

SELF-LOCALIZATION BY USING ARTIFICIAL NEURAL NETWORKS FOR
HUMANOID ROBOT NAO



by
Yusuf Can Semerci

Submitted to Graduate School of Natural and Applied Sciences
in Partial Fulfillment of the Requirements
for the Degree of Master of Science in
Computer Engineering

Yeditepe University
2015

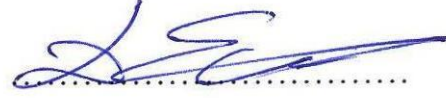
SELF-LOCALIZATION BY USING ARTIFICIAL NEURAL NETWORKS FOR
HUMANOID ROBOT NAO

APPROVED BY:


Assoc. Prof. Dr. Emin Erkan Korkmaz
(Thesis Supervisor)



Assist. Prof. Dr. Dionysis Goularas
(Thesis Co-Supervisor)



Prof. Dr. Cem Ünsalan



Assoc. Prof. Dr. Şima Etaner Uyar



Assist. Prof. Dr. Onur Demir



DATE OF APPROVAL:/...../2015

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisors Assoc. Prof. Dr. Emin Erkan Korkmaz and Assist. Prof. Dr. Dionysis Goularas for their guidance, suggestions, insight and encouragement towards this project. I also thank Erman Gönül for his additional work on the subject.

To my family, I offer sincere gratitude for their unconditional support. I also thank Ali Yılmaz, İsmail Uğur Bayındır and Çağrı Yeşil for their useful discussions and great friendship.

ABSTRACT

SELF-LOCALIZATION BY USING ARTIFICIAL NEURAL NETWORKS FOR HUMANOID ROBOT NAO

NAO is a humanoid robot that is widely used in robot soccer games. Position estimation is an important process in such robotics applications. It can be defined as finding the position of the robot in a known environment. The current solutions proposed in the literature usually utilize proximity markers that provide the necessary information to determine the position of the robot. However, self-localization without using an external marker is a challenging problem. In this study, a novel approach that is based on Artificial Neural Network(ANN) learning is proposed for the self-localization problem of robot NAO on a soccer field. The method uses images captured by the vision sensors of the robot and a supervised learning process is carried out in order to obtain a self localization system. Some image processing methods are also utilized in order to extract the features that are used in the learning process. Various tests are carried out and it has been observed that the NAO robot can estimate its position on the soccer field quite accurately.

ÖZET

YAPAY SİNİR AĞLARI KULLANARAK İNSANSI ROBOT NAO'DA KENDİLİĞİNDEN YER BULMA

NAO robot futbol yarışmalarında yaygın olarak kullanılan bir insansı robottur. Pozisyon tahmini bu tür robotik uygulamalarda önemli bir süreçtir. Pozisyon tahmini, bir robotun ortamdaki yerini belirleme olarak tanımlanabilir. Literatürde önerilen güncel çözümler robotun konumunu belirlemede gerekli bilgileri sağlamak için yakınlık işaretçileri kullanırlar. Ancak, harici işaretçi kullanmadan kendi lokasyonunu bulma zorlayıcı bir problemdir. Bu çalışmada, NAO'nun bir futbol sahasında kendi yerini bulma problemi için Yapay Sinir Ağları (YSA) ile öğrenmeye dayalı yeni bir yaklaşım önerilmiştir. Yöntemde, robotun görme sensörleri tarafından çekilmiş resimleri, kendi yerini bulma sistemi elde etmek için oluşturulan denetimli öğrenme sürecinde kullanılmaktadır. Öğrenme sürecinde kullanılan özel nitelikler bir takım görüntü işleme yöntemleri kullanılarak elde edilmektedir. Yapılan çeşitli testler sonucunda NAO'nun bir futbol sahasında kendi yerini oldukça doğru bir şekilde tespit edebildiği görülmüştür.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	III
ABSTRACT.....	IV
ÖZET	V
LIST OF FIGURES	VIII
LIST OF TABLES	X
1. INTRODUCTION	1
2. BACKGROUND	4
2.1. PROBLEM DEFINITION	4
2.2. OVERVIEW OF LOCALIZATION METHODS	5
2.2.1. OUTDOOR LOCATION ESTIMATION	5
2.2.2. INDOOR LOCATION ESTIMATION	5
2.3. IMAGE PROCESSING BACKGROUND	8
2.3.1. RGB TO HSV CONVERSION	9
2.3.2. HSV TO RBG CONVERSION	9
2.3.3. RGB TO GRAY CONVERSION	10
2.3.4. MORPHOLOGICAL DILATION	10
2.4. ARTIFICIAL NEURAL NETWORKS	11
3. METHODOLOGY	13
3.1. FEATURE EXTRACTION	15
3.1.1. FIELD EXTRACTION.....	15
3.1.2. FIELD LINE DETECTION.....	19
3.1.3. FEATURE EXTRACTION	21
3.2. ARTIFICIAL NEURAL NETWORK LEARNING	22
3.2.1. POSITION ESTIMATION.....	24
4. IMPLEMENTATION	26
5. TEST AND EVALUATION	30
6. CONCLUSION	41

REFERENCES42



LIST OF FIGURES

Figure 2.1.Dilation	11
Figure 2.2. An example Artificial Neural Network	12
Figure 3.1. Flow chart of proposed method.....	14
Figure 3.2. An example image taken from NAO.....	15
Figure 3.3. The thresholded binary image	17
Figure 3.4. Noise regions detected in the image.....	17
Figure 3.5.The image with outside the green cleaned	18
Figure 3.6. Field Extraction images	19
Figure 3.7. The image in Gray Scale	20
Figure 3.8. Field Line Detection images.....	21
Figure 3.9. Angle assignment according to orientation	24
Figure 4.1. The Graphical User Interface	26
Figure 4.2. The Class Diagram	27
Figure 4.3. The Use Case Diagram.....	29
Figure 5.1. The football field	30

Figure 5.2. The regions and the middle points	31
Figure 5.3. Layer labels of regions	33
Figure 5.4. Test results of robot facing goal	35
Figure 5.5. Test results of robot facing left and right	36
Figure 5.6. Test results for all obtained images	38
Figure 5.7. Test result when another robot is present in the scene	40

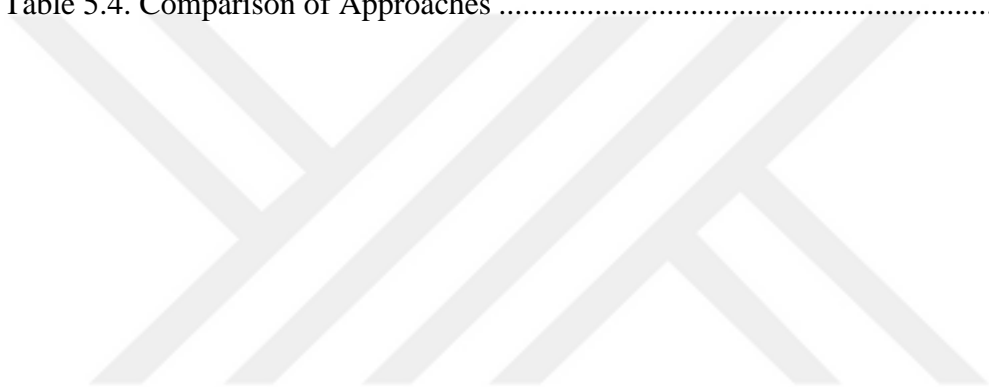
LIST OF TABLES

Table 5.1. NAO Specifics30

Table 5.2. Error color codes.....33

Table 5.3. Mean errors37

Table 5.4. Comparison of Approaches39



1. INTRODUCTION

Constructing a model of the environment is an important problem for robotics applications. Navigation of a robot in a certain environment is an application where such a model is needed crucially. Determining the position of the robot in the environment is critical for a successful navigation. This study handles the position estimation problem which can be described as finding the position of a robot in a known environment.

The existing solutions for position estimation can be divided into two categories. In the first category, the common approach is to use proximity markers placed in the environment or tools like Global Positioning System (GPS) and digital compasses provide the information to position the robot. However, it is a more challenging approach to handle the position estimation problem as a self-localization process which is based on only the data that can be collected by the robot sensors and without using any special markers placed in the environment.

This study proposes a vision based self-localization approach for the humanoid robot NAO. The approach uses the images that are collected by the camera that exists on NAO's head. An Artificial Neural Network (ANN) learning process is utilized in order to constitute the position estimation system for the robot. The supervised learning process is carried out on a training data set that consists of a set of images taken in previously known positions and in different directions. The raw images are manipulated by a set of image processing techniques in order to extract a set of features that represent the image signature.

The resulting system is tested on a set of images taken in random positions and with random directions. It has been observed that the robot can estimate its position and direction quite accurately on these test images, too. The results suggest that ANNs provides the required abstraction over the training set which can be used to achieve self-localization for the robot.

There are other studies in the literature that consider position estimation problem as self-localization process. These existing solutions can be divided into two groups: relative localization and absolute localization. Relative localization can be described as determining the position by using the movement data of the robot [7,17,18,19]. The data related to the environment is not considered in this approach. On the other side, absolute localization is carried out based on the data obtained from the environment.

A common absolute localization approach is using matching algorithms on images taken by the robot. In this approach, a set of images are taken for reference points and the new images are compared with them [4,5,9]. Iterative Closest Point [20] is the most commonly used method for the matching process [5]. However, matching algorithms require a high computation time and sometimes sufficient overlapping could not be achieved with the previous images. In [4], such an approach is utilized and satisfactory results are obtained. However, an omnidirectional camera that is capable of taking 360° images of the field, is used in the study. This is achieved by a down-facing camera placed above the head of the robot. It can be claimed that, it would be difficult to achieve the same success level with the standard front facing cameras that exist on humanoid robots.

As noted before, it is very common to use some external markers or some other tools to determine the position of a robot. The methods that use the data obtained from tools like Global Positioning System (GPS) or digital compasses in order to achieve the localization of a robot in outdoor environment, can be found in [11,1].

Proximity and vision sensors are usually used for finding the the position of a robot in an indoor environment. Proximity based approaches can use sonars and lasers to determine the position of the robot [10,15]. On the other hand, vision sensor based approaches use cameras for this task. Using vision sensors has the advantage of being low-cost and providing huge amount of information. In this approach, position estimation is achieved by using colored landmarks. The environment is usually surrounded with predefined landmarks. The robot calculates the distance to the detected landmarks. The positions of the landmarks are known by the robot. Hence, geometrical calculations, [6,16] and matching algorithms [3] could be used to determine the position of the robot whenever

landmarks are detected. Although the approach is effective, it may have some disadvantages. For instance, other objects in the scene similar to the landmarks can be confusing for the robot.

This thesis is organized as follows: Preliminary information about Image Processing techniques and Artificial Neural Networks and the related work are presented at the next chapter. The proposed method consists of the Feature Extraction and the ANN learning processes and the details of them can be found in Chapter 3. The experimental results are given in Chapter 4. Last chapter contains the discussions and the conclusions for the study.



2. BACKGROUND

2.1. PROBLEM DEFINITION

Navigation of a robot in an environment is an important problem in robotic research. In order to perform a successful navigation, the robot first needs to know its initial position. For a robot, self-localization can be described as an ability to determine its position inside an environment.

