International Journal of Inventive Engineering and Sciences (IJIES)
ISSN: 2319–9598, Volume-1 Issue-8, July 2013

OCR Based Mapless Navigation Method of Robot

Sayyan N. Shaikh, Neelakantha V. Londhe

***Abstract*— This paper reviews a method for a robot can locate and track landmarks using segmentation algorithm within proper distance, It extracts text or sign from the landmark and inputs them into the OCR engine for recognition. Simultaneously, a projection analysis of texts or signs of the landmark is conducted, finally by identifying the semanteme of text or signs the robot can find the routes to the destinations automatically.**

Experiments on the robot model shows that this method can be used to accomplish maples navigation task perfectly in the indoors or customised environment. Being able to navigate with landmarks in real life directly without generating maps or resetting new navigation sign for robots. Specially, this method can be applied in the field of service robots rapidly, which can enhance their adaptability and viability significantly.

***Index Terms*— Mapless navigation, Landmark recognition, OCR, Kohonen Neural Network..**

I. INTRODUCTION

There are enormous landmarks in the environment we live (Fig. 1), which are set to help people's travel, and they often contain various text and direction. With these signs, people can locate their positions and target their destinations easily. Even in an utterly strange place, people can find their way with landmark system. For example, in a public place like an airport or a station, people can reach their destinations successfully by just following the right landmark without being familiar with that place.

Fig. 1 Landmarks in Daily Life

Compared with robots, humans firstly labels the environment with landmarks, then directly draw the labels during the navigation process. As a result, humans are less dependent on maps. In other words, humans' navigation system is partially transported into the environment instead of being completely endogenous. A number of potential markets are slowly emerging for mobile robotic systems. Entertainment applications and household or office assistants are the primary targets in this area of development. These types of robots are designed to move around within an often highly unstructured and unpredictable environment.

Existing and future applications for these types of autonomous systems have one key problem in common: navigation.

Vision is one of the most powerful and popular sensing method used for autonomous navigation. When compared with other on-board sensing techniques, vision based approaches to navigation continue to demand a lot of attention from the mobile robot research community, due to its ability to provide detailed information about the environment, which may not be available using combinations of other types of sensors. The past decade has seen the rapid development of vision based sensing for indoor navigation tasks. For example, 20 years ago it would have been impossible for an indoor mobile robot to find its way in a cluttered hallway, and even now it still remains a challenge. Vision-based indoor navigation for mobile robots is still an open research area. Autonomous robots operating in an unknown and uncertain environment have to cope with dynamic changes in the environment, and for a robot to navigate successfully to its goals while avoiding both static and dynamic obstacles is a major challenge. Most current techniques are based on complex mathematical equations and models of the working environment, however following a predetermined path may not require a complicated solution, and the following proposed methodology should be more robust.

Basically vision based navigation falls into three main groups depends on localization methods.

•MAP BASED NAVIGATION: This consists of providing the robot with a model of the environment. These models may contain different degrees of detail, varying from a complete CAD model of the environment to a simple graph of interconnections or interrelationships between the elements in the environment.

• MAP BUILDING BASED NAVIGATION: In this approach first a 2D or 3D model of the environment is first constructed by the robot using its on-board sensors after which the model is used for navigation in the environment.

• MAPLESS NAVIGATION: This category contains all systems in which navigation is achieved without any prior description of the environment. The required robot motions are determined by observing and extracting relevant information about the elements in the environment, such as walls, desks, doorways, etc. It is not necessary that the absolute position of these objects is known as further navigation to be carried out.

Traditional robot navigation techniques mostly are map based method, and mapless method based on the VSRR (View Sequenced Route Representation) model. The map based method need to build the map first, which can be done either by human or the robot itself. While the VSRR-based approach need the robot roam around the scene, extract and save the feature at each position, and localize it by matching during the navigation. Both of the two approaches are vulnerable to wider scenes therefore can't be applied in a strange environment.

Manuscript received on July, 2013.

Mr.Sayyan N. Shaikh, A budding professional pursuing M. Tech in Industrial Automation and Robotics from Srinivas Institute of Technology, Mangalore.India.

Dr. Neelakantha V. Londhe, He is an Professor, Department of Mechanical Engineering at Srinivas Institute of Technology, Valachil, Mangalore, India.

GPRS is another widely used navigation tool, but it fails to function in indoor environments. Despite we can label the environment for robots (ex. RFID) just like we set landmarks for ourselves, however, it's not efficient either in terms of time or money. So, to solve this problem, we propose using a landmark system for human directly to navigate robots.

