9383	197.519	321.051
RS_BOTH	TRUE	17.9652	15.78277	30.7485	261.2473	219.006	485.853
	FALSE	12.1194	10.16834	24.38838	54.51582	52.3473	135.9
RS_SINGLE	TRUE	17.9232	15.58502	26.8265	234.5172	118.076	557.683
	FALSE	10.94612	10.3443	11.99822	53.72602	52.7528	86.7602

Table 5.4.2.Comparision of Performance on both the platforms

CHAPTER 6:

ENERGY MEASUREMENTS FOR SIGN BOARD DETECTION:

6.1.Hardware setup for energy measurement:

The arrangement for energy measurement is shown in figure 6.1. A 10 milliohm, 1 Watt current sense resistor is in series with the 12V input power supply. To measure the voltage across the resistor, one jumper (J21) is connected with it. Agilent 34410A high-performance sampling multimeter was used to measure the voltage. This multimeter was directly interfaced to PC via USB port. From the PC itself it is possible to start/stop the voltage measurement, see the waveform and also to export the results in either CSVfile format or in excel sheet for further calculations.

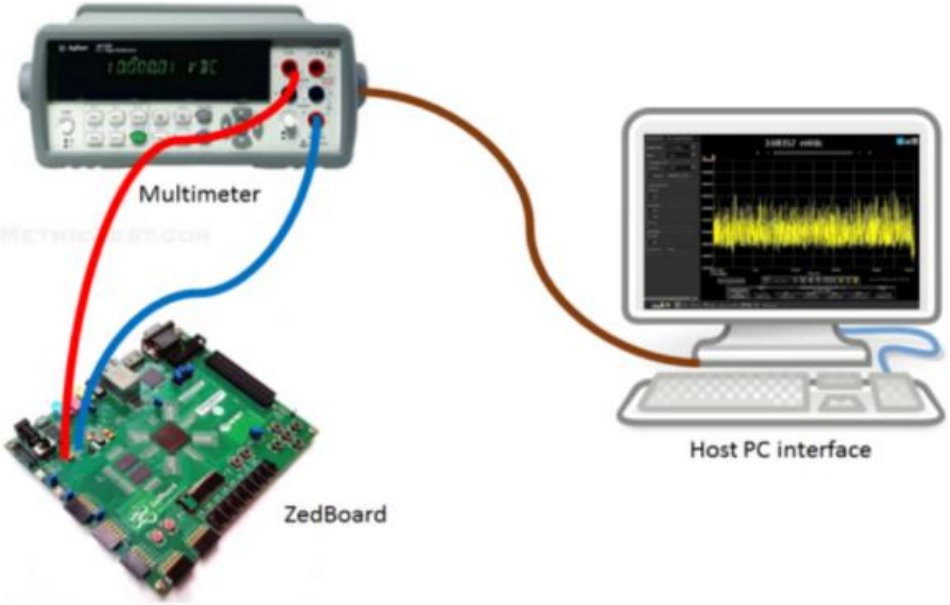


Fig.6.1.Set up For Energy measurement

The energy for various modes of signboard detection is tabulated below. The zed board base voltage is 2.9787mV.The Vdc of zed board is 12v.

6.2.NRS_BOTH mode:

6.2.1.TRUE_NRS_BOTH :

	Voltage (mv)	Current(V/R)	Power (Watts)	Execution Time (ms)	Energy consumed(J)
Average	3.09800	0.309080	3.71760	457.1337	1.69944
Minimum	3.05170	0.305170	3.66204	369.546	1.35329
Maximum	3.22237	0.322237	3.866844	1041.82	4.02855

Table 6.2.1.Energy measurements for NRS_BOTH mode for TRUE cases

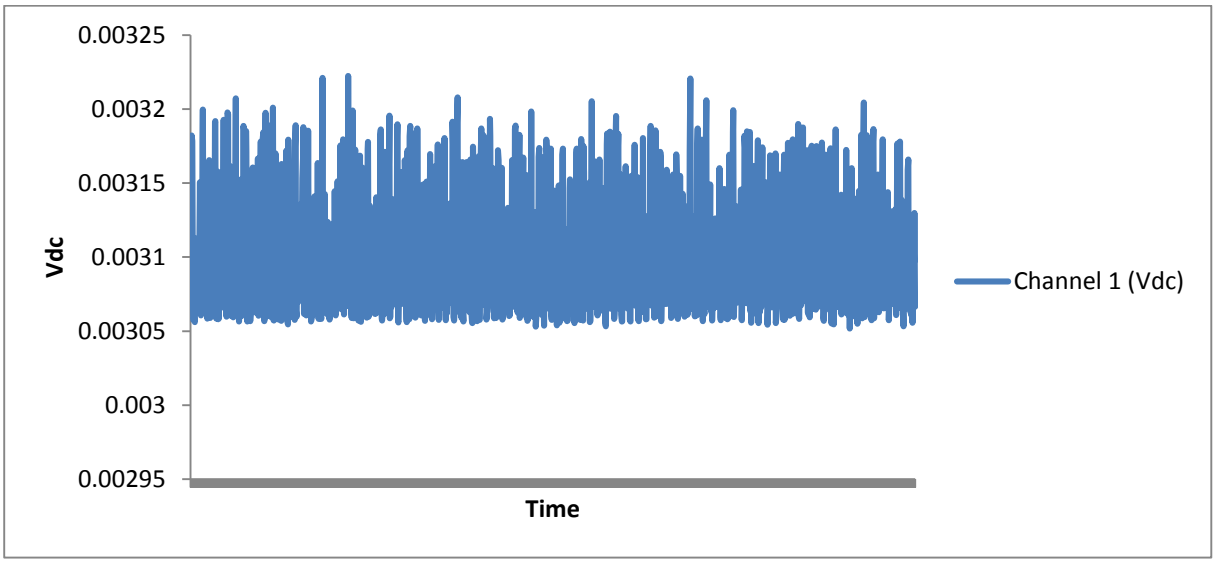


Fig.6.2.1. Voltage graph obtained for TRUE_NRS_BOTH mode

6.2.2.FALSE_NRS_BOTH :