When a robot tries to locate itself in an outdoor location, the GPS technology is sufficient to find the position of the robot with the help of longitude and altitude values. This gives the information needed to establish a navigation path in the environment.

On the other hand, when the robot tries to find its location in an indoor location such as a building or a room, the longitude and altitude information becomes insufficient, since the GPS precision is not high and the different positions inside a room would give almost the same information. In order to solve this indoor self-localization problem, a robot needs to have some a priori information related to the environment.

The information about indoor locations can be obtained from various sensors such as cameras, lasers and sonars. Sonars and lasers are proximity sensors that are efficient in establishing a mapping of the environment by using the objects or walls in the environment. Then, the data obtained from these objects and their relative positions are essential in order to solve the self-localization problem. Cameras are vision sensors and they can be used also to obtain information related to the environment. However, the data collected by the cameras needs to be processed in order to obtain the mapping of the environment.

2.2. OVERVIEW OF LOCALIZATION METHODS

2.2.1. OUTDOOR LOCATION ESTIMATION

GPS works with satellites and a GPS receiver could be installed on the desired agent such as a robot. Initially the receiver gets a distance value from one satellite. This distance corresponds to a radius of a sphere where the robot is supposed to be inside. Then two other satellites give two additional distances corresponding to the radius of two other spheres. The intersection of the three spheres gives the location of the robot. This process is called triangulation. In this process the satellite velocities and altitudes are regulated and checked constantly in case of an error. If there is an error in the position of a satellite, the calculated error is sent to the satellite with a timestamp so that the receivers can update the location information of the satellites.

2.2.2. INDOOR LOCATION ESTIMATION

As noted before, the indoor location estimation for a robot can be done with two types of sensors: Proximity sensors and Vision sensors. Proximity sensor approach measures the distances to objects in the scene with the use of sonars and lasers. Vision sensor approaches can be divided into subcategories according to the type of camera used and the position of the camera on the robot: omnidirectional cameras, single camera placed on robot's head facing downward and single camera facing forward.

The use of proximity sensors is proven to be an efficient way to obtain the presence information for an object in an environment [10,15]. These sensors use particular techniques to achieve self-localization for robots. One of them, the sonars, became essential for robots. Nowadays, most of the humanoid robots have sonar transmitters and receivers embedded in their bodies. Another type of proximity sensor approach is based on laser beams which are placed on the robot separately; however these sensors are quite expensive.

In order to reconstruct the environment in a digital form, a robot needs to analyze its surroundings. For the detection of objects inside a scene, a strategy has to be chosen in order to scan the environment. It is only with this information that a robot can later determine its position.

When a robot tries to estimate its position with the use of sonars, it transmits signals that travel through the environment. The corresponding receivers on the robot's body collect the signal reflections occurring when the signal makes contact with an object. The distances are measured from the elapsed time to receive the reflected signal. With this procedure the robot obtains the required data to reconstruct the environment in a digital form. Then the reconstructed model is compared with a previously known mapping. The information about the surroundings of the robot and the mapping of the environment is sufficient enough to estimate the position of the robot. With robots having laser sensors, the self-localization process is the same but it is more accurate.

In recent years the use of omnidirectional cameras has increased in non-humanoid robots. The reason is that when an omnidirectional camera is placed on top of a robot, important information about the surroundings is collected.

In [3], an omnidirectional camera takes images from the whole field in a range of 360 degrees. An initial calibration is performed for detecting various elements inside the scene like lines, obstacles and objects. This calibration is done manually and the result is placed into a look-up table. Later the elements of this calibration are being compared with the new images in order to obtain information about the actual position.

After the images are calibrated, color transitions must be stored and transformed into 2D coordinate data containing pixel locations and corresponding color information. In order to do this step, the dimensions of the field must be known. Then, distance and gradient matrices are calculated. These matrices are stored in robot's memory to be compared with the new images that will be taken by the robot for the localization process.

Nevertheless, the information collected by the camera is not sufficient to obtain the above matrices with a satisfactory precision. For this reason, 2D laser information is also used. The data from the laser is used to obtain the distance matrix that can be defined as the matrix holding the distance information to the lines, obstacles and objects in the scene. The gradient matrix is calculated with the errors of distance calculations, where an error is calculated from data collected continuously by the constant moves of the head. The comparisons on the data collected are recorded as the cost of a single position. Then the costs are recorded into a matrix that is called gradient matrix.

For the position estimation, several hypotheses generated for the coordinates on a 2D plane. Each hypothesis is evaluated by using the matrices obtained by the calibration process and the hypothesis with the minimum distance and gradient error is determined. First the hypotheses having big errors are discarded. Then similar hypotheses are merged and the one having the minimum error gives the approximated location of the robot.

In [2], a single regular camera placed on robot's head is used to take images from the field. The field is constructed in a way that the robot can estimate its position from distance measurements based on particular landmarks. These landmarks are placed in the middle of the field at each side. In order to estimate its position, the robot first has to detect at least one of these landmarks.

After the landmark is detected, the robot needs to locate the upper-left, upper-right, lower-left and lower-right corners and the middle of the detected landmark. The robot tilts its head until it obtains all the information mentioned above. When the information from the landmark detection process is collected, the robot calculates the distance to the detected landmark.

While calculating this distance, the robot uses the height of the camera and also the angle of the camera relative to the floor. The angle and the height values can be disturbed by noise, resulting a poor estimation position. In order to improve estimation quality, a neural network approach is proposed in the study.

The input of the neural network consists of some known distances to the landmark and landmark pixel size pairs in the picture taken at this distance. The output is the corresponding angle and height information. After the training process, the neural network is expected to produce accurate estimations for the height and angle values.

Since the position of a landmark is known, these calculations will be sufficient to obtain the position of the robot. It is possible to determine the distance (r) to the landmark by multiplying tangent of the angle and the height (Equation 2.1.) Finally, the position of the robot (x', y') is calculated as in equation (2.2).

$$r = h \times \tan \theta \quad (2.1)$$

$$\begin{aligned} x' &= x + r \cos \theta \\ y' &= y - r \sin \theta \end{aligned} \quad (2.2)$$

2.3. IMAGE PROCESSING BACKGROUND

A digital image consists of pixels which are the smallest elements of an image. The collection of the pixels forms a matrix that can be considered as the digitalization of an analog image. The images can generally be represented in three forms: 1-bit monochrome, 8-bit gray scale and 24-bit color.

1-bit monochrome images are called binary images and the pixels of these images can have only two values. The pixels of an object are represented as 1's and they are called foreground pixels while the remaining pixels are represented as 0's and they form the background.

A pixel with an 8-bit gray scale image holds only the intensity information. The value of a pixel varies between 0 and 255 that shows the intensity in shades of grey. In this representation the value 0 corresponds to black and 255 corresponds to white.

A 24-bit color (or true color) representation consists of three layers of 8-bit grayscale images in which every layer has a different color model. There are different color models.

The most widely used model is RGB and it can be also transformed to the HSV color model. RGB stands for Red-Green-Blue and these are the colors utilized in the three layers. HSV stands for Hue-Saturation-Value. The Hue layer representing the color information, the Saturation layer representing the purity of a color and the brightness values represented by the Value layer, composes finally the HSV color model. [21]

2.3.1. RGB TO HSV CONVERSION

When converting a pixel from RGB to HSV the H, S and V values are calculated as functions of R, G and B values. Before the conversion, the RGB values are scaled to fit the range $0 < RGB < 1$. Then the calculation for H, S and V values are done according to the equations in 2.3. [22]

$$\begin{aligned}
 V &= \max(R, G, B) \\
 S &= \begin{cases} V - \min(R, G, B) & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases} \\
 H &= \begin{cases} \frac{60(G - B)}{V - \min(R, G, B)} & \text{if } V = R \\ 120 + \frac{60(B - R)}{V - \min(R, G, B)} & \text{if } V = G \\ 240 + \frac{60(R - G)}{V - \min(R, G, B)} & \text{if } V = B \end{cases} \quad (2.3) \\
 &\text{if } H < 0, H = H + 360 \\
 V &= 255V, S = 255S, H = \frac{H}{2}
 \end{aligned}$$

2.3.2. HSV TO RBG CONVERSION

When calculating RGB values from HSV values, Hue value should be between 0 and 360 and Saturation Value between 0 and 1. In order to fit the values to these scales Saturation and Value are divided by 255 and Hue is multiplied with 2. After the scaling, the calculations are done according to the equations in 2.4. [22]

$$C(\text{Chroma}) = V \times S$$

$$H' = \frac{H}{60}$$

$$X = C (|1 - |H(\text{mod}2) - 1||)$$

$$(R_1, G_1, B_1) = \begin{cases} (0,0,0) & \text{if } H' \text{ is undefined} \\ (C, X, 0) & \text{if } 0 \leq H' < 1 \\ (X, C, 0) & \text{if } 1 \leq H' < 2 \\ (0, C, X) & \text{if } 2 \leq H' < 3 \\ (0, X, C) & \text{if } 3 \leq H' < 4 \\ (X, 0, C) & \text{if } 4 \leq H' < 5 \\ (C, 0, X) & \text{if } 5 \leq H' < 6 \end{cases} \quad (2.4)$$

$$m = V - C$$

$$(R, G, B) = (R_1 + m, G_1 + m, B_1 + m)$$

2.3.3. RGB TO GRAY CONVERSION

The conversion to gray scale is achieved by applying the transformation equation to each pixel one by one. The transformation equation is as follows:

$$\text{Gray} = \frac{R + G + B}{3} \quad (2.1)$$

2.3.4. MORPHOLOGICAL DILATION

The morphological dilation is used for enlarging foreground information boundaries [14]. In image processing, it is used with binary images for enlarging the pixels with the value one in order to clean the noises in the image. In the morphological dilation operation, the pixel values of the output image are determined by a rule concerning the corresponding pixel in the input image and its neighbours. The neighbouring pixels can be defined with different rules. The most common used neighbouring rule is a cross shaped neighbourhood and it can be seen in Figure 2.1.a. In a binary image, if any of the neighbouring pixels of

the input image is set to the value 1, the corresponding output pixel is also set to 1. (Figure 2.1.b)

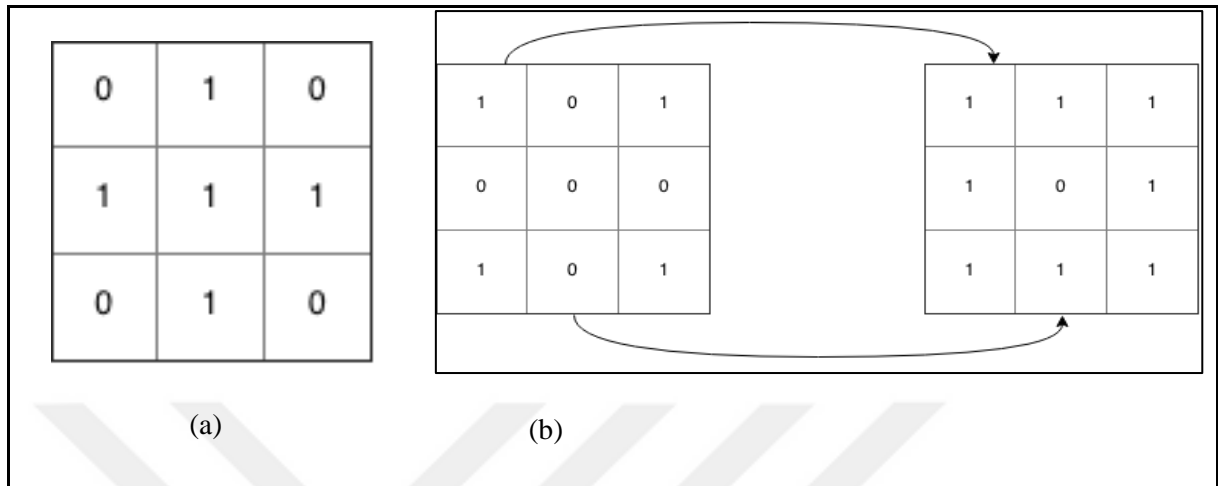


Figure 2.1. Dilation a) Cross-shaped neighborhood. b) An example dilation.