In this proposed method, the robot is allowed to discover a path automatically by detecting and reading textual information or signs located on the landmark by using OCR.

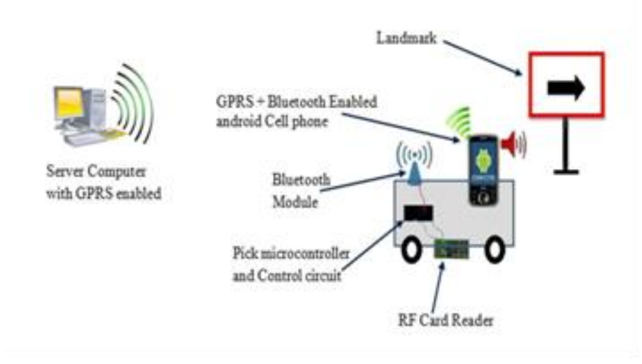
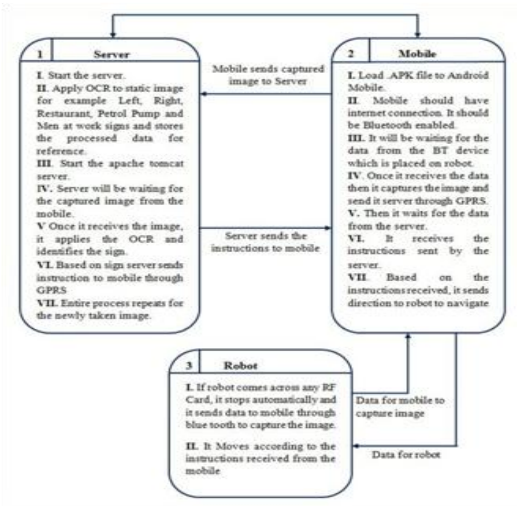

Landmark
GPRS + Bluetooth Enabled android Cell phone
Server Computer with GPRS enabled
Bluetooth Module
Pick microcontroller and Control circuit
RF Card Reader

Fig. 2 Proposed System

The navigation hardware is connected with Android phone through the Bluetooth module for transfer of data for its ordered movement. On the other hand the Android phone is connected to the server through the internet (GPRS) .Whenever navigator hardware comes across RF card which is placed near a landmark. Then the navigator will stop and send the signal to the phone. The phone, on receiving the signal invokes its camera to capture the snap of the landmark and sends the image to the server through the internet. The server processes the received image using OCR mechanism. Later its meaning is reverted back to the phone which speaks up the meaning through its 'Text To Speech' synthesizer, and transfers the signal to the navigator through Bluetooth for the possible move.

II. METHODOLOGY

The design methodology expands the details of an analysis model by taking into account all technical implementations and restrictions. The purpose of the design is to specify a working solution that can be easily translated into programming code and implementation models.


1 Server
Mobile sends captured image to Server
2 Mobile
Server sends the instructions to mobile
3 Robot
Data for mobile to capture image
Data for robot

I. Start the server.
II. Apply OCR to static image for example Left, Right, Restaurant, Petrol Pump and Men at work signs and stores the processed data for reference.
III. Start the apache tomcat server.
IV. Server will be waiting for the captured image from the mobile.
V Once it receives the image, it applies the OCR and identifies the sign.
VI. Based on sign server sends instruction to mobile through GPRS
VII. Entire process repeats for the newly taken image.

I. Load .APK file to Android Mobile.
II. Mobile should have internet connection. It should be Bluetooth enabled.
III. It will be waiting for the data from the BT device which is placed on robot.
IV. Once it receives the data then it captures the image and send it server through GPRS.
V. Then it waits for the data from the server.
VI. It receives the instructions sent by the server.
VII. Based on the instructions received, it sends direction to robot to navigate

I. If robot comes across any RF Card, it stops automatically and it sends data to mobile through blue tooth to capture the image.
II. It Moves according to the instructions received from the mobile

Fig. 3 DFD for the Proposed System

III. IMAGE EXTRACTION

On the server side we have to extracting images from landmark using image extraction algorithm.It includes following steps.

A.Landmark Image Extraction

1. Binarization: Landmark extraction is the first stage of this algorithm. Image captured from the camera is first converted to the binary image consisting of only 1's and 0's (only black and white) by thresholding the pixel values of 0 (black) for all pixels in the input image with luminance less than threshold value and 1 (white) for all other pixels.
2. Smearing: To find the Landmark region, the smearing algorithm is used. Smearing is a method for the extraction of text or sign areas on a mixed image. With the smearing algorithm, the image is processed along vertical and horizontal runs (scan-lines). If the number of white pixels is less than a desired threshold or greater than any other desired threshold, white pixels are converted to black. In this system, threshold values are selected as 10 and 100 for both horizontal and vertical smearing.