	Voltage (mv)	Current(V/R)	Power (Watts)	Execution Time (ms)	Energy consumed(J)
Average	3.07855	0.307855	3.69426	209.8506	0.7752426
Minimum	3.04681	0.304681	3.656172	197.316	0.7214212
Maximum	3.22080	0.322080	3.86496	540.873	2.09045251

Table 6.2.2.Energy measurements for NRS_BOTH mode for FALSE cases

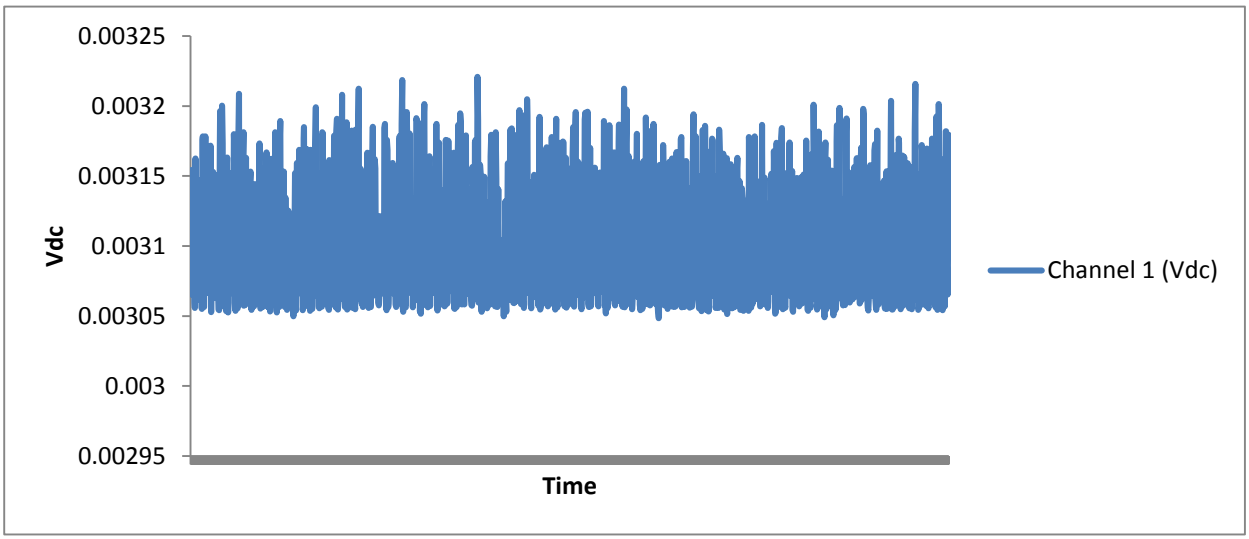


Fig.6.2.2. Voltage graph obtained for FALSE_NRS_BOTH mode

6.3. RS_BOTH mode:

6.3.1.TRUE_RS_BOTH :

	Voltage (mv)	Current(V/R)	Power (Watts)	Execution Time (ms)	Energy consumed(J)
Average	3.0850	0.30850	3.702	261.2473	0.9671375
Minimum	2.9604	0.29604	3.55248	219.066	0.7782275
Maximum	3.2102	0.32102	3.85224	485.853	1.8716223

Table 6.3.1.Energy measurements for RS_BOTH mode for TRUE cases

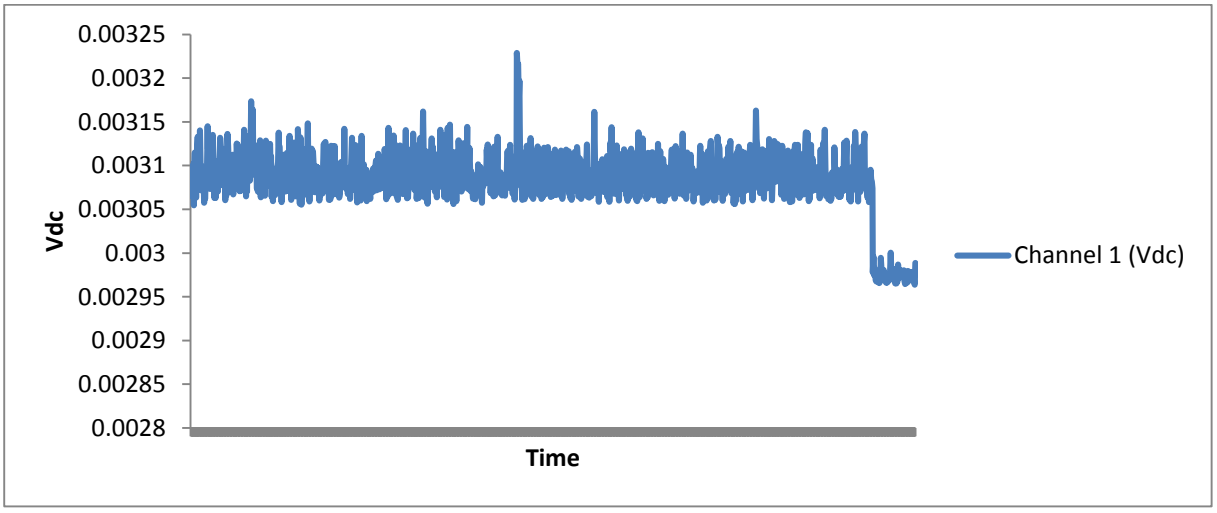


Fig.6.3.1.Voltage graph obtained for TRUE_RS_BOTH mode

6.3.2.FALSE_RS_BOTH:

	Voltage (mv)	Current(V/R)	Power (Watts)	Execution Time (ms)	Energy consumed(J)
Average	3.075602	0.3075602	3.6907224	54.51582	0.201202752
Minimum	3.05481	0.305481	3.665772	54.3473	0.19922481
Maximum	3.20253	0.320253	3.843036	135.900	0.501569119

Table 6.3.2.Energy measurements for RS_BOTH mode for TRUE cases

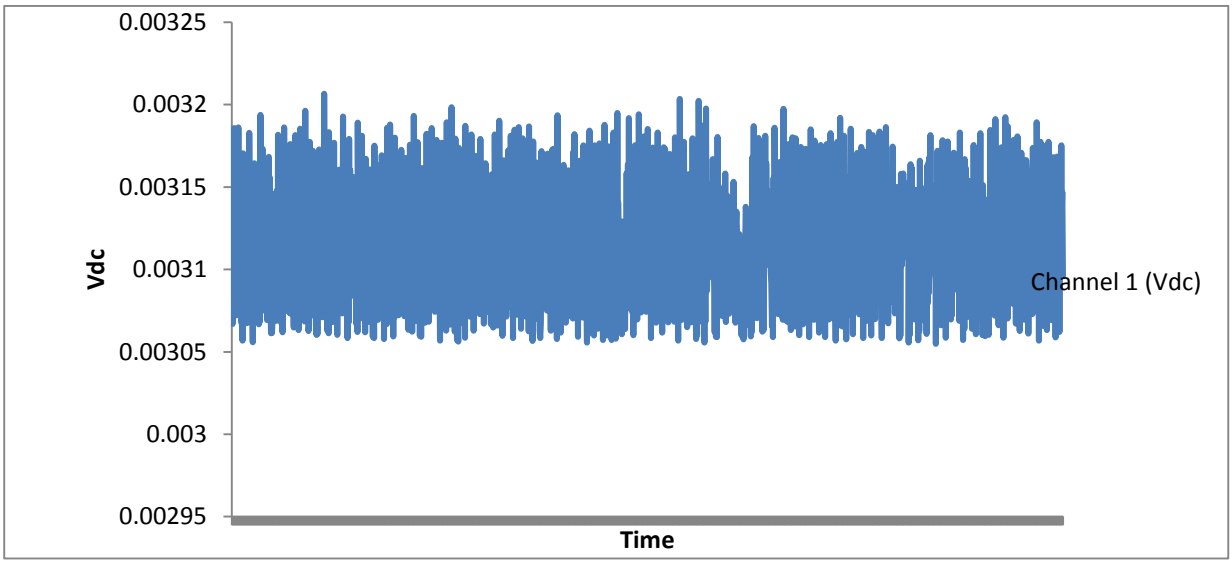


Fig.6.3.2.Voltage graph obtained for FALSE_RS_BOTH mode

6.4. NRS_SINGLE mode:

6.4.1.TRUE_NRS_SINGLE :

	Voltage (mv)	Current(V/R)	Power (Watts)	Execution Time (ms)	Energy consumed(J)
Average	3.0931	0.30931	3.71172	444.551	1.6500488
Minimum	3.0548	0.30548	3.66576	261.681	0.9592597
Maximum	3.2217	0.32217	3.87564	935.226	3.624599

Table 6.4.1.Energy measurements for NRS_SINGLE mode for TRUE cases

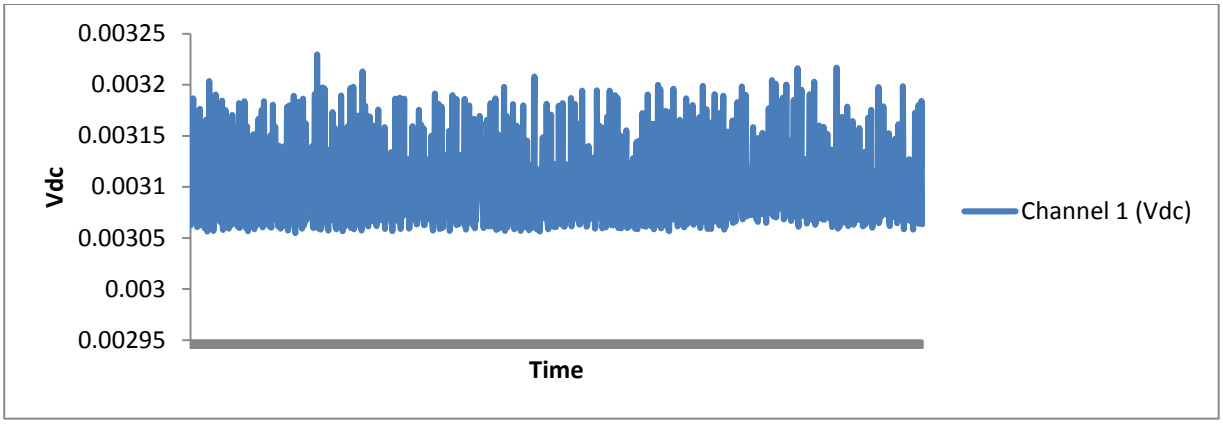


Fig.6.4.1.Voltage graph obtained for TRUE_NRS_SINGLE mode

6.4.2.FALSE_NRS_SINGLE:

	Voltage (mv)	Current(V/R)	Power (Watts)	Execution Time (ms)	Energy consumed(J)
Average	3.080058	0.3080058	3.612696	206.9383	0.7476051
Minimum	3.050071	0.3050071	3.6600852	197.519	0.7229369
Maximum	3.214782	0.3214782	3.8577384	321.051	1.238530

Table 6.4.2.Energy measurements for NRS_SINGLE mode for FALSE cases

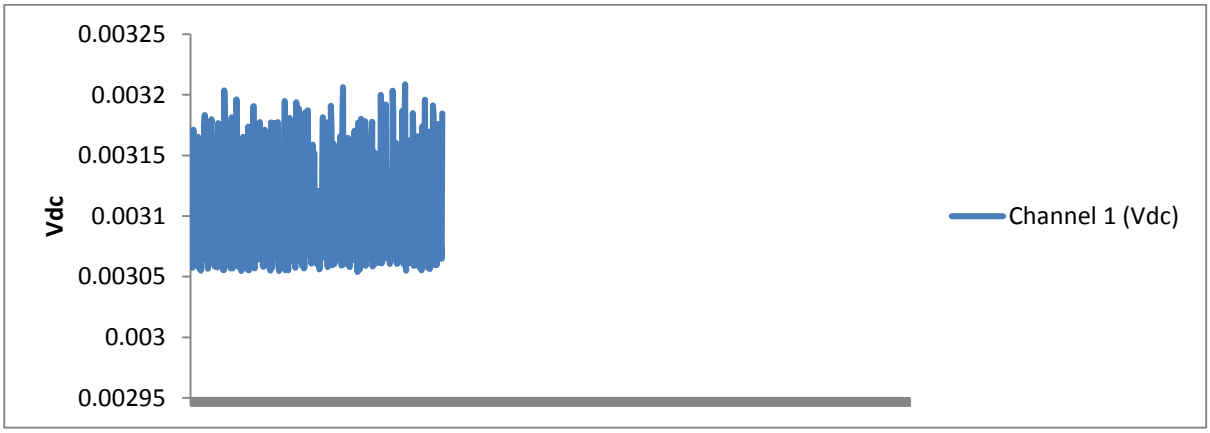


Fig.6.4.2. Voltage graph obtained for FALSE_NRS_SINGLE mode

6.5.RS_SINGLE_mode:

6.5.1.TRUE RS_SINGLE:

	Voltage (mv)	Current(V/R)	Power (Watts)	Execution Time (ms)	Energy consumed(J)
Average	3.007900	0.30079	3.60948	234.5172	0.8464851
Minimum	2.965048	0.2965048	3.5580576	118.076	0.42012120
Maximum	3.154773	0.3154773	3.7857276	557.683	2.11123592

Table 6.5.1.Energy measurements for RS_SINGLE mode for TRUE cases

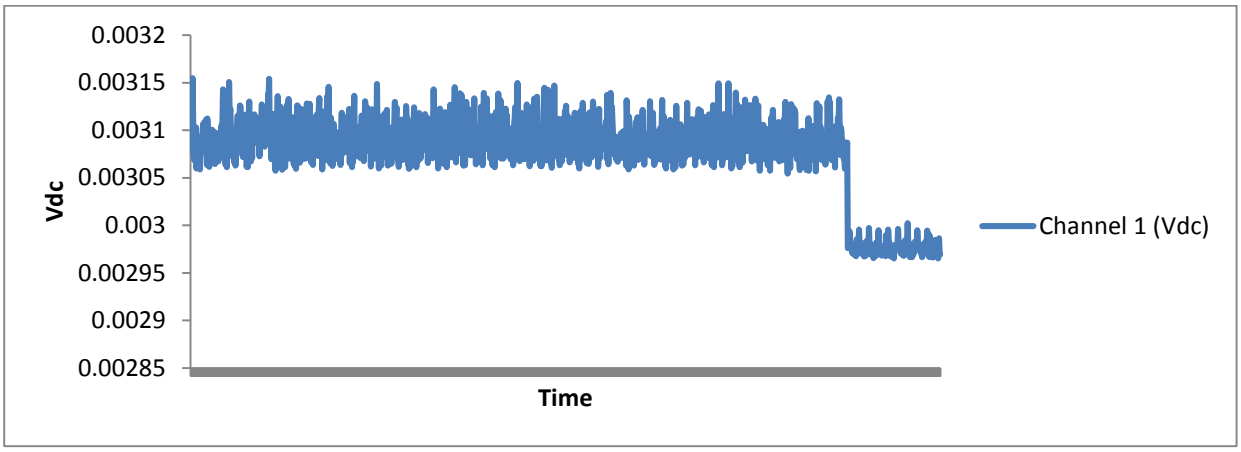


Fig.6.5.1.Voltage graph obtained for TRUE_RS_SINGLE mode

6.5.2.FALSE_RS_SINGLE:

	Voltage (mv)	Current(V/R)	Power (Watts)	Execution Time (ms)	Energy consumed(J)
Average	0.00309014	0.309014	3.708168	53.72602	0.199225108
Minimum	0.00294174	0.294174	3.530088	52.7528	0.188222262
Maximum	0.00320078	0.320078	3.840936	86.7602	0.333240375