2.4. ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks (ANN) provide a robust technique which is used in different machine learning applications. The model is inspired from the biological neurons and it can be used for classification and approximation problems. ANNs can be simply described as a set of nodes and their inter-connections that form a network (Figure 2.2). Feed Forward Neural Network is the simplest model of ANNs where the network is organized as a combination of different neuron layers. [23] A neuron in a layer is connected only with the neurons of the next layer and no cycles exist in the network. A Standard feed forward ANN is composed of three layers: The Input layer, the Hidden layer and the Output layer. Input-output pairs are represented with the Input and Output layers. The Hidden layer represents the transformation layer which transforms the input into output form. It is possible to utilize more than one Hidden layer in order to represent more complicated functions.

The pattern of interconnections differs according to the problem that is being solved by the ANN. The nodes in these layers are called neurons and they store a weight value that determines the size of the signal they transfer via their connections to other neurons. The

number of neurons in each layer differs according to the data provided during the learning process. Every neuron has an activation function that converts the weighted input of the neuron to its output activation. The number of neurons in each layer differs according to the data provided during the learning process. Every neuron has an activation function that converts the weighted input of the neuron to its output activation.

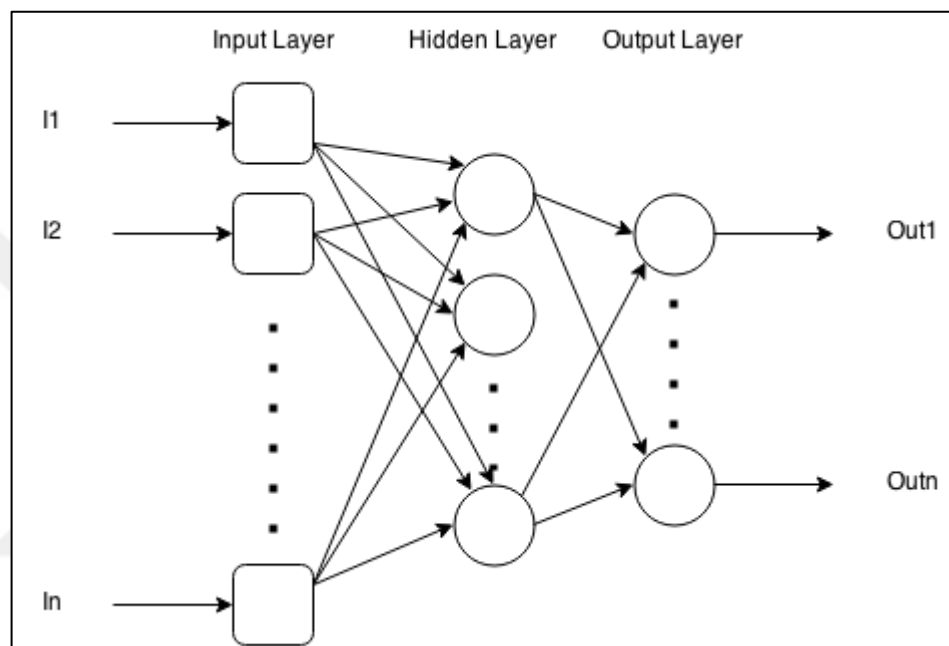


Figure 2.2. An example Artificial Neural Network.

In order to train an ANN, an update algorithm is used to update the weights associated with the edges connecting the neurons. An update algorithm updates the weights based on the error measured on the output layer for the elements of the training data. In supervised learning a set of input-output pairs are given to the ANN and a function is induced by the network which can correctly map the input to the desired output. The ANN could be used for the classification of new data after the error measured at the output of the ANN is minimized by the iterative weight update algorithm. [24]

3. METHODOLOGY

The proposed method utilizes a two phase algorithm for the solution of self-localization problem. In the training phase, a learning process is carried out on the robot's environment. Then the robot can use the abstraction obtained in the learning phase to estimate its position in new circumstances.

In the training phase, a set of pictures taken by the robot's camera are utilized. These pictures are taken at predetermined positions that cover different regions of the soccer field. Also the same set of directions is used at each position. Each image is labeled with the corresponding position and direction information in order to create the training data. Each image is also processed by using a set of image processing techniques in order to extract the features to be used in the training process. As noted in chapter 1, Artificial Neural Networks (ANNs) are used in the learning process. The features that generate the image signature and the position-direction information related to the image form the final training data set for the training process. (Figure 3.1)

The training phase is divided into two processes which are direction learning and position learning. In the first phase, the position information in the training data is ignored and an ANN is trained by using only the direction information as the class label of the images. Hence this initial network can determine the direction of the robot given an image signature. Then the training data is divided into three subsets according to the three directions used when the images are taken. The images were taken when the robot is facing the opponent's goal net straight ahead and with an angle of 45° to the left or to the right. Then a separate ANN is trained for each subset. Certainly, the position information is utilized as the class label in this phase. Hence, each ANN is expected to estimate the position of the robot when an image that has the same direction with the training data is given to the network.

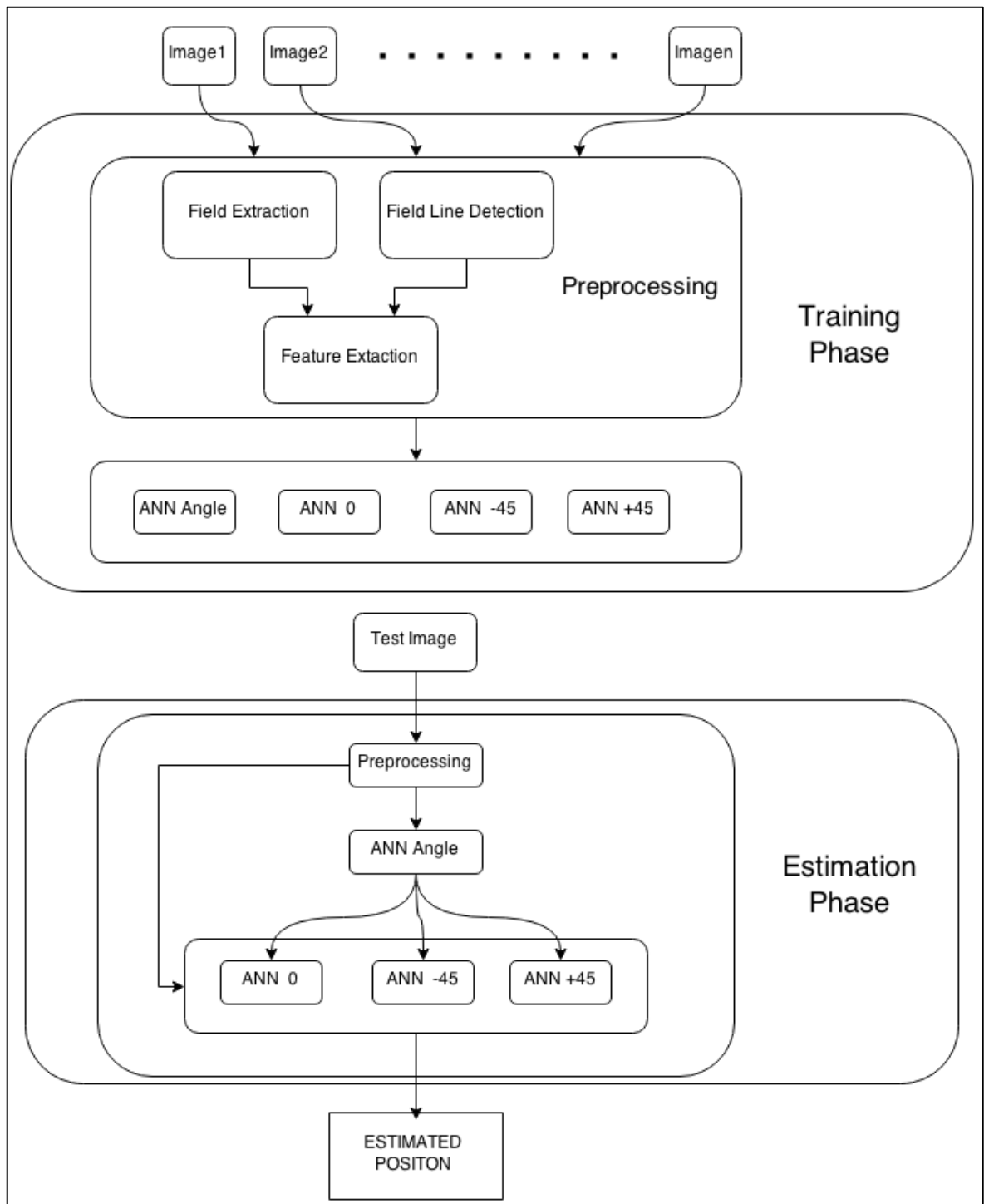


Figure 3.1. Flow chart of proposed method.

In the estimation phase, when a new image is received, the same feature extraction process is applied to generate the signature for that image. Then this signature is firstly sent to the ANN which is trained for estimating the direction of the robot. Depending on the result, one of the ANNs trained for position estimation is chosen and the same signature is sent to

the selected ANN this time. The output of this network provides the final position estimation for the robot.

3.1. FEATURE EXTRACTION

The feature extraction method is explained in detail in this section. It is not possible to use the whole images directly in the learning process. Consequently, each image has to be represented with a set of features named as image signature. The signature generation process is composed of the following steps: Field Extraction, Field Line Detection and Feature Generation.

3.1.1. FIELD EXTRACTION

In order to extract some features from an image, firstly the image has to be simplified. (Figure 3.2) The simplification process is done by converting the image from a Red-Green-Blue (RGB) color space representation into a binary form where the green field will be represented as white and the rest of the image as black.



Figure 3.2. An example image taken from NAO.

Color detection in RGB color space is a difficult process. The detection of darker or lighter tones of a color is problematic in this representation. Different tones can occur because of the changes caused by the different light conditions such as casting a shadow on the scene.

Hence the image needs to be converted to Hue-Saturation-Value (HSV) color space in order to eliminate shadow changes and tone differences. In HSV color space such issues can be handled by ignoring the V(Value or Luminance) value.