If the number of 'white' pixels < 10; pixels become 'black' and removing the noises and unwanted spots.
Else; no change
If the number of 'white' pixels > 100; pixels become 'black'
Else; no change

B. Landmark Location Finder:

After smearing, a morphological operation, dilation, is applied to the image for specifying the landmark location. However, there may be more than one candidate region for landmark location. To find the exact region of landmark and eliminate the other regions, some criteria tests are applied to the image by smearing and filtering operation.

C. Segmentation:

In the segmentation of Landmark image, text or sign landmark region is segmented into its constituent parts obtaining the images individually. It includes the following steps

1. Filtering: Firstly, the image is filtered for enhancing the image and removing the noises and unwanted spots. Images are often corrupted by random variations in intensity, illumination, or have poor contrast and can't be used directly. In Filtering we transform pixel intensity values to reveal certain image characteristics. Filtering is a necessary process because the presence of noise or unwanted spot will produce wrong results.
2. Dilation: Dilation operation is applied to the image for separating the images from each other if the images are close to each other. After this operation, horizontal and vertical smearing are applied for finding the image regions.
3. Individual Image Separation: The next step is to cut the landmark image. It is done by finding starting and end points of each individual image in the horizontal direction.
4. Normalization: Normalization is to refine the images into a block containing no extra white spaces (pixels) in all the four sides of the images. Then each image fits to equal size.

5. Fitting approach is necessary for template matching. For matching the images to the database, input images must be equal sized with the database images.
6. Template Matching: Template matching is an effective algorithm for recognition of images. The image is compared with the ones in the database and the best similarity is measured. A basic method of template matching uses a convolution mask (template), tailored to a specific feature of the search image, which we want to detect. This technique can be easily performed on grey images or edge images.

After extracting the image we need to recognize image using the Kohonen Neural Network.

IV. IMAGE RECOGNITION PROCEDURE WITH KOHONEN

As like all other recognition procedure character recognition is nothing but a recognition process. Here a character recognition procedure using Kohonen is described below.

A. Landmark image which is taken as input.
B. Extracted landmark image is now grayscale and then converted into a BW image in the preprocessing stage.
C. Pixels are grabbed and mapped into specific areas and vector is extracted from the image containing given word or image. This part is considered as processing stage.
D. The Lastly Kohonen Neural Network is used as classification stage.

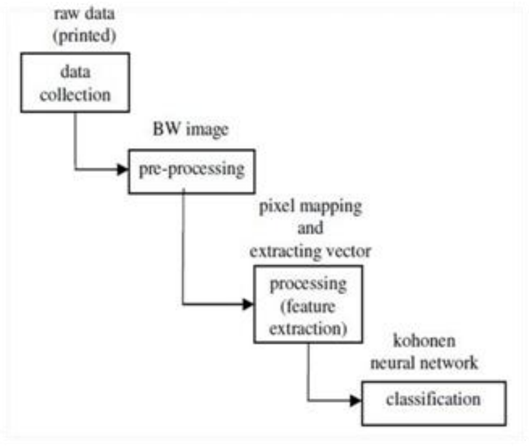


Fig. 4 Image Recognition Procedure Using Kohonen Neural Network

A. DATA COLLECTION:

In this system we need to choose the data type with great care because we need to develop this system according to input raw data or collected data. Here we are talking about character or sign recognition in landmark so obviously we need Landmark Image and also text character. But we are also considering some word as well. But no word level or character level segmentation is considered here rather whole single character image or single word image is taken as raw input. And before entering into the system the image is resized into 250 X 250 pixels to satisfy the procedure. No matter, whether it's text character or sign image. Here the whole text or sign is taken and trained. Because there are lots of features, irregular shapes and curvatures in image of landmarks as we have seen. And still no general formula is generated for feature extraction from the landmark. So rather than extracting character or individual image from the landmark, we will take whole text or whole landmark image as input data in the first phase of the image recognition system. One thing is really important here. That is the character size of the image. The character shouldn't be partially present on the landmark image and it shouldn't be too small in size.

B. *IMAGE PRE-PROCESSING*:

A digital text image that contains character is generally an RGB image. In pre-processing stage mainly image processing procedures take place. Like grayscale image conversion, binary image conversion, skew correction. Our processing stage depends on pre-processing stages. So we need to design pre-processing steps with great care.