Table 6.5.2.Energy measurements for NRS_SINGLE mode for FALSE cases

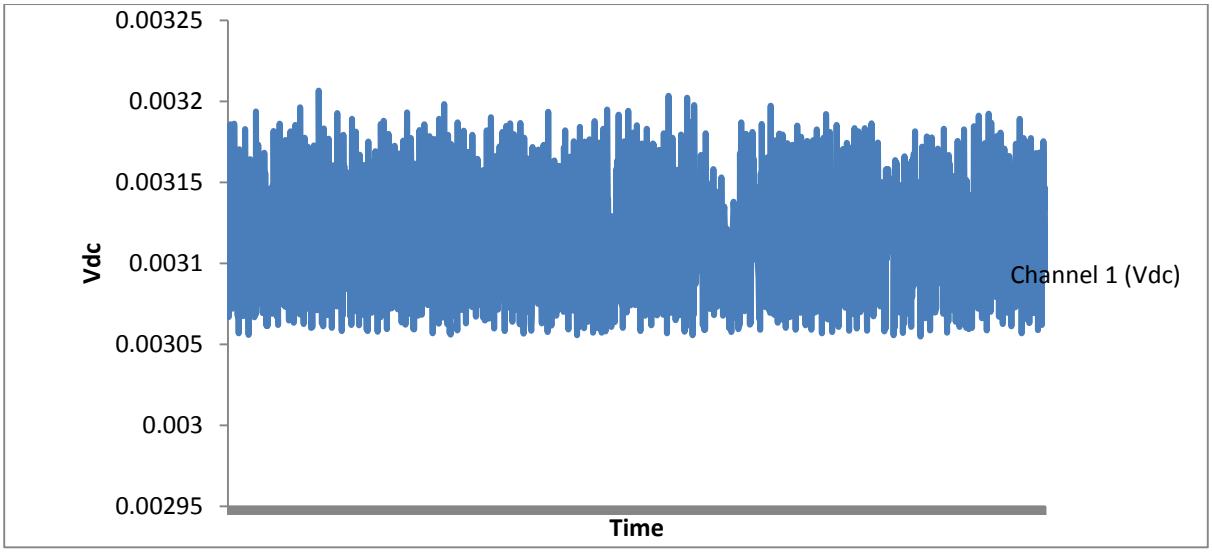


Fig.6.5.2.Voltage graph obtained for FALSE_RS_SINGLE mode

Energy Consumed	NRS_BOTH		NRS_SINGLE		RS_BOTH		RS_SINGLE	
	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
AVERAGE	1.69944	0.775243	1.650048	0.747605	0.967137	0.201203	0.846485	0.1992251
MINIMUM	1.35329	0.721421	0.959259	0.722936	0.778227	0.199225	0.420121	0.18822226
MAXIMUM	4.02855	2.090452	3.624599	1.23853	1.871622	0.501569	2.111236	0.33324037

Table 6.6.Comparison of Energy measurements for all modes

***Note: All measurements are performed on Data Set3*

CHAPTER 7

HARDWARE SOFTWARE CODESIGN:

7.1.Introduction:

Hardware Software co design tries to exploit the synergy of hardware and software with the goal to optimize and/or satisfy design constraints such as cost, performance, and power of the final product. At the same time, it targets to reduce the time-to market frame considerably.[9].The major purposes of software and hardware co-design are to implement complex systems with correctness using coordination and concurrency.[9].Thus the Hardware software co design is employed to achieve the objectives of system level design by exploiting the flexibility in design changes allowed, features, reusability offered by software and performance ,low power advantages offered by hardware.[9]. The codesign methodology increases the predictability of the embedded system design by using Analysis methodologies and synthesis methodologies.

The levels of abstraction of hardware and software are shown in figure 7.1.

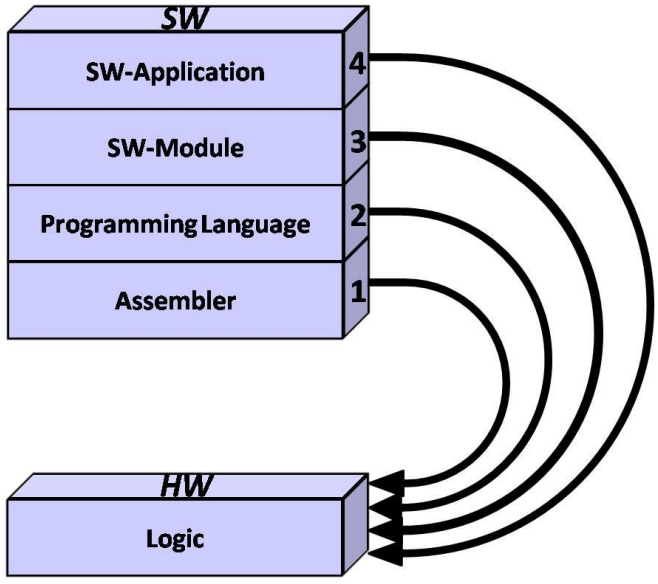


Fig.7.1.Levels of Abstraction of HW/SW Codesign[14]

The harware software co design flow is depicted below in Fig.7.2.