After the image is converted into HSV color space, the green field needs to be extracted from the image. For this purpose three threshold functions are utilized on the HSV layers of the image. The threshold function returns 1 if the value of the given pixel is in a specified range and 0 if it is not. After several trials, the best range of green values is measured to be between $HSV(46,20,30)$ and $HSV(126,255,255)$. The threshold values are applied to the image after separating the HSV layers. When the equation 3.1 is applied for the Hue layer, the threshold function returns 1 for the Hue values that are in the range [46,126]. Equation 3.2 is applied to the Saturation layer and 1 is returned for Saturation values that are between 20 and 255. Finally, equation 3.3 is applied to the Luminance layer and 1 is returned for the values between 30 and 255 this time.

$$H' = \begin{cases} 1 & \text{if } 46 \leq H \leq 126 \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

$$S' = \begin{cases} 1 & \text{if } 20 \leq S \leq 255 \\ 0 & \text{otherwise} \end{cases} \quad (3.2)$$

$$V' = \begin{cases} 1 & \text{if } 30 \leq V \leq 255 \\ 0 & \text{otherwise} \end{cases} \quad (3.3)$$

As mentioned above, the threshold functions are applied on the HSV image and the calculated layers results a new binary image (Figure 3.3) As seen in the figure, the white pixels represent the green field and the black pixels are the rest of the colors in the image.

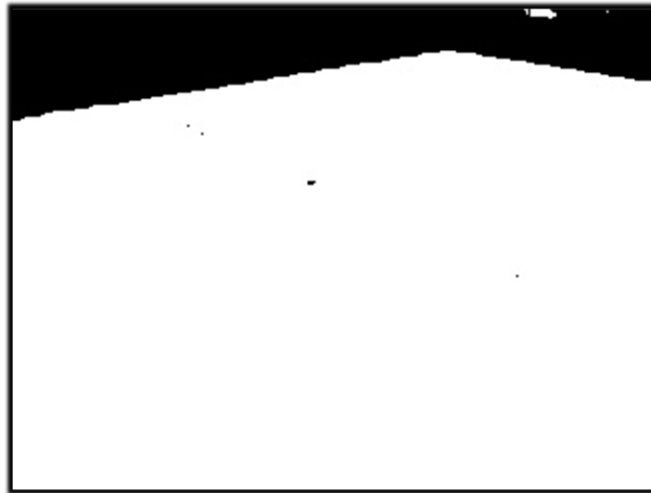


Figure 3.3. The thresholded binary image.

A certain amount of noise is encountered in the images. This can occur both outside and inside the green field. In order to eliminate the noise in an image, the small white and black regions in the image are detected by the blob detection algorithm (Figure 3.4). In the algorithm, every white pixel is compared with the neighborhood pixels. The white neighbor pixels are called to be connected to the reference pixel. All connected pixels are labeled with a color of choice to represent different connectivities. The algorithm for labeling these connected components is named as flood fill algorithm [8].

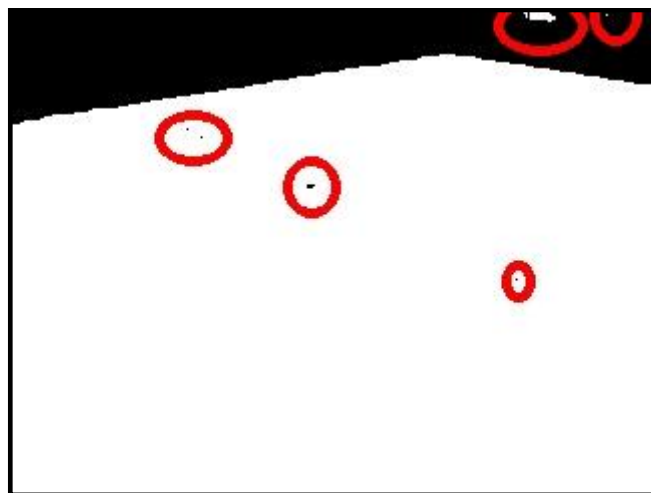


Figure 3.4. Noise regions detected in the image.

In the flood fill algorithm, a target color and a replacement color are determined and a reference point is given to the algorithm. If the reference point has the target color, then the neighbors that have the same value will be connected to the reference point. For this

purpose, the color of reference point is set as the replacement color. In a four direction neighborhood system, the neighbors that have the target color are also set with the replacement color. Then the same process is repeated by setting the neighbors as reference points one by one until there are no neighbors with the target color. When this process is finished, the points that have the replacement color will be named as a blob. Then the process continues to find blobs with new replacement colors and new reference points.

The maximum sized blob represents the green field pixels. The components smaller than the maximum sized blob need to be deleted. This deletion is done by making all the pixels in the smaller components black. As a result, the regions outside the field are cleaned completely (Figure 3.5).



Figure 3.5. The image with outside the green cleaned.

In order to remove the noise inside the field, the image is inverted such that the white pixels become black and black pixels become white (Figure 3.6.a). The blob detection algorithm is executed on the negative image this time. Now the total region outside the field will be the maximum blob that is detected by the algorithm. The smaller components which are the noises inside the field now can be deleted as described before (Figure 3.6.b). Finally the image is inverted again to obtain the final noise free field image (Figure 3.6.c).

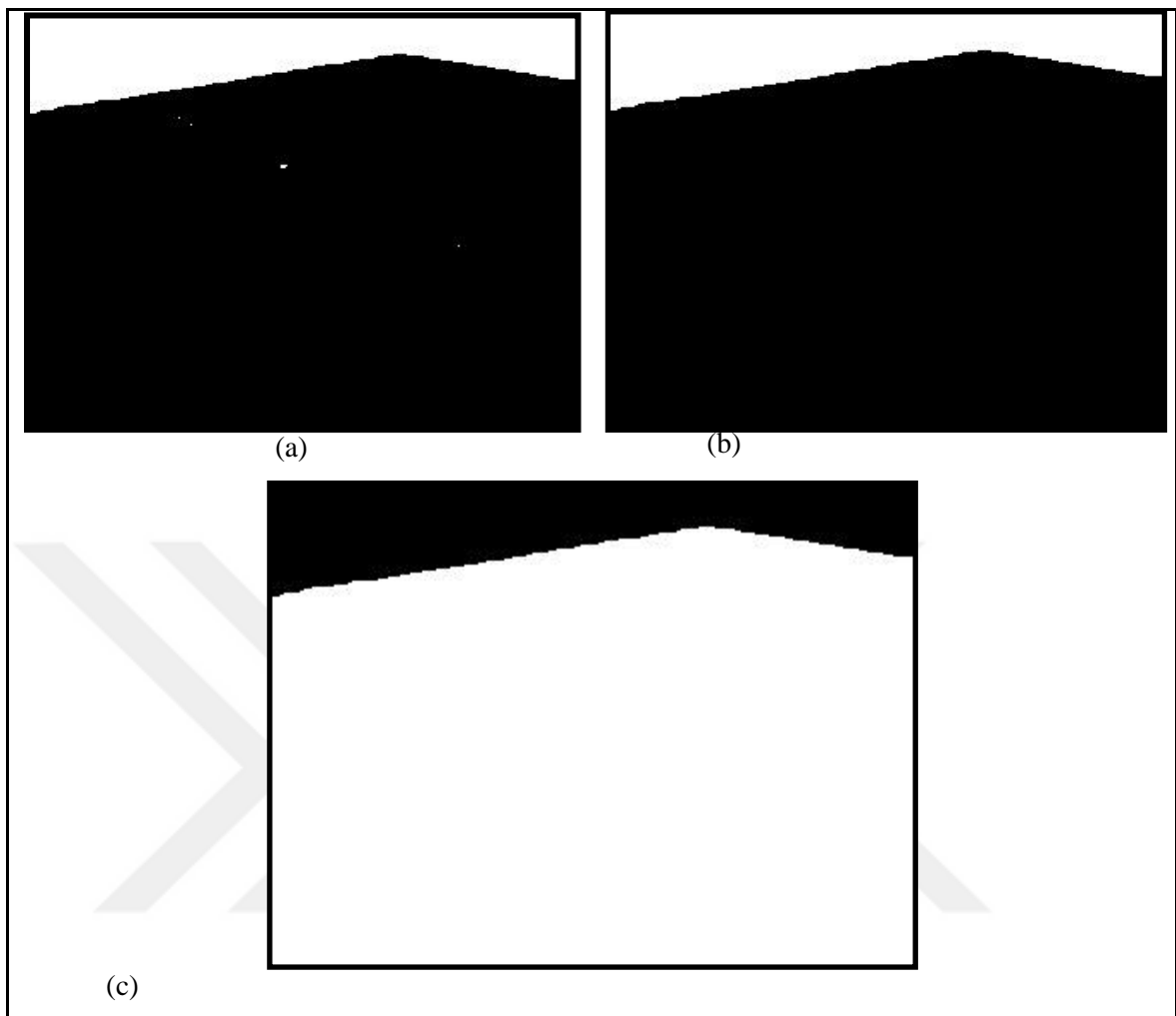


Figure 3.6. Field Extraction images (a) Inverted image. (b) Removal of small white components.(c) Resulted image.

3.1.2. FIELD LINE DETECTION

For the signature generation process, the lines that exist in the field are also utilized. Hence, these lines need to be detected in the image in order to increase the accuracy of the signature. White detection process can be best done by taking the white values in a grayscale image. For this task, gray scale images are used. Since the white lines reside only inside the field, the process is carried out by using the field pixels and the rest of the image is discarded.

The binary image generated after the field detection process can be used as a mask to determine the field region in the original HSV color space image. The coordinates of the black pixels are obtained from the binary image. In the original HSV image the Luminance(V) values of the pixels in these coordinates become 0. Then the HSV image becomes ready to be converted to gray scale. However, the HSV image is firstly converted back to RGB and then the Gray Scale conversion is applied on this RGB image (Figure 3.7).

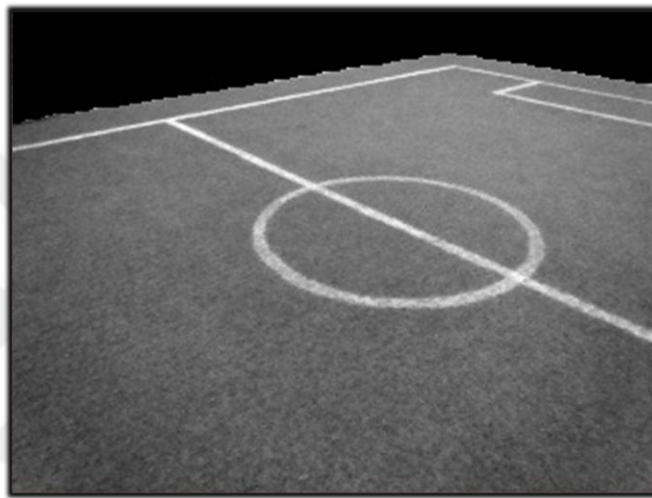


Figure 3.7. The image in Gray Scale.

The white detection threshold function returns 1 if the pixel value is between 160 and 255 and 0 if not (Equation 3.4). Certainly, it is not possible to fully extract the lines and some pixels inside the lines can be still black as seen in Figure 3.8.a. Such errors can be corrected by dilating the image, such that the white pixels in the binary image will be enhanced to their neighborhood pixels. After the image is dilated, regions with poor line definition can be corrected as white pixels and the lines in the field can be obtained more accurately as seen in Figure 3.8.b.

$$Gray' = \begin{cases} 1 & \text{if } 160 \leq Gray \leq 255 \\ 0 & \text{otherwise} \end{cases} \quad (3.4)$$

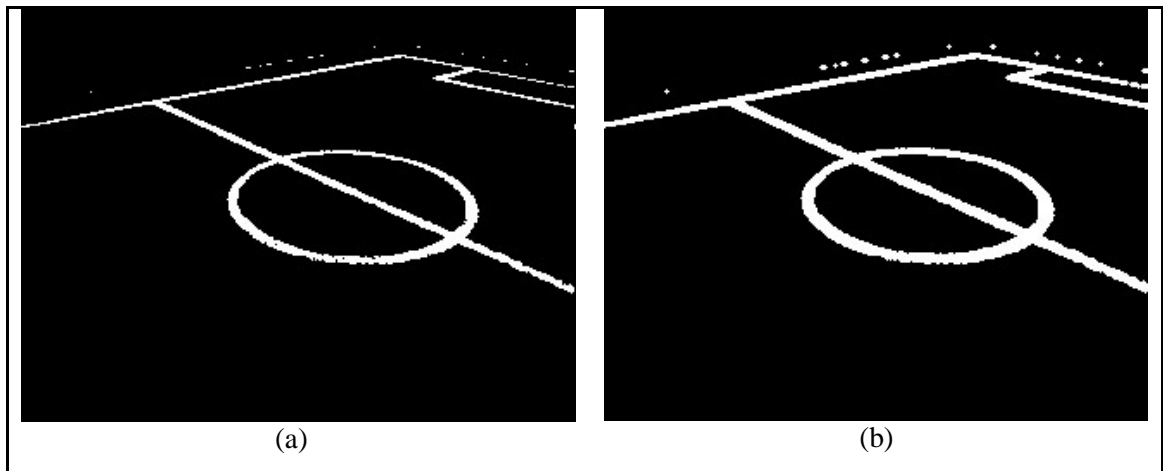


Figure 3.8. Field Line Detection images (a) The thresholded image. (b) The resulting dilated image.

3.1.3. FEATURE EXTRACTION

The images obtained by the field extraction and field line detection processes need to be combined into a single image and this would be the final image that will be used to generate the image signature. Note that the field and the lines detected by the two processes are represented with white pixels. In order to have a common representation in one image only, in the final image the lines will be represented as black in a white field.

After this step, the final image is divided into 120 identical regions to form the features of the image. The regions are obtained by dividing the image into 20x32 pixel blocks.

However, it is not possible to use the pixels in a region directly as features. Hence, each region should be reduced to a utility value that represents the characteristics of the line pieces in this region. For this purpose, the ratio of white pixels in each region is used. Note that the field lines are represented by the white pixels in the final image obtained. This ratio can change according to the distance of the robot to a line on the field and also the direction of the robot when the image was taken. When we consider the white pixel ratios of all regions in the image, we can claim that this data form a discriminative feature set for the position and the direction of the robot on the field. In order to calculate a ratio of white

pixels in a region, the pixel values in the region are simply summed (since black pixels have value 0) and then divided by the number of pixels in the region. By selecting 120 regions in the image we finally have 120 feature values which compose the signature of the image. For a given region r , the corresponding feature value r_f is calculated with the equation 3.5 where s and t are the number of pixels in the rows and columns in the region and $r(i, j)$ is the pixel value at coordinate $[i, j]$.

$$r_f = \frac{\sum_{i=0}^s \sum_{j=0}^t r(i, j)}{s * t} \quad (3.5)$$

3.2. ARTIFICIAL NEURAL NETWORK LEARNING

The method proposed in this study uses a supervised ANNs learning where a training data is utilized in order to induce a function that represents the input output relationship in the training data. Then the trained ANN is used to determine the direction and position of the robot on a set of test data. In this study, the *Resilient Backpropagation* (Rprop) algorithm is used [12]. The method is advantageous for feedforward ANNs in terms of the efficiency it provides for the weight update process.

In Rprop learning, partial derivative of the error calculated at the output layer is used to update the weight values. The partial derivative of the error is put into consideration in the weight update process. The equation 3.6 is used to determine the update amount (Δ_{ij}) where w_{ij} is the weight from neuron j to neuron i , n^+ is the increasing factor which is set as 1.2, n^- is the decreasing factor which is set as 0.5 and t is the time. Then the updated weight is calculated with equation 3.7. [13]

$$\Delta_{ij} = \begin{cases} n^+ * \Delta_{ij}(t-1) & \text{if } \frac{\partial E(t)}{\partial w_{ij}} * \frac{\partial E(t-1)}{\partial w_{ij}} > 0 \\ n^- * \Delta_{ij}(t-1) & \text{if } \frac{\partial E(t)}{\partial w_{ij}} * \frac{\partial E(t-1)}{\partial w_{ij}} < 0 \\ \Delta_{ij}(t-1) & \text{else} \end{cases} \quad (3.6)$$

$$w(t+1)_{ij} = \begin{cases} w(t)_{ij} - \Delta_{ij} & \text{if } \frac{\partial E(t)}{\partial w_{ij}} > 0 \\ w(t)_{ij} + \Delta_{ij} & \text{if } \frac{\partial E(t)}{\partial w_{ij}} < 0 \end{cases} \quad (3.7)$$

As mentioned before, this study proposes an ANN learning approach for the estimation of the direction and the position of a robot based on the visual information gathered directly by the robot vision sensors. Direction estimation and position estimation are tackled as two different problems in this study. The features of the image taken by the robot can change considerably based on the direction of the robot. In other words, the robot might be located in the same position, but the field lines that will be observed by the robot would change dramatically when the robot is facing different directions. It can be claimed that position can be correctly estimated only if the direction of the robot is known. Hence direction of the robot should be determined first and then the position estimation can be carried out afterwards.

The soccer field used in this study is symmetric. It is not possible for the robot to determine its exact direction without using an external marker. There are two goal nets in the field. A visual marker can be placed on these nets to distinguish the opponent's and robot's own net. Detection of the goal keepers is another alternative to solve the symmetry problem. In this study the symmetry problem is not handled and it is assumed that the robot is facing always to a specific goal net. What is more, in some situations the robot can be facing to a direction where the goal net cannot be observed. In such a case, it is assumed that the robot can start turning around itself until the goal net can be observed.

The training data set is created according to the above assumptions. Only one goal net is used and three different directions are determined to take the images by using the robot camera. The first group of images is taken at different positions when the robot is directly facing the goal net. The other two groups are when the robot's direction is 45° 's to the left or to the right of the goal net. These three groups of images are going to be named as 45° , 90° and 135° images considering the mid-field line as the reference line (Figure 3.9).

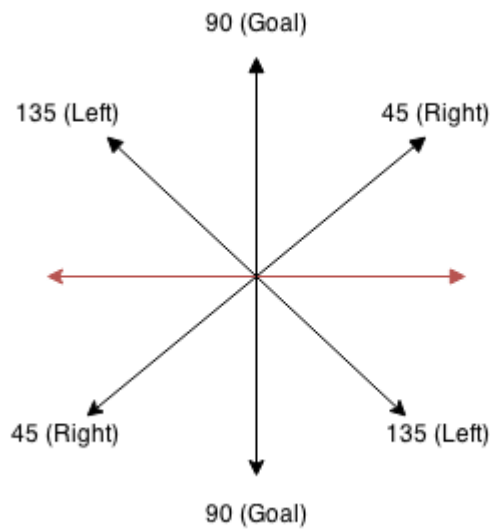


Figure 3.9. Angle assignment according to orientation.

As noted in the previous section, the class labels in the training data are set to these three different directions initially. Then, an ANN is trained on the data. Hence this initial ANN would be specialized to determine the direction of the robot. The empirical results show that a single hidden layer ANN is sufficient to classify the training data in terms of direction labels. Note that a single output neuron is sufficient where the output denotes one of the three directions for the robot.

3.2.1. POSITION ESTIMATION

For the position estimation process, 45° , 90° and 135° images are placed into separate sets and an ANN is trained on each set. Certainly the x,y coordinates of the robot are placed into the datasets as the class labels this time. In order to simplify the learning process, the field is divided into 91 equal size regions. The x,y coordinates used in the data set denote the region that the robot is in, rather than specifying the exact position of the robot. The regions are obtained by dividing the field into 7 columns and 13 rows. At the end of the training process, three ANNs are obtained where each one can provide position estimation for the robot when the robot is facing a certain direction. Hence, an ANN is obtained which can approximate the position when the robot is directly facing the goal net and the other two ANNs determine the position when the robot is facing the net with 45° 's to the left or to the right. The number of output neurons utilized this time is 2 where the output of the first

neuron would denote the x coordinate of the robot and the second one the y coordinate. The empirical results show that a single hidden layer is sufficient for this second classification task, too.

For the test process, new images are taken totally in random positions and random directions. Note that in the training phase, the pictures are taken when the robot is exactly in the middle of each region and three specific directions are used. Certainly, the same feature extraction process is carried out on the images used in the test phase. Each image is first input to the ANN that is trained for direction estimation and based on the output of this ANN, the corresponding ANN is utilized for position estimation this time.

4. IMPLEMENTATION

The proposed method is implemented with C++ and OpenCV libraries, since the official language for NAO is C++ and the recommended image processing library is OpenCV 2.4.5. A Graphical User Interface (GUI) is designed to simulate the tests in a computer environment and this GUI is implemented using QT 5.0.2. In the GUI the simulated field is displayed to the user in the right part of the screen and the results of the tests are shown on this image (Figure 4.1).



Figure 4.1. The Graphical User Interface.

The proposed methods are implemented with an object oriented approach and the system is composed to a number of classes and a class hierarchy. The classes and their relations are shown in the Figure 4.2. The user interaction with the GUI is handled by a class named 'MainWindow', responsible for generating the displays, executing the user actions and

displaying the results of those actions. When the system starts, the neural network trainings and display generations are started by a class hierarchy defined into the 'MainWindow' class. The processes that take place can be seen in Figure 4.3.

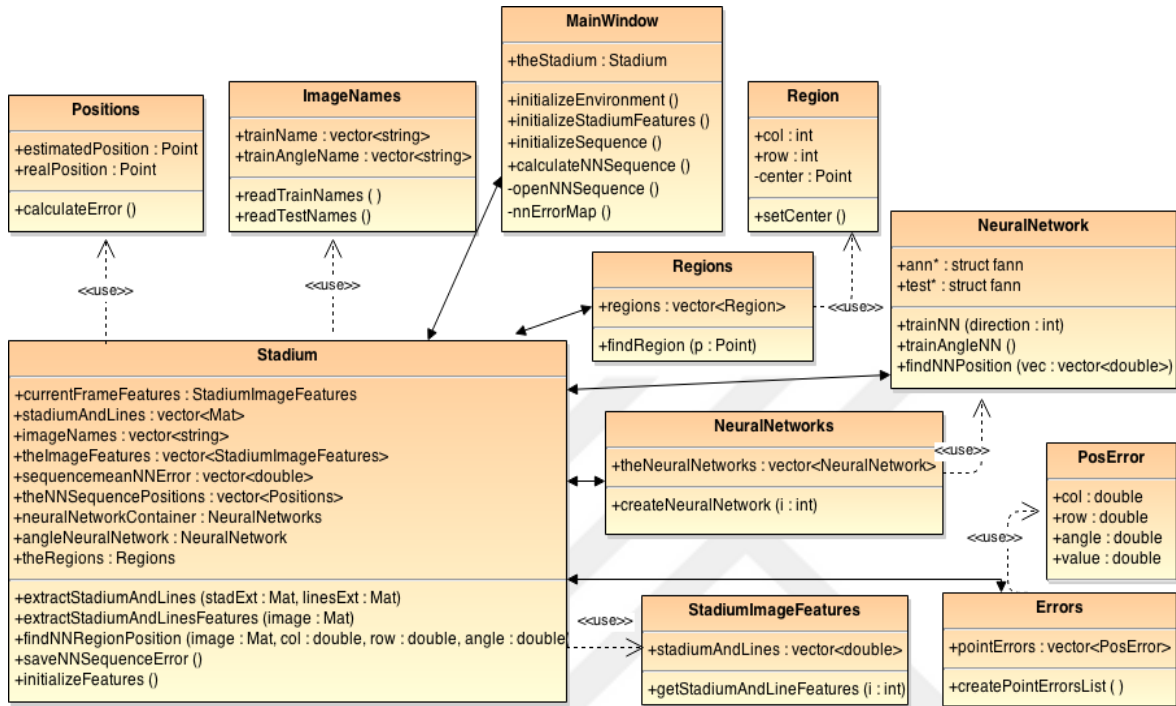


Figure 4.2. The Class Diagram.