1. RGB to Grayscale Image Conversion: In the pre-processing first stage we are converting the input RGB image into greyscale image. Here we are considering the Othu's algorithm for RGB to grayscale conversion. The algorithm is given below:

1. *Count the number of pixels according to colour (256 colours) and save it to matrix count.*
2. *Calculate probability matrix P of each colour, Pi = counti / sum of the count, where i= 1, 2, … … 256.*
3. *Find matrix omega, omegai = cumulative sum of Pi, where i= 1, 2,..... 256.*
4. *Find matrix mu, mui = cumulative sum of Pi *i, where i= 1, 2,.............256 and mu_t = cumulative sum of P256 * 256*
5. *Calculate matrix sigma_b_squared where, sigma_b_squaredi = (mu_t × omegai − mui) 2 /omegai - (1- omegai)*
6. *Find the location, idx, of the maximum value of sigma_b_squared. The maximum may extend over several bins, so average together the locations.*
7. *If the maximum is not a number, meaning that sigma_b_squared is all not a number, and then the threshold is 0.*
8. *If the maximum is a finite number, threshold = (idx - 1) / (256 - 1);*

2. Grayscale to Binary Image Conversion: In the pre-processing second stage we are converting the grayscale image into a binary image. In a grayscale image there are 256 combinations of black and white colours where 0 means pure black and 255 means pure white. This image is converted to binary image by checking whether or not each pixel value is greater than 255•level (level, found by Otsu's Method). If the pixel value is greater than or equal to 255•level then the value is set to 1 i.e. white otherwise 0 i.e. black.

C. *FEATURE EXTRACTION:*

Next and the most important feature of character recognition is feature extraction. In this system we are considering a few steps for extracting a vector. Our main target is finding a vector from the image. So the image is processed and then binary image is created. So we have only two types of data on the image. Those are 1 for the white space and 0 for the black space. Now we have to pass the following steps for creating 625 length vectors for a particular character or image. Those are:

1. Pixel grabbing.
2. Finding the probability of making a square.
3. Mapped to sampled area.
4. Creating vector.

5. Representing character with a model no.

1. Pixel Grabbing from Image: As we are considering the binary image and we also fixed the image size, so we can easily get 250 X 250 pixels from a particular image containing given text character or sign. One thing is clear that we can grab and separate only character portion of the digital image. In specific, we took a given character contained image. And obviously it's a binary image. As we specified that the pixel containing the value 1 is a white spot and 0 for a black one, so naturally the 0 portioned spots are the original character.
2. Finding Probability of Making Square: Now we are going to sample the entire image into a specified portion so that we can get the vector easily. We specified an area of 25 X 25 pixels. For this we need to convert the 250 X 250 image into the 25 X 25 area. So for each sampled area we need to take 10 X 10 pixels from binary image. We can give a short example for that. Table 2 is the original binary image of 25 X 15 pixels. We need to sample it 5 X 3 pixel area. So, for each area we will consider 5 X 5 pixels from the binary image. Table 3 will show how pixels are classified for finding the probability of making a square.

Table 1. Binary Converted Grid Value

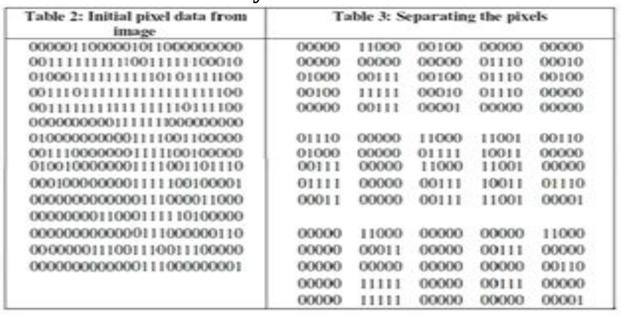


Table 2: Initial pixel data from image	Table 3: Separating the pixels				
00000110000001011000000000	00000	11000	00100	00000	00000
00111111111110011111100010	00000	00000	00000	01110	00010
0100011111111110101111100	01000	00111	00100	01110	00100
00111011111111111111111100	00100	11111	00010	01110	00000
001111111111111111110111100	00000	00111	00001	00000	00000
00000000001111111000000000					
0100000000000111110011000000	01110	00000	11000	11001	00110
0011100000000111111001000000	01000	00000	01111	10011	00000
010010000000011111001101110	00111	00000	11000	11001	00000
000100000000011111100100001	01111	00000	00111	10011	01110
00000000000000111000011000	00011	00000	00111	11001	00001
00000000110001111110100000					
00000000000001110000000110	00000	11000	00000	00000	11000
00000001110011100111000000	00000	00011	00000	00111	00000
00000000000000111000000001	00000	00000	00000	00000	00110
	00000	11111	00000	00111	00000
	00000	11111	00000	00000	00001