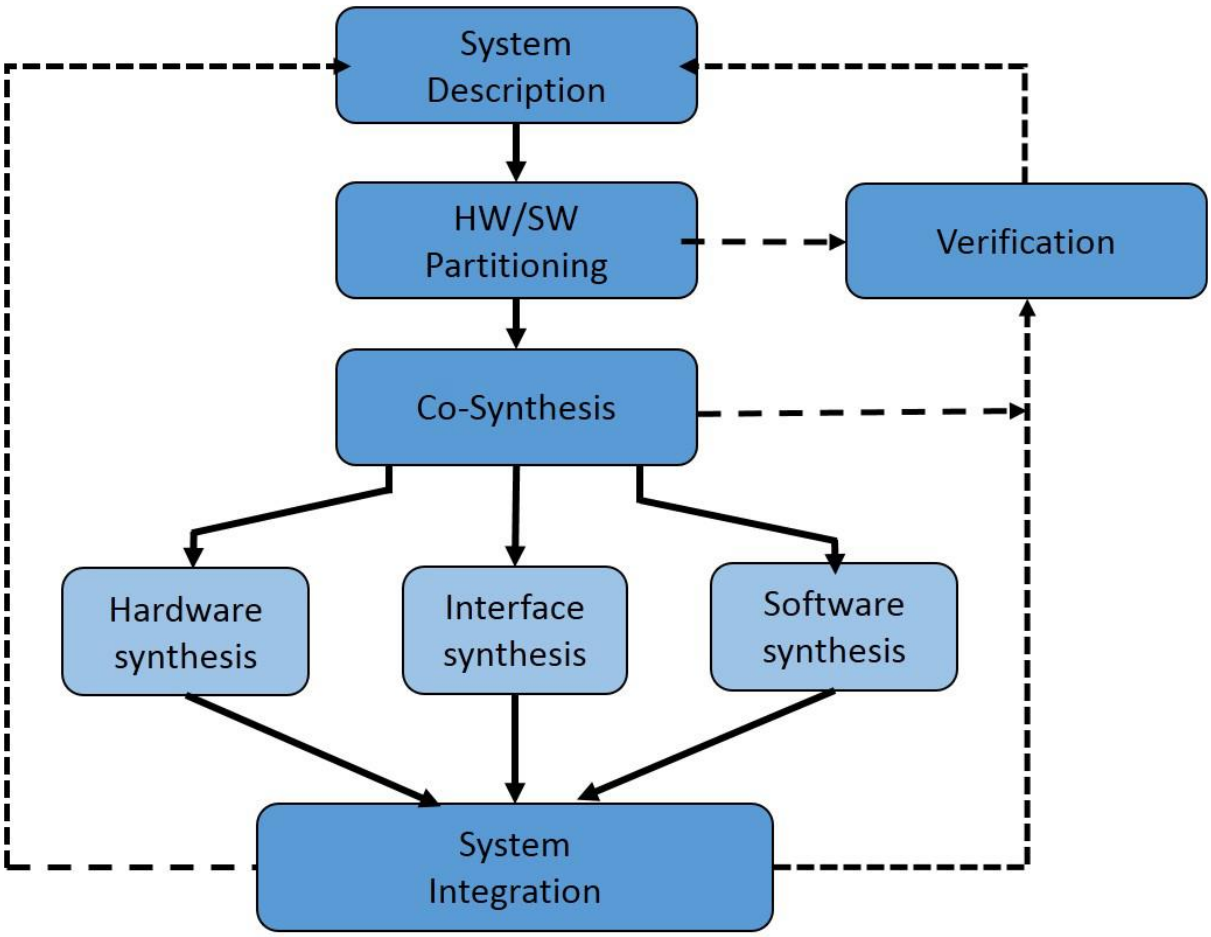


Fig.7.2 Hardware Software CoDesign flow.

7.2.Profiling :

Profiling is a key technology to ensure optimal match between target hardware and software by 0 the software efficiency. As the main focus is on accelerator synthesis, so the profiling granularity, used here is coarse. Otherwise, for fine granularity the bottleneck would be the communication between processor and hardware. From profiling, the information regarding the occurrence of each function and the time lapsed in executing each function is obtained : occurrence of function and time spent in each function. After profiling we will get the candidates for hardware acceleration. The profiler used here is Gprof.

7.3.Results:

The results of profiling in various modes are as follows:

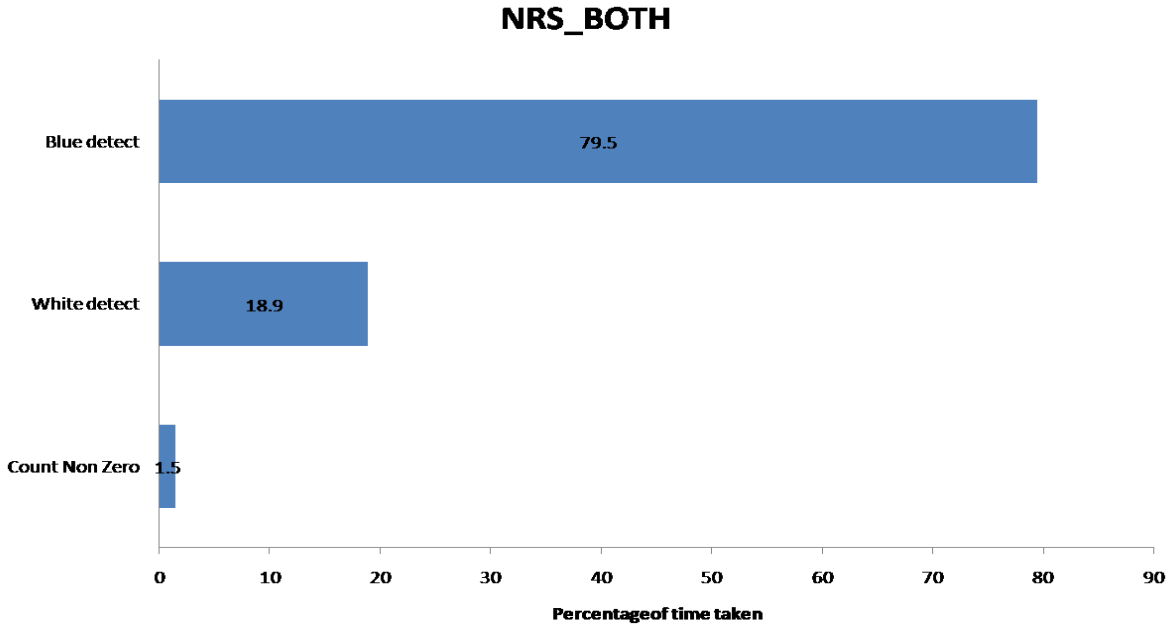


Fig.7.3.1.Profiling results for NRS_BOTH mode

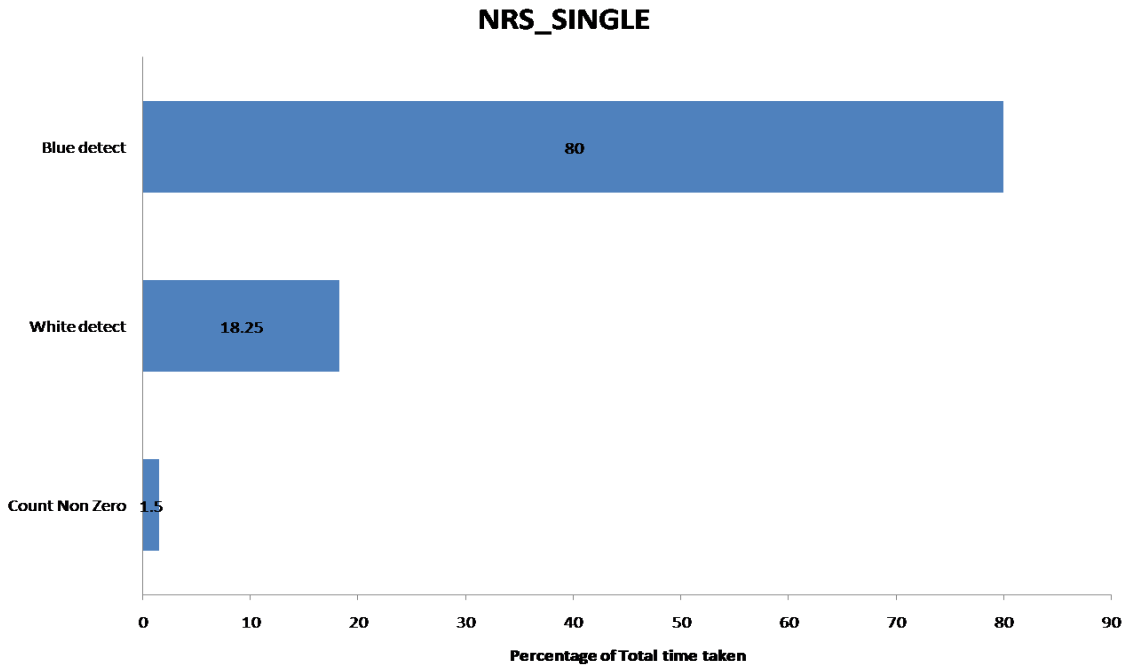


Fig.7.3.2.Profiling results for NRS_SINGLE mode

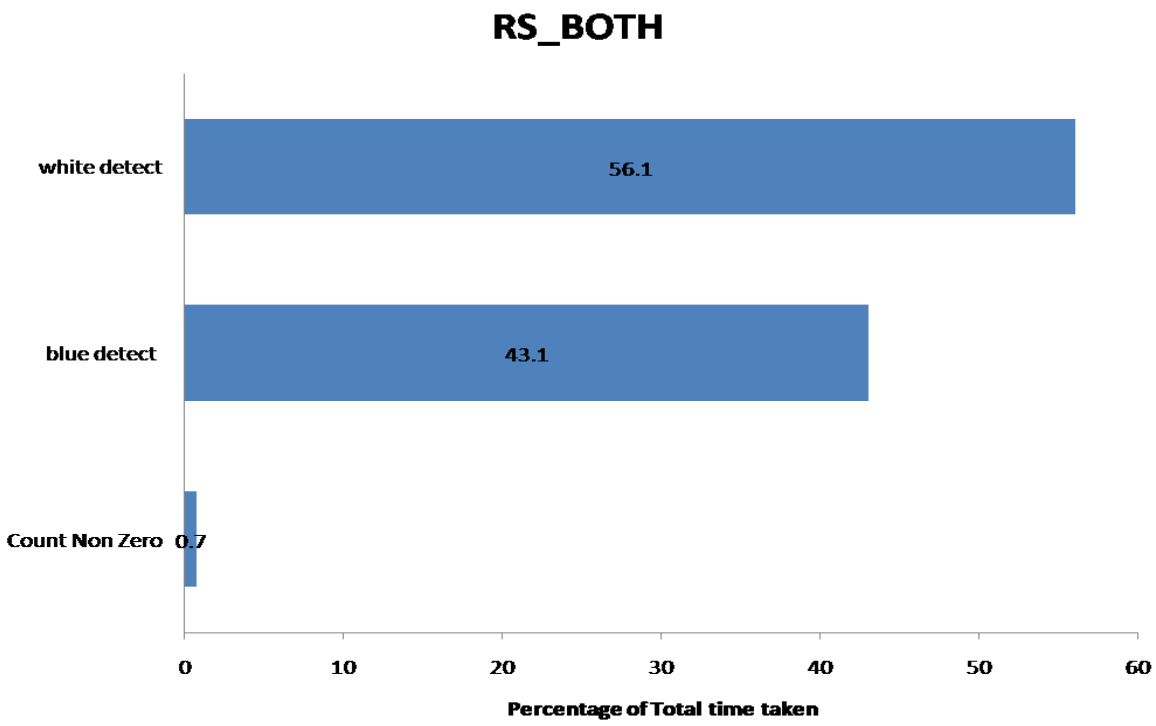


Fig.7.3.3.Profiling results for RS_BOTH mode

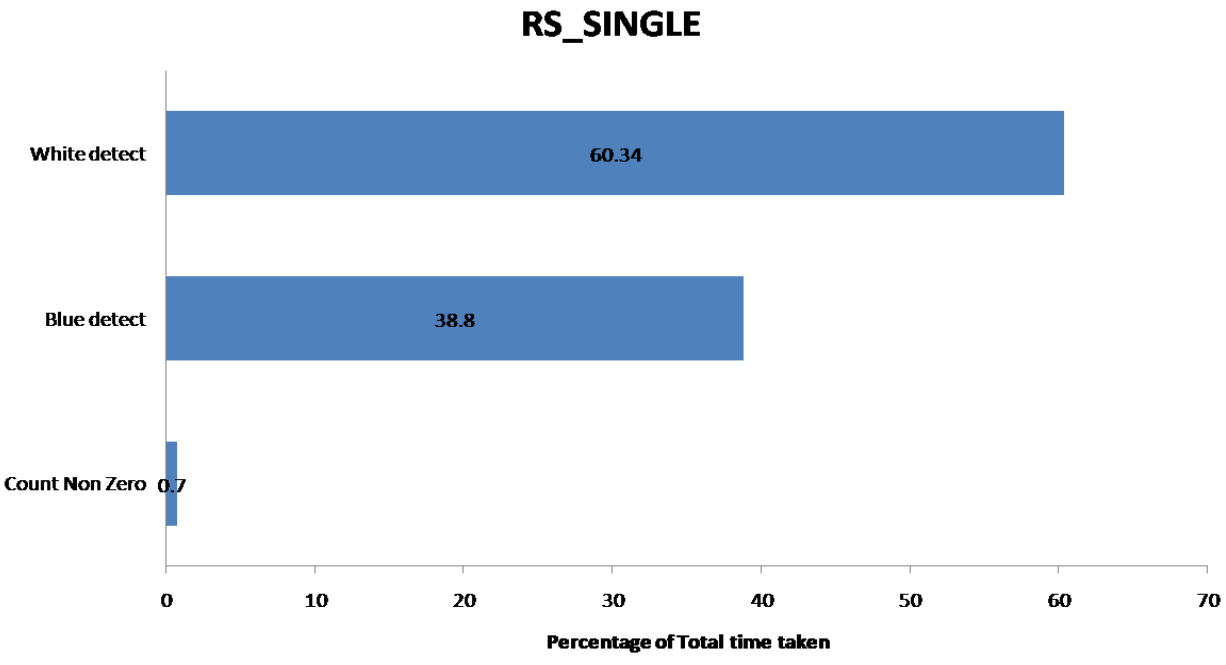


Fig.7.3.4.Profiling results for RS_SINGLE mode

CHAPTER 8

CONCLUSION AND FUTURE WORK:

Currently the algorithm works only for the sign boards with the text in white colour. This needs to be extended to all possible signboards by making the algorithm less sensitive to colour. The information extraction part is restricted in its accuracy to the digital sign boards as the OCR engines performance is very poor for hand written text. The noise removal algorithms include certain assumptions that in some cases lead to loss of information. The current state implementation does not have any provision for shadow removal and it needs to be taken care of in future. Also the hardware acceleration has to be performed.

.

REFERENCES:

[1]. http://assistech.iitd.ernet.in/smartcane.php

[2]. Seyedeh Fatemeh Razavi , Hedieh Sajedi ; Mohammad Ebrahim Shiri" Integration of colour and uniform interlaced derivative patterns for object tracking" IET Image Processing ,Volume:10 , Issue: 5 , pg no.381 - 390.