The 'MainWindow' class starts the process by initializing an object derived from the class 'Stadium', that handles all the images and the neural network. In this process, the 'Stadium' class initiates several classes. The class 'ImageNames' which handles the conversion of image file names to column, row and angle information and holds the converted values. On the other side, the class 'StadiumImageFeatures' is the container for the signature values extracted by the 'Stadium' object from the training images. Lastly, the class 'Regions' is the container for the structure that holds the column, the row and the center point coordinates of a region which is defined by the class 'Region'.

First the 'Stadium' class generates the 'Region' class objects which are predefined as the 91 regions of the field. These 'Region' objects are then put inside a 'Regions' object to be contained as a whole. The signatures of the images that are in the 'ImageNames' object are generated by the 'Stadium' class and these values are stored into the container 'StadiumImageFeatures' object. By using these values, the 'Stadium' class creates an

orientation estimation and three position estimations with the '*NeuralNetwork*' class (derived from the Neural Network library Fast Artificial Neural Network - *FANN*) which is responsible for creating and training a neural network and estimating the direction or the position of a robot. Finally we have 4 neural networks, one for angle estimation and three for position estimation. With these ANNs (Artificial Neural Networks) at hand, the '*Stadium*' class creates a container with the '*NeuralNetworks*' class holding the four neural networks. Then the regions are sent to the '*MainWindow*' class to be displayed to the user.

When the user initiates test sequence, the image names are converted to column, row and angle values by the '*ImageNames*' object and the signatures for these images are calculated and stored in the '*StadiumImageFeatures*' object. Then each image first is tested through the Artificial Neural Network (ANN) trained for detecting the direction and depending on the answer, the appropriate ANN corresponding to the resulted orientation is activated. The real position and the estimated position is stored in the '*Positions*' class object which is responsible for storing these values and calculating the error between them. The errors of the images are stored in the '*Errors*' class which holds these values in a '*posError*' structure which holds column, row, angle and error of the image. Then the '*Stadium*' class object sends the Errors object to the '*MainWindow*' object to be displayed to the user on the simulated field image.

When a user choose the '*ErrorMap*' dialog menu to be displayed for a better understanding the '*MainWindow*' class object accesses the values in the '*Errors*' object and calculates the color codes for the errors and displays them in the simulated field.

In the implementation process several configurations for the proposed method has been tested for minimum error in the neural network training. The neural network training has been tried for hidden layers, regions and outputs with both less and more numbers. The proposed number of hidden layers, regions and outputs were the most successful ones with less neural network errors.

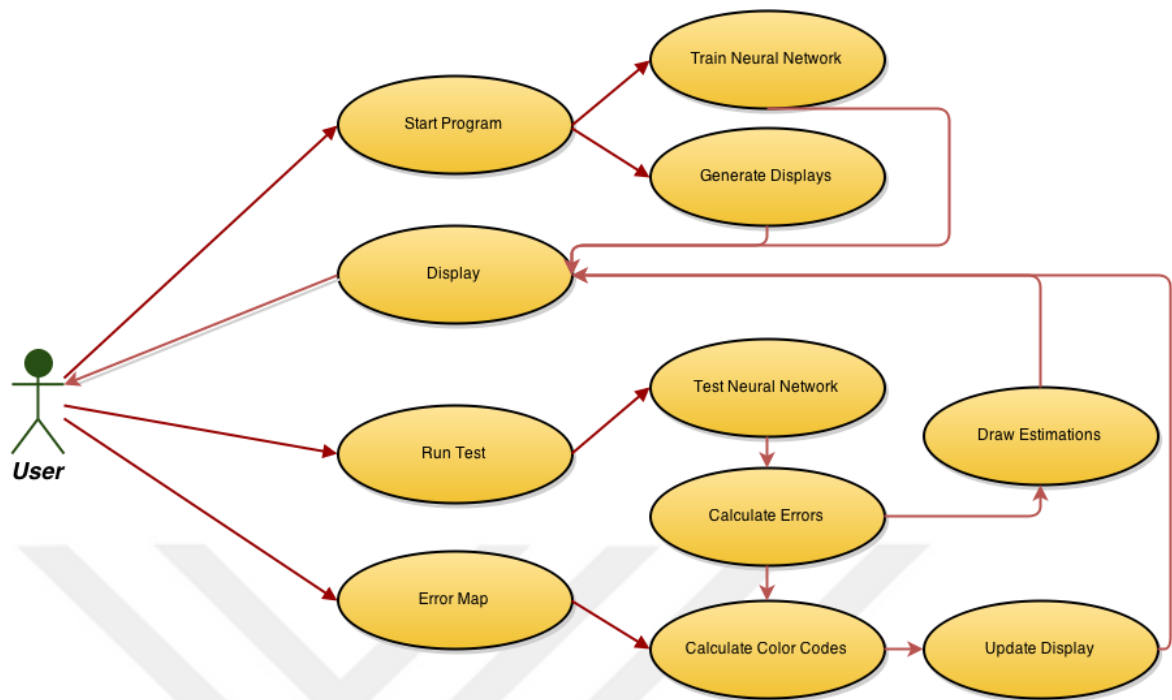


Figure 4.3. TheUse Case Diagram.

5. TEST AND EVALUATION

NAO is a humanoid robot manufactured by Aldebaran Robotics. It has been chosen as the official robot for RoboCup Standard Platform League in 2007. Its specifications are presented in Table 5.1.

Table 5.1. NAO Specifics

NAO V4	
Height	58 centimeters (23in)
Wight	4.3 kilograms (9.5 lb.)
Autonomy	60 minutes (active use), 90 minutes (normal use)
CPU	Intel Atom @ 1.6 GHz
Built-in OS	Linux
Compatible OS	Windows, Mac OS, Linux
Programing Languages	C++, Python, Java, MATLAB, Urbi, C, .Net
Connectivity	Ethernet, Wi-Fi

The test environment that has been constructed is a scaled version of the regulated football field used for the RoboCup. Consequently, the football field utilized in this study has a length of 3.75 meters and a width of 2.65 meters. The white lines have a length of 3.05 meters and a width of 2.05 meters. (Figure 5.1)



Figure 5.1. The football field.

The field is divided into 7 columns and 13 rows. Hence, 91 equal size regions are obtained on the field. The images are taken in the middle of these 91 different regions in order to

provide coverage of the field for the learning process. In each region, the images are taken in using 3 different directions (45° , 90° , 135° degrees to the middle line). (Figure 5.2)

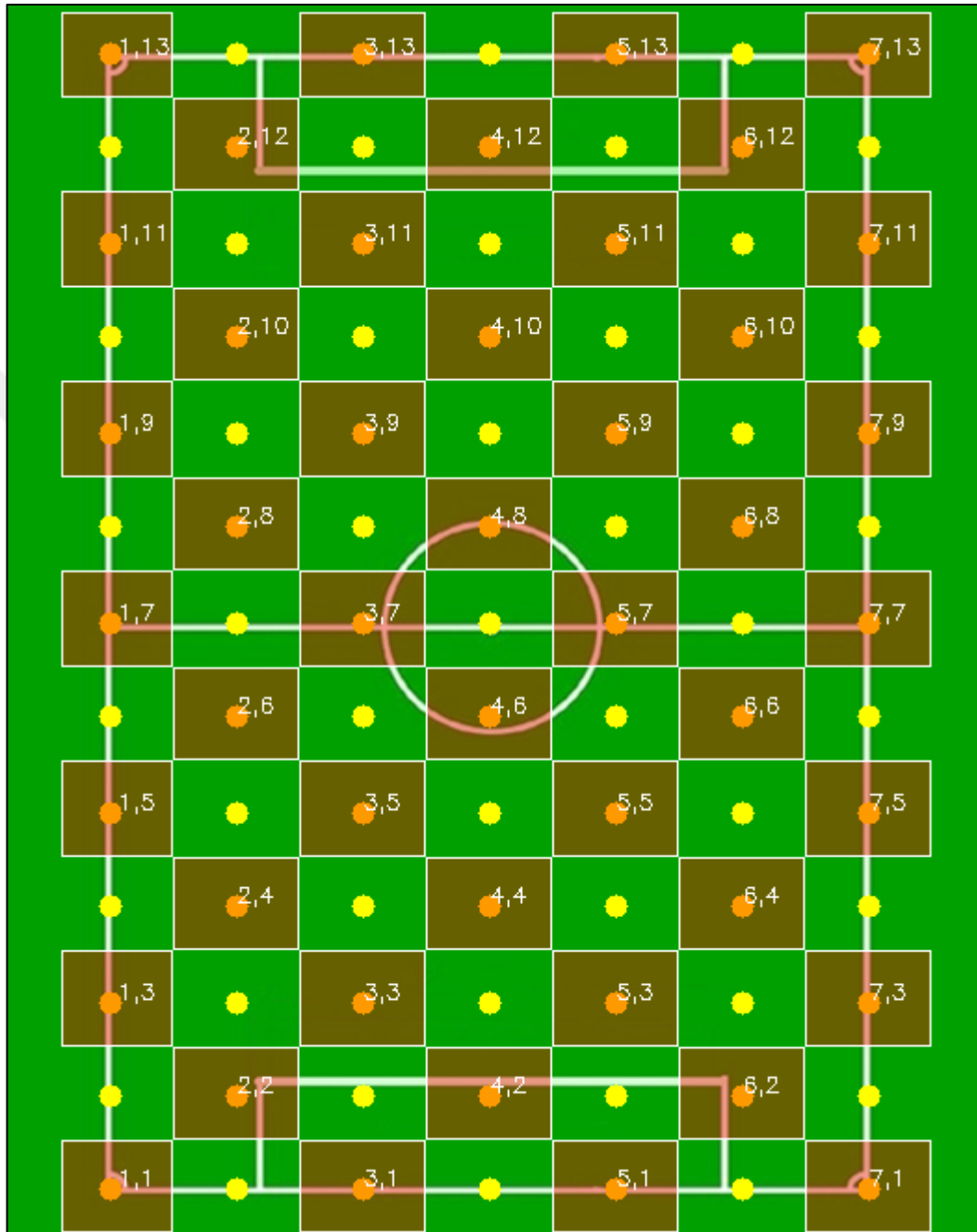


Figure 5.2. The regions and the middle points.

A set of test images are utilized to evaluate the system. Note that the images for the training set are taken in the middle of each region. The test set is formed by taking images at random positions in the regions. The robot is facing the goal post in test images,

too. However, the direction of the robot has a random angle relative to the goal post in the test set. Note that three specific angles were utilized for the direction of the robot in the training set. Four different test scenarios have been prepared to evaluate the system. For each case, a set of images are chosen randomly from the set of test images. However, it is guaranteed that each case consists of at least one image for each region in the field.

The first test case includes images where the robot is facing directly the goal post. The next two sets consist of images where the robot is facing to the left and to the right of the goal post. Hence, these three sets are used to measure the performance of the three ANNs trained to estimate the position of the robot given the correct direction for the robot. Then these three sets are combined into a single set to measure the overall performance of the system. In this fourth test scenario, certainly the image is first input to the ANN trained for direction estimation and then the output of this initial ANN determines the ANN that will be used for position estimation. For a better understanding of the results, the field is divided into three layers: Top, Middle and Bottom. Top Layer consists of the first four rows in the field, Middle Layer refers to the middle five rows and Bottom Layer is the last four rows in the field (Figure 5.3).

Note that the ANN output is the xy-coordinates of a region. It is assumed that the ANN estimation for the robot position is the middle point of the region. The coordinates of this middle point in centimeters are used in error calculation. The error of estimation is defined as the Euclidean distance between the middle point coordinates of the estimated region and the real position of the robot. In equation 5.1, x, y are the actual coordinates of the robot and x', y' are the middle point coordinates of the estimated region. The average error obtained for each region is represented on the field by different colors. The color of the error is determined by the ranges presented in Table 5.2.

$$error = \sqrt{(x - x')^2 + (y - y')^2} \quad (5.1)$$

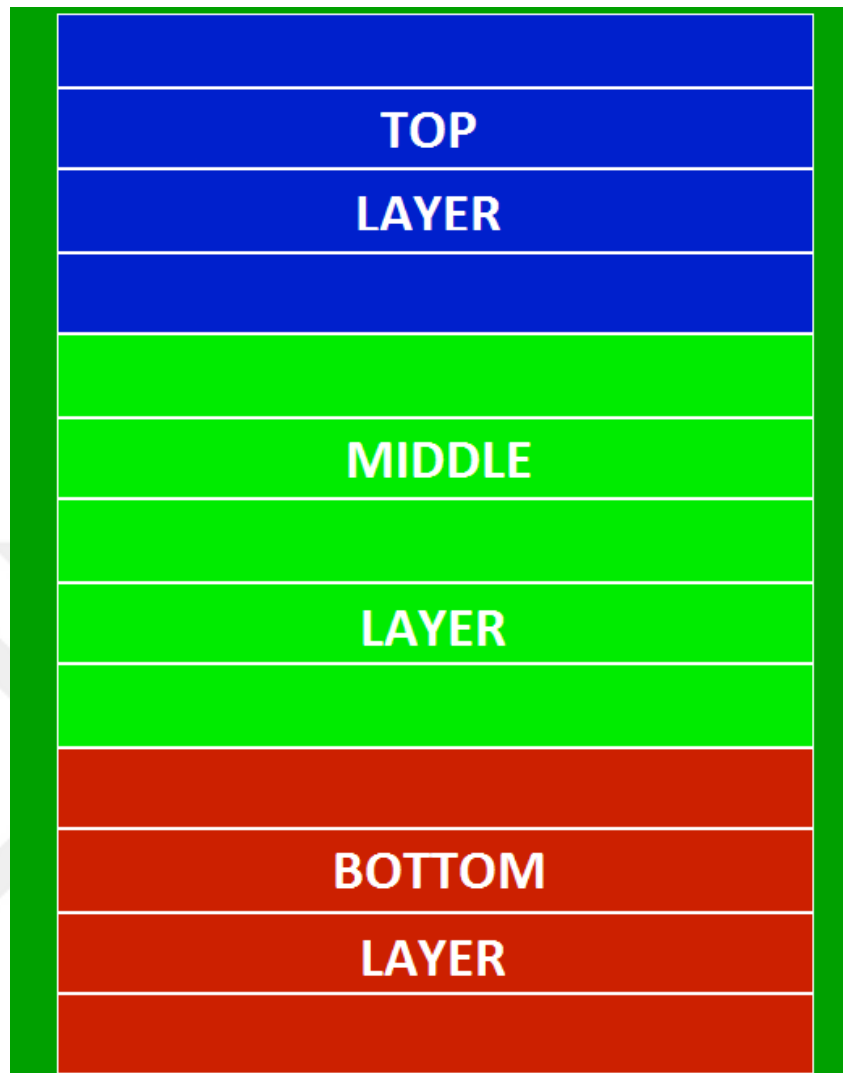


Figure 5.3. The layer labels of regions.

Table 5.2. Error Color Codes

$0 \leq error \leq 25cm$	GREEN
$25cm < error \leq 50cm$	YELLOW
$50cm < error$	RED

In the beginning of the testing process 273 images are used to train the proposed neural networks. A Feature extraction process is applied on these images for the training. The signatures of these images are obtained in 10.59 seconds deducing that the signature generation for a single image takes 0.038 seconds. The orientation estimation neural network is trained in 87 seconds by using these 273 signatures. The average time for the training of the neural networks dealing with position estimation is 25.98 seconds. In total, the training process takes 175.53 seconds.

In order to have a robust system, the training process is carryout for once. Then, the signatures of the training images and the trained neural networks are written to specific files. When the system is restarted, 0.11 seconds is needed for reading all the signatures from the existing files. Loading of the neural network models take 0.01 seconds for each and the total process takes 0.15 seconds.

In the testing phase, an image is processed in 0.038 seconds. This is the time to generate the signature for a single image and it is a fixed measure because of the fixed image size of the robots camera. Finding the position from the signature with the help of the neural networks takes less than 1 millisecond. As a result, it can be said that the system could be used in real time because robots approximate time to take a step in an environment is 0.51 seconds.

In Figure 5.4, the results of the first test are presented. Note that, the images taken when the robot is directly facing the goal post are utilized with the corresponding ANN in this test. In the figure the average error in centimeters is presented for each region. The results denote that when the robot is facing the goal post, it can estimate its position almost perfectly. In the Top Layer, the estimation gives the correct region coordinates for %52 of the images. This percentage goes up to %71 and %46 in Middle Layer and Bottom Layer. When the correct region cannot be determined, the estimation is one of the neighborhood regions in %32, %12 and %29 of the cases in the three layers. Hence, the correct region or a neighborhood region is the result of the estimation process for %75 of the images when the robot is directly facing the goal post. We can support that since the robot is facing the goal post there are no disturbances in the images because of the fact the robot sees the field clearly. In that case, the estimations give very accurate results.

In the following two tests where the robot is facing the goal post with a certain angle, the success rate decreases. This is because in some images only a small portion of the field exists due to the angle utilized when the image is taken. What is more, the orientation of the robot changes the line configuration that exists in the image considerably.

33.22	3.45	5.39	224.29	103.1	101.1	232.09
67.2	105.14	12.15	48.42	39.38	67.2	17.43
33.01	37.01	13.37	6.73	26.22	33.01	107
96.92	3.45	38.46	19.83	25.99	55.62	11.63
67.2	9.42	6.9	19.17	5.43	33.32	7.23
8.95	3.03	10.7	12.07	6.21	8.95	5.03
67.27	8.84	15.92	13.88	6.92	21.47	96.07
72.74	40.31	28.23	16.71	16.85	68.75	6.06
81.18	14.6	12.34	44.2	7.41	11.88	3.28
12.65	6.75	47.14	81.38	66.59	71.47	12.21
67.09	22.8	31.46	18.16	18.08	33.14	5.69
69.37	7.24	15.97	23.18	36.63	6.9	5.28
80.19	33.18	26.04	27.39	1.68	38.57	56.84

Figure 5.4. Test results of robot facing the goal post.

In Figure 5.5, the results of these two tests are presented. In the test where the robot is facing to left of the goal post, the correct region estimation percentages of Top, Middle and Bottom Layers are %64, %54 and %54 respectively. These results are %40, %46 and %46 for the test in which the robot is facing to right of the goalpost. When we sum the correct and neighborhood region estimation percentages, it is %79, %83 and %64 for the left facing test and %68, %57 and %50 for right facing test. (Figure 5.5) In the test where

the robot is facing right of the goal post, the errors increase because of the disturbances in the images caused by the shadows and the lighting conditions of the environment. These disturbances are the results of the shadows cast on the field by the location of the windows in the room where the field is located. This is not the issue for the other test since the lighting is not affecting the result when the robot is facing the left of the goal post and the windows in the room does not cast a shadow on the left of the field.



Figure 5.5. Test results of robot facing left and right a) Left b) Right

In the final test scenario, all of the images used in the first three test cases are combined into a single test set. In this case, first the ANN trained for direction estimation is used to determine the direction of the robot. Based on the result of this first ANN, the corresponding ANN is utilized to estimate the position of the robot on the field. The percentage of estimations for the correct and neighborhood regions is %67 percent in this test. The average error rate for each region can be seen in Figure 5.6.

For the four test scenarios presented, the mean error is also calculated for each test case. The error is calculated according to the equation 5.2 and the results are displayed in Table 4.3.

$$mean = \frac{\sum error}{\#ofimages} \quad (5.2)$$

In Table 5.3, it can be seen that the robot can estimate its position with an average error of 42cm in general. The total standard deviation is 27cm and it shows that the calculated error is capable of representing the distribution of the errors in the whole field. When the robot is facing the goal post the mean error decreases to 28cm. In the training data set, note that the images are taken only using three specific directions. However, the directions are random in the test set. It is thought that the success rate could be increased if more directions are utilized in the training set, too. However, it can be claimed that the error that occurs with the current position estimation process is still in an acceptable range.

Table 5.3. Mean Errors

	Left	Goal	Right	All
Bottom	54.35cm	32.97cm	62.47cm	41.57cm
Middle	43.22cm	2.81cm	57.66cm	41.47cm
Top	31.31cm	55.99cm	41.88cm	42.77cm
All	42.98cm	28.45cm	54.28cm	41.9cm

The results of the related works described before and the results of the proposed method, the field dimensions and the method's significance is displayed in the Table 5.4. The first approach [3] is using an omnidirectional camera which can see the whole field in any position the robot is in. The second approach [2] only considers the half of the field and there are landmarks in the scene that the positions of them are known by the robot.

36.61	4.91	9.5	16.21	59.81	99.46	65.37
37.83	77.62	58.86	55.48	67.13	81.08	68.08
31.66	24.97	20.73	23.87	25	17.69	42.24
116.69	13.63	20.36	22.5	22.29	50.09	27.9
110.93	20.71	14.64	20.36	12.15	31.14	41.98
92.27	3.94	24.7	19.88	12.41	33.81	54.98
53	17.13	38.16	22.19	11.22	49.69	77.53
31.01	12.39	66.24	59.1	51.23	83.85	77.56
32.04	20.79	39.73	49.98	49.49	11.89	12.17
11.44	55.29	60.85	39.06	29.88	65.85	69.2
153.66	18.6	28.61	48.31	23.54	56.09	24.32
75.58	8.16	63.83	36.4	43.39	8.69	6.94
73.62	39.53	46.1	14.37	6.13	39.83	21.74

Figure 5.6. Test results for all obtained images.

When the results of these approaches are compared, the proposed approach does not seem to be superior because the other approaches tackle a simplified version of the problem. They either must see the whole field from above or the need to detect landmarks in order to calculate the position of the robot. The proposed approach considers the whole field, uses a single camera located in front of its head, does not use a known landmark and it utilizes a learning based process for self-localization.

Table 5.4. Comparison of Approaches.

	Error	Field Dimensions	Region Size	Significance
Pinto et al. [3]	7.7 cm	9x6 m ²	2m	Omnidirectional Camera
Chang et al. [2]	6.8cm	3.9x5.1 m ² (Only half of the field)	0.3m	Known Landmarks
Proposed Method	42 cm	3.75x2.65 m ²	0.25m	Human-like Process

Another test has been conducted when another robot is present in the scene and the robot can see at least some parts of this robot in its view. At least one image for every region. However less number of regions are used in this test. The outermost regions in Figure 5.6 are excluded from this test since the other robot would not be in the scene in most of the cases. The images obtained for this test goes through the same process and the signatures are generated for each image. When the image is processed, the other robot is also deleted from the image by the techniques used. The results show that even with a flawed image the system gives acceptable results with a 39.53 cm error and a 24.82cm standard deviation. (Figure 5.7)

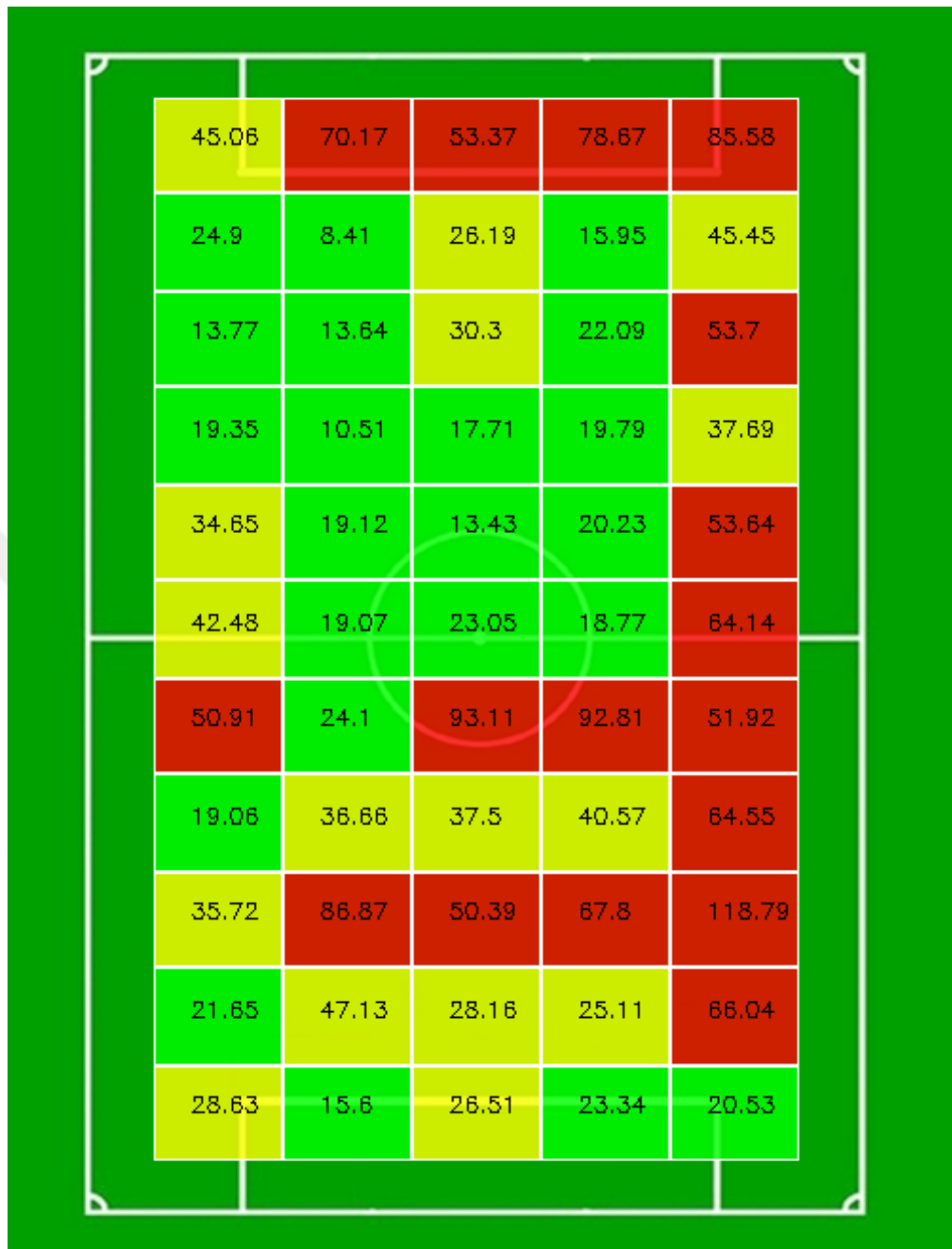


Figure 5.7. Test results when another robot is present in the scene.

6. CONCLUSION

In this study, a vision sensor based solution is proposed for the self-localization problem of humanoid robot NAO on a soccer field. A learning process with Artificial Neural Networks (ANNs) is applied on images taken from the camera of the robot. A training set is formed by taking images in specific positions on the field. The images are represented with a set of features which are extracted by using a set of image processing techniques. This extraction process determines the ratio of white lines in different regions of the image and these features are utilized by a supervised learning algorithm.

The data set formed is used for training several ANNs. The first ANN is responsible for determining the direction of the robot and the other three ANNs are used to determine the position of the robot on the field. A second set of images are used for the evaluation process. The images in this set are taken at random positions on the field by using random directions. The average error of position estimation is 41.9 cm in the empty field and 39.53 cm in the field with another robot present. It can be concluded the estimation process produces satisfactory results for the self-localization of the robot. The robot can have quite accurate estimations for the position and direction of itself in most of the cases. A scaled version of regulated football field of RoboCup is used in the tests. However, the method is applicable to the actual RoboCup field, too.

In this study, the features are extracted manually by using image processing techniques. However, auto encoders can be used to extract the features as the feature work. Autoencoders provide a method where the signatures for the images can be generated automatically without the need of a preprocessing for the images. It would be interesting to compare the current results with the output that would be obtained with the features extracted with autoencoders.

REFERENCES

1. J. Borenstein, H. R. Everett, L. Feng and D. Wehe, Mobile robot positioning-sensors and techniques. *Naval Command Control and Ocean Surveillance Center RDT and E Div.* San Diego CA.(1997).
2. S. H. Chang, W. H. Chang, C. H. Hsia, F. Ye and J. S. Chiang, Efficient neural network approach of self-localization for humanoid robot. *Pervasive Computing (JCPC), 2009 Joint Conferences. IEEE, 2009.*
3. M. Pinto, H. Sobreira, A. Paulo Moreira, H. Mendonça and A. Matos, Self-localisation of indoor mobile robots using multi-hypotheses and a matching algorithm. *Mechatronics, 23(6), 727-737, 2013.*
4. E. Frontoni and P. Zingaretti, An efficient similarity metric for omnidirectional vision sensors. *Robotics and Autonomous Systems, 54(9), 750-757, 2006.*
5. J. Minguez, L. Montesano and F. Lamiroux, Metric-based iterative closest point scan matching for sensor displacement estimation. *Robotics, IEEE Transactions on, 22(5), 1047-1054, 2006.*
6. H. Fujii, Y. Ohde, M. Kato, F. Otsuka, N. Sema, M. Kawashima, E. Sugiyama, S. Niizuma and K. Yosida, EIGEN team description, *In Robocup-2004, 2004.*
7. S. Thrun, Probabilistic robotics. *Communications of the ACM, 45(3), 52-57, 2002.*
8. P. S. Heckbert, editor. *Graphics gems IV (Vol. 4).* Morgan Kaufmann Publishers, 1994.
9. E. Frontoni, A. Mancini and P. Zingaretti, Vision based approach for active selection of robot's localization action. *In Proceedings of the 15th Mediterranean Conference on Control & Automation, Athens, Greece, 2007.*

10. P. Zingaretti and E. Frontoni, Vision and sonar sensor fusion for mobile robot localization in aliased environments. *In Mechatronic and Embedded Systems and Applications, Proceedings of the 2nd IEEE/ASME International Conference on* (pp. 1-6). IEEE, 2006.
11. G. Reina, A. Vargas, K. Nagatani and K. Yoshida, Adaptive kalman filtering for gps-based mobile robot localization. *In Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop* (pp. 1-6). IEEE, 2007.
12. M. Riedmiller and H. Braun, RPROP-A fast adaptive learning algorithm. *In Proceedings of ISICIS VII, Universitat*, 1992.
13. C. Igeland M. Hüsken, Improving the Rprop learning algorithm. *In Proceedings of the second international ICSC symposium on neural computation (NC 2000) (Vol. 2000, pp. 115-121). Genova: ICSC Academic Press*, 2000.
14. C. R. Giardina and E. R. Dougherty, Morphological methods in image and signal processing. *Engelwood Cliffs: Prentice Hall, 1*, 1988.
15. J. J. Leonard and H. F. Durrant-Whyte, Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions*, 7(3), 376-382, 1991.
16. M. Betke and L. Gurvits, Mobile robot localization using landmarks. *Robotics and Automation, IEEE Transactions*, 13(2), 251-26, 1997.
17. C. C. Wang, C. Thorpe and S. Thrun, Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. *In Robotics and Automation, Proceedings of ICRA'03. IEEE International Conference on*, Vol. 1, 842-849, 2003.
18. D. Hahnel, W. Burgard, D. Fox and S. Thrun, An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range

measurements. *Intelligent Robots and Systems, Proceedings of IEEE/RSJ International Conference on Vol.1, 206-211, 2003.*

19. L. Montesano, J. Minguéz and L. Montano, Modeling the static and the dynamic parts of the environment to improve sensor-based navigation. *In Robotics and Automation, ICRA 2005. Proceedings of the 2005 IEEE International Conference on, 4556-4562, IEEE, 2005.*
20. P. J. Besl and N. D. McKay, Method for registration of 3-D shapes. *In Robotics-DL tentative. International Society for Optics and Photonics, 586-606, 1992.*
21. G. H. Joblove and D. Greenberg, Color spaces for computer graphics. *In ACM siggraph computer graphics, Vol. 12, No. 3, 20-25, 1978.*
22. J. F. Hughes and J. D. Foley. *Computer graphics: principles and practice.* Pearson Education, 2013.
23. P. Auer, H. Burgsteiner and W. Maass, A learning rule for very simple universal approximators consisting of a single layer of perceptrons. *Neural Networks 21.5, 786-795, 2008.*
24. P. Werbos, *Beyond regression: New tools for prediction and analysis in the behavioral sciences.* 1974.