3. Mapped to Sampled Area: We can recall the previous example from Table 4 Now the same sample pixel from binary image after separating is shown in Table 5. Now we will find out for each 5 X 5 pixels from the separated pixel portion and gives a unique number for each separated pixel class. And this number will be equal to the 5 X 3 sampled areas. Now we need not consider whether 5 X 5 pixels will make a black area or square or a white area or square. We will take the priority of 0s or 1s from 5 X 5 pixels. And from there we can say, if the 0s get the priority from 5X5 in ith location then we will make a black square on ith position of the sample area. Table 4 is having a unique number of 5 X 5 separated pixels and in Table 5 covering black or white depending on the probabilistic manner.

Table 2. Aligned Marked Value

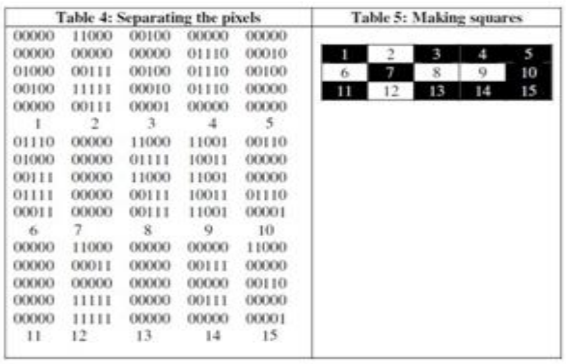


Table 4: Separating the pixels

00000	11000	00100	00000	00000
00000	00000	00000	01110	00010
01000	00111	00100	01110	00100
00100	11111	00010	01110	00000
00000	00111	00001	00000	00000
1	2	3	4	5
01110	00000	11000	11001	00110
01000	00000	01111	10011	00000
00111	00000	11000	11001	00000
01111	00000	00111	10011	01110
00011	00000	00111	11001	00001
6	7	8	9	10
00000	11000	00000	00000	11000
00000	00011	00000	00111	00000
00000	00000	00000	00000	00110
00000	11111	00000	00111	00000
00000	11111	00000	00000	00001
11	12	13	14	15

Table 5: Making squares

■ 1	□ 2	■ 3	■ 4	■ 5
□ 6	■ 7	□ 8	□ 9	■ 10
■ 11	□ 12	■ 13	■ 14	■ 15

Here is an example of how 250 X 250 pixels of English character 'T' is sampled into 25 X 25 sampled area.

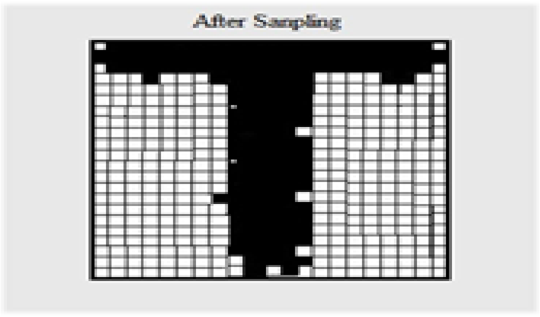
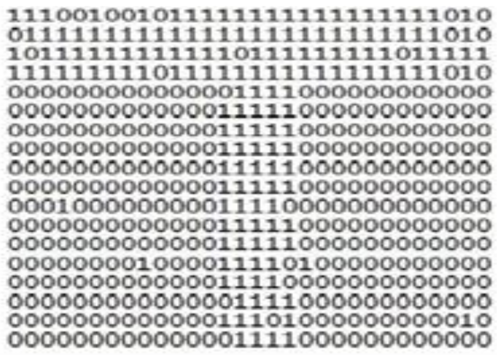


Fig. 5 Sampled Image

4. Creating Vector: Once we have sampled the binary image we have black area and white are. Now we will put a single 1 (one) for each black square and 0 (zero) for each white square. And Figure 5 from above is represented by 1s and 0s combination in Figure 6 below.



```
1110010010111111111111111111010
0111111111111111111111111111010
10111111111111101111111111011111
1111111110111111111111111111010
0000000000000011110000000000000
00000000000000111110000000000000
00000000000000111110000000000000
00000000000000111110000000000000
00000000000000111110000000000000
00000000000000111110000000000000
0001000000000011110000000000000
00000000000000111110000000000000
00000000000000111110000000000000
000000000100000111101000000000000
0000000000000011110000000000000
0000000000000011110000000000000
0000000000000011101000000000010
0000000000000011110000000000000
```

Fig.6 Vector Representation

Now we will collect each row, combine together and it will make a vector. Vector for Figure 6 is given as below.

1110010010111111111111111111010000000111111111111
1111111111111101000000001011111111111111111111110
1111100000001111111110111111111111111111010000000
0000000000000001111000000000000000000000000000000
0111110000000000000000000000000000000001111100000
0000000000100000000000000001111100000000000000000
0000000000000001111100000000000000000000000000000
0011111000000000100000000100001000000000011110000
0000000001000000000000000000001111100000000000000
0000000000000000011111000000010000000000000000000
0010000111101000000000000000000000000000000001111
0000000000000000000000000000000001111000000000000
0000000000000000000000111010000000000010000001000
00000000000001111000000000

5. Representing a Character with a Model Number: One thing we need to mention here. That is, we gave a numerical number as a model for each vector and also the corresponding input word or text character from that particular model. This is because given character is unique length. But we are also considering word which has an irregular length. When we need to train, we will train it with its model no. and model number knows its corresponding input character. So in short we can say that, a particular model has a unique vector with the length of 625 characters of 1s and 0s and a unique character.

D. KOHONEN NEURAL NETWORK:

We did lots of activities in pre-processing stages and as well as in the processing stage. The main idea is to make it simple and acceptable for the Kohonen Neural Network. The Kohonen neural network contains no hidden layer. This network architecture is named after its creator, Tuevo Kohonen. The Kohonen neural network differs from the feed forward back propagation neural network in several important ways.

1. Introduction to Kohonen Network : The Kohonen neural network differs considerably from the feed forward back propagation neural network. The Kohonen neural network differs both in how it is trained and how it recalls a pattern. The Kohohen neural network does not use any sort of activation function. Further, the Kohonen neural network does not use any sort of a bias weight.

Output from the Kohonen neural network does not consist of the output of several neurons. When a pattern is presented to a Kohonen network one of the output neurons is selected as a "winner". This "winning" neuron is the output from the Kohonen network. Often these "winning" neurons represent groups in the data that is presented to the Kohonen network. For example, in this system we consider 10 digits, 5 vowels, 21 consonants, and some signs from total model. The most significant difference between the Kohonen neural network and the feed forward back propagation neural network is that the Kohonen network trained in an unsupervised mode. This means that the Kohonen network is presented with data, but the correct output that corresponds to that data is not specified. Using the Kohonen network this data can be classified into groups. We will begin a review of the Kohonen network by examining the training process.

Consider the vector length is 625 so its input layer has 625 neurons. But in output layer the number of neurons depends on the number of character trained with the network.

2. The Structure of Kohonen Network: The Kohonen neural network contains only an input and output layer of neurons. There is no hidden layer in a Kohonen neural network. First we will examine the input and output to a Kohonen neural network.

The input to a Kohonen neural network is given to the neural network using the input neurons. These input neurons are each given the floating point numbers that make up the input pattern to the network. A Kohonen neural network requires that these inputs be normalized to the range between -1 and 1. Presenting an input pattern to the network will cause a reaction from the output neurons. In a Kohonen neural network only one of the output neurons actually produces a value. Additionally, this single value is either true or false. When the pattern is presented to the Kohonen neural network, one single output neuron is chosen as the output neuron. Therefore, the output of the Kohonen neural network is usually the index of the neuron that fired. The structure of a typical Kohonen neural network is shown in Figure 7.

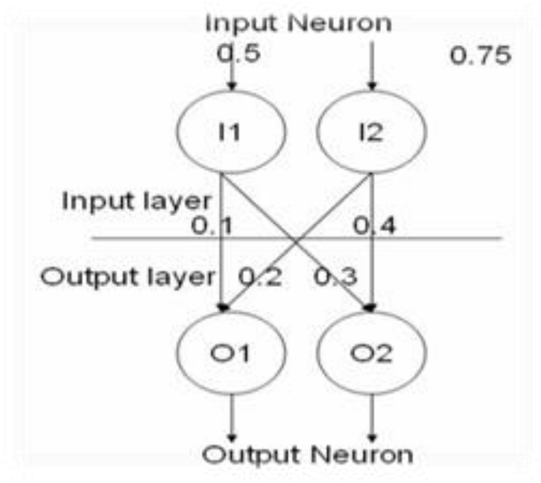


Fig.7 Simple Kohonen Network

Using connection weights between the neuron values we will now examine which neuron would win and produce output. We will begin by normalizing the input.

3. Normalizing the Input: The requirements that the Kohonen neural network places on its input data are one of the most severe limitations of the Kohonen neural network. Input to the Kohonen neural network should be between the values -1 and 1. In addition, each of the inputs should fully use the range. If one, or more, of the input neurons were to use only the numbers between 0 and 1, the performance of the neural network would suffer. To normalize the input we must first calculate the "vector length" of the input data, or vector. This is done by summing the squares of the input vector. In this case it would be. ((0.5 * 0.5) + (0.75 * 0.75)). This would result in a "vector length" of 0.8125. If the length becomes too small, say less than the length is set to that same arbitrarily small value. In this case the "vector length" is a sufficiently large number. Using this length we can now determine the normalization factor. The normalization factor is the reciprocal of the square root of the length. For this value, the normalization factor is calculated as follows, $1/\sqrt{\ 0.8125}$ and this results in a normalization factor of 1.1094. This normalization process will be used in the next step where the output layer is calculated.

4. Calculating Each Neuron’s Output: To calculate the output the input vector and neuron connection weights must both be considered. First the "dot product" of the input neurons and their connection weights must be calculated. To calculate the dot product between two vectors you must multiply each of the elements in the two vectors. We will now examine how this is done.

   The Kohonen algorithm specifies that we must take the dot product of the input vector and the weights between the input neurons and the output neurons. The result of this is as follows.

$$|\mathbf{0.50.75}| \bullet |\ \mathbf{0.10.2}| = (\mathbf{0.5} * \mathbf{0.75}) + (\mathbf{0.1} * \mathbf{0.2})$$

As we can see from the above calculation the dot product would be 0.395. This calculation will be performed for the first output neuron. This calculation will have to be done for each of the output neurons.

Through this example we will only examine the calculations for the first output neuron. The calculations necessary for the second output neuron are calculated in the same way.
This output must now be normalized by multiplying it by the normalization factor that was determined in the previous step. We must now multiply the dot product of 0.395 by the normalization factor of 1.1094. This results in an output of 0.438213. Now that the output has been calculated and normalized it must be mapped to a bipolar number.

5. Mapping to Bipolar: In the bipolar system the binary zero maps to -1 and the binary remains a 1. Because the input to the neural network normalized to this range we must perform a similar normalization to the output of the neurons. To make this mapping we add one and divide the result in half. For the output of 0.438213 this would result in a final output of 0.7191065. The value 0.7191065 is the output of the first neuron. This value will be compared with the outputs of the other neuron. By comparing these values we can determine a "winning" neuron.
6. Choosing a Winner: We have seen how to calculate the value for the first output neuron. If we are to determine a winning output neuron we must also calculate the value for the second output neuron. We will now quickly review the process to calculate the second neuron. For a more detailed description you should refer to the previous section.

The second output neuron will use exactly the same normalization factor as was used to calculate the first output neuron. As you recall from the previous section the normalization factor is 1.1094. If we apply the dot product for the weights of the second output neuron and the input vector we get a value of 0.45. This value is multiplied by the normalization factor of 1.1094 to give the value of 0.0465948. We can now calculate the final output for neuron 2 by converting the output of 0.0465948 to bipolar yields 0.49923.

As we can see we now have an output value for each of the neurons. The first neuron has an output value of 0.7191065 and the second neuron has an output value of 0.49923. To choose the winning neuron we choose the output that has the largest output value. In this case the winning neuron is the first output neuron with an output of 0.7191065, which beats neuron two's output of 0.49923.
We have now seen how the output of the Kohonen neural network was derived. As we can see the weights between the input and output neurons determine this output.

7. Kohonen Network Learning Procedure: The training process for the Kohonen neural network is competitive. For each training set one neuron will "win". This winning neuron will have its weight adjusted so that it will react even more strongly to the input the next time. As different neurons win for different patterns, their ability to recognize that particular pattern will be increased. We will first examine the overall process involving training the Kohonen neural network.
8. Calculating the Errors : Before we can understand how to calculate the error for Kohonen neural network must first understand what the error means. The coming neural network is trained in an unsupervised fashion so the definition of the error is somewhat different involving they normally think of as an error. The purpose of the Kohonen neural network is to classify the input into several sets. The error for the Kohonen neural network, therefore, must be able to measure how well the network is classifying these items.
9. Recognition with Kohonen Network : So for a given pattern we can easily find out the vector and can send it through the Kohonen Neural Network. And for that particular pattern any one of the neuron will be fired. As for all input pattern the weight is normalized so the input pattern will be calculated with the normalized weight. As a result the fired neuron is the best answer for that particular input pattern.

V. RESULTS

The objective of this project is to develop a mapless navigation method for service robots, where this project fulfills this purpose. According to this system, we have conducted a study on the accuracy level for different input data.
The experiments show consistent results with accurate classifications of landmark sign patterns with speech output. Table 3 shows some accuracy level for different input data. The data are considered as trained text or signs, untrained similar, irregular font size.

Table 3. Accuracy Rates Of The System

Text/Sign	Accuracy rate
Trained	100%
Untrained	97%
Irregular font size	98%

As this method is new and we are at the preliminary level of the Optical Character Recognition so the main drawback we can consider is that we need to modify and make it more accurate. Again like all other Neural Network training time increase with the increase in number of characters or words and more landmark signs in Kohonen Neural Network.

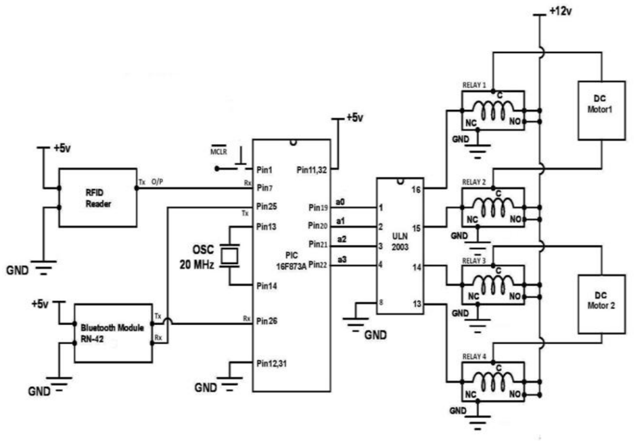


Fig.8 Circuit Diagram of the Proposed System

VI. CONCLUSION

This paper tries to emphasis on a way for mapless navigation method in the simplest possible manner. So we propose a method that applies human navigation system—landmark to fulfill mapless navigation of robots. In this project we have implemented OCR (optical character recognition) to track the landmark by using the Kohonen Neural Network.

But there are also lots of ways to implement OCR that could be more efficient than a Kohonen Neural Network. The main reason to select a Kohonen Neural Network is to reduce the computational cost in order to facilitate the real time implementation. After the locating and tracking of the landmark, we extract the semantic information of texts and arrows contained in those signs, and use the result to guide the robot to the destination.

ACKNOWLEDGMENT

I would like to express my gratitude to my guide Dr. Neelakantha. V. Londhe, Professor, Department of Mechanical Engineering, Srinivas Institute of Technology, Mangalore, for his guidance, encouragement and support for successful completion of this project work. I would like to thank the almighty, for always being there for me and guided me to work on the right path of life. My greatest thanks are to my parents who bestowed ability and strength in me to complete this work.

REFERENCES

1. R.C. Gonzalez and R.E. Woods, Digital Image Processing, second edition, Pearson Education, 2005
2. F. Moutarde, A. Bargeton, A. Herbin, and L. Chanussot, “Robust on vehicle real-time visual detection of American and European speed limit signs, with a modular traffic sign recognition system,” in Intelligent Vehicles Symposium, IEEE 2007 .
3. C. Keller, C. Sprunk, C. Bahlmann, J. Giebel, and G. Baratoff, “Real-time recognition of us speed signs,” in Intelligent Vehicles Symposium, IEEE 2008 .
4. G. Qingji, Y. Yue, and Y. Guoqing, “Detection of public information sign in airport terminal based on multi-scales spatio-temporal vision information,” International Conference on IEEE 2008.
5. J. Maye, L. Spinello, R. Triebel, and R. Siegwart, “Inferring the semantics of direction signs in public places,” in Robotics and Automation (ICRA), IEEE 2010.
6. T. Breuer, G. Giorgana Macedo, R. Hartanto, N. Hochgeschwender, D. Holz, F. Hegger, Z. Jin and G. Kraetzschmar “Johnny: An autonomous service robot for domestic environments,” Journal of Intelligent &amp; Robotic Systems, Jul. 2011.
7. I. Posner, P. Corke, and P. Newman, “Using text-spotting to query the world,” in Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on IEEE 2010.
8. Auranuch Lorsakul , Jackrit Suthakorn “Traffic Sign Recognition for Intelligent Vehicle/Driver Assistance System Using Neural Network on OpenCV”. International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2007).
9. J. Heaton, Introduction to Neural Network in Java, HR publication, 2010.
10. N. Otsu, “A Threshold Selection Method from Gray-Level Histograms”, IEEE Transactions on Systems, Man, and Cybernetics, 2001.
11. A.K. Jain and T. Taxt, Feature Extraction Methods for Character Recognition - a survey, Michigan State University, 2001.
12. Adnan Md. Shoeb Shatil, “Research Report on Bangla Optical Character Recognition Using Kohonen Network”, BRAC University, 2005.