[3].*Object detection for low power embedded applications,* M.Tech Thesis by Deepak Sahoo. IIT Delhi.

[4]. https://en.wikipedia.org/wiki/Optical_character_recognition#cite_note-1.

[5].https://web.archive.org/web/20061026075310/http://google-code-updates.blogspot.com/2006 /08/announcing-tesseract-ocr.html

[6]. https://en.wikipedia.org/wiki/Tesseract_(software)

[7]. https://**tesseract**-**ocr**.googlecode.com/files/**Tesseract**OSCON.pdf

[8]. C. Haubelt, J. Teich, K. Richter, and R. Ernst, " System design for flexibility" , in Proc. Conf. Des. Autom. Test Eur., Washington, DC, 2002, pp. 854–861.

[9]. C. U. Smith, G. A. Frank, and J. L. Cuadrado, "An architecture design and assessment system for software/hardware co design," in Proc. 22nd ACM/IEEE Des. Autom. Conf., 1985, pp. 417–424.

[10].http://zedboard.org/sites/default/files/documentations/ZedBoard_HW_UG_v2_2.pdf

[11]. http://zedboard.org/sites/default/files/documentations/GS-AES-Z7EV-7Z020-G-14.1-V6%5B1%5D.pdf [12]. userguides/zedboard/

[13].http://literature.cdn.keysight.com/litweb/pdf/59680162EN.pdf?id=1000070110:epsg:dow

[14].HardwareSoftware codesign.///https: