A project report on

ROAD SIGNS DETECTION USING OpenCV AND HAARCASCADE

submitted in partial fulfillment of the requirements for the degree of

B. Tech
In
Electronics and Computer Science Engineering

By

ADIP RANJAN DAS	**1630033**
SEPHALI KUMARI	**1630023**
HARSHIT SINGH	**1630057**
ADRIJA DEY	**1630035**
ADITYA PODDAR	**1630034**
HRITTICK KUNDU	**1630059**

under the guidance of

Prof. RISHI KUMAR KHANNA

School of Electronics Engineering

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

(Deemed to be University)

BHUBANESWAR

MAY 2019

CERTIFICATE

This is to certify that the project report entitled **"ROAD SIGNS AND LANE DETECTION USING OpenV AND HAARCASCADE"** submitted by

ADIP RANJAN DAS	**1630033**
SEPHALI KUMARI	**1630023**
HARSHIT SINGH	**1630057**
ADRIJA DEY	**1630035**
ADITYA PODDAR	**1630034**
HRITTICK KUNDU	**1630059**

in partial fulfilment of the requirements for the award of the **Degree of Bachelor of Technology** in **Electronics and Computer Science Engineering** is a Bonafede record of the work carried out under my(our) guidance and supervision at School of Electronics Engineering, KIIT (Deemed to be University).

Signature of Supervisor 1

Prof. RISHI KUMAR KHANNA

School of Electronics Engineering

The Project was evaluated by us on _____

ACKNOWLEDGEMENTS

We feel immense pleasure and feel privileged in expressing our deepest and most sincere gratitude to our supervisor **Professor Rishi Kumar Khanna**, for his excellent guidance throughout our project work. His kindness, dedication, hard work and attention to detail have been a great inspiration to us. Our heartfelt thanks to you sir for the unlimited support and patience shown to us. We would particularly like to thank him for all his help in patiently and carefully correcting all our manuscripts.

We are also very thankful to **Professor Ghanashyam Rout** B.tech project coordinator (E&TC), Associate Dean Professor **Dr. Amlan Datta** and **Professor Dr. Arun Kumar Ray**, Dean (School Of Electronics) for their support and suggestions during our course of the project work in the final year of our undergraduate course.

ADIP RANJAN DAS (1630033)
ADITYA PODDAR (1630034)
ADRIJA DEY (1630035)
SEPHALI KUMARI (1630023)
HARSHIT SINGH (1630057)
HRITTICK KUNDU (1630059)

ABSTRACT

ROAD SIGNS DETECTION USING IMAGE PROCESSING LIBRARY

(OpenCV AND HAARCASCADE).

The aim of the project is to detect and recognize traffic signs provided in image sequences. The purpose of our Project is to explore the field of Machine Vision as a Fundamental Sensor technology in Robotics. Traffic Sign Recognition (TSR) is used to regulate traffic signs, warn a driver, and command or prohibit certain actions. A fast real-time and robust automatic traffic sign detection and recognition can support and disburden the driver and significantly increase driving safety and comfort. Automatic recognition of traffic signs is also important for automated intelligent driving vehicle or driver assistance systems. This paper presents a study to recognize traffic sign patterns using OpenCV and Haarcascade. The images are extracted, detected and recognized by pre-processing with several image processing techniques. Then, the stages are performed to detect and recognize the traffic sign patterns. The experimental results show the accurate classifications of traffic sign patterns with complex background images and the computational cost of the proposed method.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 BACKGROUND

Machine vision (other names include computer vision and artificial vision is an important sensor technology with potential applications in many industrialOperations. Many of the current applications of machine vision are in inspection;however, it is anticipated that vision technology will play an increasingly significant role in the future of robotics.

Vision systems designed to be utilized with robot or manufacturing systems must meet two important criteria which currently limit the influx of vision systems to the manufacturing community. The first of these criteria is the need for relatively low-cost vision system. The second criterion is the need for relatively rapid response time needed for robot or manufacturing applications, which should be typically a fraction of a second.

Nevertheless, there has been a significant influx of vision systems into the manufacturing world. The systems are used to perform tasks which include selecting parts that are randomly oriented from a bin or conveyer, parts identification, and limited inspection. These capabilities are selectively used in traditional applications to reduce the cost of part and tool fixturing, and to allow the robot program to test for and adapt to limited variations in the environment. Advances in vision technology for robotics are expected to broaden the capabilities of robotic vision systems to allow for vision-based guidance of the robot arm, complex inspection for close dimensional tolerances, and improved recognition and part location capabilities. These will result from the constantly reducing cost of computational capability, increased speed, and new and better algorithms currently being developed.

1.2 ORGANIZATION OF THE REPORT

The software used to complete the project on recognition of road signs:

1. Python 3
2. MATLAB 2019

Libraries used:

1. OpenCV:

   OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

2. Numpy:

   NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

Chapter 2

INTRODUCTION TO MACHINE VISION

Machine vision is concerned with the sensing of vision data and its interpretation by computer. The typical vision system consists of the camera and digitizing hardware,digital computer, and hardware and software necessary to interface them. This interface hardware and software is often referred to as a preprocessor. The operation of the vision system consists of three functions:

1.Sensing and digitizing image data
2. Image processing and analysis
3. Application

The sensing and digitizing functions involve the input of vision data by means of a camera focused on the scene of interest. Special lighting techniques are frequently used to obtain an image of sufficient contrast for later processing. The image viewed by the camera is typically digitized and stored in computer memory. The digital image is called a frame of vision data, and is frequently captured by a hardware device called a frame grabber. These devices are capable of digitizing images at the rate of 30 frames per second. The frames consist of a matrix of data representing projections of the scene sensed by the camera. The elements of the matrix are called picture elements, or pixels. The number of pixels are determined by a sampling process per formed on each image frame. A single pixel is the projection of a small portion of the scene which reduces that portion to a single value. The value is the measure of the light intensity for that element of the scene. Each pixel intensity is converted into a digital value. (We are ignoring the additional complexities involved in the operation of a color video camera).

The digitized image matrix for each frame is stored and then subjected to image processing and analysis functions for data reduction and interpretation of the image. These steps are required in order to permit the real-time application of vision analysis required in robotic applications. Typically, an image frame will be thresholded to produce a binary image, and then various feature measurements will further reduce the data representation of the image. This data reduction can change the representation of a frame from several hundred thousand bytes of raw image data to several hundred bytes of feature value data. The resultant feature data can be analyzed in the available time for action by the robot system.

Various techniques to compute the feature values can be programmed into the computer to obtain feature descriptors of the image which are matched against previously computed values stored in the computer. These descriptors include

shape and size characteristics that can be readily calculated from the thresholded image matrix.

To accomplish image processing and analysis, the vision system frequently must be trained In training, information is obtained on prototype objects and stored ascomputer models. The information gathered during lists of features such as the area of the object, its perimeter length, major and minor diameters, and similar features. During subsequent operation of the system, feature values computed on unknown objects viewed by the camera are compared with the computer models to determine if a match has occurred,The third function of a machine vision system is the applications function.

The current applications of machine vision in robotics include inspection, part identification, location, and orientation. Research is ongoing in advanced applications of machine vision for use in complex inspection, guidance, and navigation.

Vision systems can be classified in a number of ways. One obvious classification is whether the system deals with a two-dimensional or three-dimensional model of the scene. Some vision applications require only a two-dimensional analysis. Examples of two-dimensional vision problems include checking the dimensions of a part of verifying the presence of components on a subassembly. Many two-dimensional vision systems can operate on a binary image which is the result of a simple thresholding technique. This is based on an assumed high contrast between the objects) and the background. The desired contrast can often be accomplished by using a controlled lighting system Three-dimensional vision systems may require special lighting techniques and more sophisticated image processing algorithms to analyze the image. Some systems require two cameras in order to achieve a stereoscopic view of the scene, while the other three-dimensional systems rely on the use of structured light and optical triangulation techniques with a single camera. An example of a structured light system is one that projects a controlled band of light across the object. The light band is distorted according to the three-dimensional shape of the object. The vision system sees the distorted band and utilizes triangulation to deduce the shape.

Another way of classifying vision systems is according to the number of gray levels (light intensity levels) used to characterize the image. In a binary image the gray level values are divided into either of two categories, black or white. Other systems permit the classification of each pixel's gray level into various levels, the range of which is called a gray scale.

Chapter 3

SENSING AND DIGITIZING FUNCTION IN MACHINE VISION

Our description of the typical machine vision system in the preceding section identified three functions: sensing and digitizing, image processing and analysis, and application. This and the following sections in the chapter will elabora functions. The present section is concerned with the sensing and digitizing aspects of machine vision.

Image sensing requires some type of image formation device such as a camera and a digitizer which stores a video frame in the computer memory. We divide the sensing and digitizing functions into several steps. The initial step involves capturing the image of the scene with the vision camera. The image consists of relative light intensities corresponding to the various portions of the scene. These light intensities are continuous analog values which must be sampled and converted into a digital form.

The second step, digitizing, is achieved by an analog-to-digital (A/D) converter. Converter is either a part of a digital video camera or the front end of a frame grabber. The choice is dependent on the type of hardware in the system. The frame grabber, representing the third step, is an image storage and computation device which stores a given pixel array. The frame grabber can vary in capability from one which simply stores an image to significant computation capability. In the more powerful frame grabbers, thresholding, windowing, and histogram modification calculations can be carried out under computer control. The stored image is then subsequently processed and analyzed by the combination of the frame grabber and the vision controller.

Chapter 4

IMAGING DEVICES

There are a variety of commercial imaging devices available. Camera technologies available include the older black-and-white vidicon camera, and the newer, second generation, solid state cameras. Solid state cameras used for robot vision include charge-coupled devices (CCD), charge injection devices (CID), and silicon bipolar sensor cameras. For our purposes, we review two such devices in this subsection, the vidicon camera and the charge-coupled device (CCD).

In the operation of this system, the lens forms an image on the glass faceplate of the camera. The faceplate has an inner surface which is coated with two layers of material. The first layer consists of a transparent signal electrode film deposited on the faceplate of the inner surface. The second layer is a thin photosensitive material deposited over the conducting film. The photosensitive layer consists of a high density of small areas. These areas are similar to the pixels mentioned previously. Each area generates a decreasing electrical resistance in response to increasing illumination. A charge is created in each small area upon illumination. An electrical charge pattern is thus generated corresponding to the image formed on the faceplate. The charge accumulated for an area is a function of the intensity of impinging light over a specified time. Once a light sensitive charge is built up, this charge is read out to produce a video signal. This is accomplished by scanning the photosensitive layer by an electron beam. The scanning is controlled by a deflection coil mounted along the length of the tube. For an accumulated positive charge the electron beam deposits enough electrons to neutralize the charge. An equal number of electrons flow to cause a current at the video signal electrode. The magnitude of the signal is proportional to the light intensity and the amount of time with which an area is scanned. The current is then directed through a load resistor which develops a signal voltage which is

further amplified and analyzed. Raster scanning eliminates the need to consider the time at each area by making the scan time the same for all areas. Only the intensity of the impinging light is considered. In the United States, the entire faceplate is scanned approximately 30 frames per second. The European standard is 25 frames per second. Raster scanning is typically done by scanning the electron beam from left to right and top to bottom. The process is such that the system is designed to start the integration with zero-accumulated charge. For the fixed scan time, the charge accumulated is proportional to the intensity of that portion of the image being considered. The output of the camera is a continuous voltage signal for each line

scanned. The voltage signal for each scan line is subsequently sampled and quantized resulting in a series of sampled voltages being stored in digital memory. This analog to-digital conversion process for the complete screen (horizontal and vertical) results in a two-dimensional array of picture elements (pixels). Typically, a single pixel is quantized to between six and eight bits by the A/D converter.

Another approach to obtaining a digitized image is by use of the charge coupled device (CCD). In this technology, the image is projected by a video camera CCD which detects, stores, and reads out the accumulated charge generated by the light on each portion of the image. Light detection occurs through of light on a photoconductive substrate (e.g., silicon). Charges accumulate under positive control electrodes in isolated wells que lo voltages applied to the central electrodes. Each isolated well represents one pixel and can be transferred to output

storage registers by varying the voltages on the metal control electrodes. Charges are accumulated for the time it takes to complete a single image after which they are transferred line by line into a storage register. For example, register A accumulates the pixel charge produced by the light image. Once accumulated for a single picture, the charges are transferred line by line to register B. The pixel charges are read out line by line through a horizontal register C to an output amplifier. During readout, register A is accumulating new pixel elements The complete

cycle is repeated approximately every 1/60th of a second.

Chapter 5

LIGHTING TECHNIQUES

An essential ingredient in the application of machine vision is proper lighting. Good illumination of the scene is important because of its effect on the level of complexity of image-processing algorithms required. Poor lighting makes the task of interpreting the scene more difficult. Proper lighting techniques should provide high contrast and minimize specular reflections and shadows unless specifically designed into the system. The basic types of lighting devices used in machine vision may be grouped into the following categories:

1. Diffuse surface devices Examples of diffuse surface illuminators are the typical fluorescent lamps and light tables.
2. Condenser projectors A condenser projector transforms an expanding light source into a condensing light source. This is useful in imaging optics.
3. Flood or spot projectors Floodlights and spotlights are used to illuminate surface areas.
4. Collimators Collimators are used to provide a parallel beam of light on the subject
5. Images Images such as slide projectors and optical enlargers form an image of the target at the object plane.

Various illumination techniques have been developed to use these lighting devices.The purpose of these techniques is to direct the path of light from the lighting device to the camera so as to display the subject in a suitable manner to the camera.

There are two basic illumination techniques used in machine vision: front lighting and backlighting. Front lighting simply means that the light source is on the same side of the scene as the camera. Accordingly, reflected light is used to create the image viewed by the camera. In backlighting, the light source is directed at the camera and is located behind the objects of interest. The image seen by the camera is a silhouette of the object under study. Back lighting is suitable for applications in which a silhouette of the object is sufficient for recognition or
where there is a need to obtain relevant measurements. The table also lists other miscellaneous techniques that may be used to provide illumination.

Chapter 6

ANALOG TO DIGITAL SIGNAL CONVERSION

For a camera utilizing, the vidicon tube technology it is necessary to convert the analog signal for each pixel into a digital form. The analog-to-digital (A/D) conversion process involves taking an analog input voltage signal and producing an output that represents the voltage, signal in the digital memory of a computer. A/D conversion consists of three phases: sampling, quantization, and encoding.

Sampling A given analog signal is sampled periodically to obtain a series of discrete time analog signals. By setting a specified sampling rate, the analog signal can be approximated by the sampled digital outputs. How well we approximate the analog signal is determined by the sampling rate of the A/D converter. The sampling rate should be at least twice the highest frequency in the video signal if we wish to reconstruct the signal exactly. In practice, an allowance would have to be made for the time required when the electron beam is shut off during its raster from one line to the next. This dead time would decrease the number of pixels used in the vidicon system. For example, the number of pixels that the system could handle might be reduced from 651 pixels per line to 512 pixels per line, which is similar to the number of pixels per line used in home television. For a 512 x 512 pixel image, there are a total of 262,144 pixels to consider during the faceplate scanning period. Each pixel must be processed and some type of image processing function must be carried out during the sampling period. This results in substantial demand on the digital computer performing the function. The requirement on the capability of the vision system computer has been one of the limiting factors in the development of machine vision. Systems with a smaller number of pixels (128 x 12816384 pixels or 256 x 256 = 65,536 pixels) impose lower computational requirements: however, the image resolution of these systems is much lower than for systems possessing a greater pixel density. For a given video signal representing one line, the number of samples taken determines the horizontal resolution of the imaging system. The total number of lines determines the vertical resolution.

Quantization Each sampled discrete time voltage level is assigned to a finite number of defined amplitude levels. These amplitude levels correspond to the gray scale used in the system. The predefined amplitude levels are characteristic 10 particular A/D converter and consist of a set of discrete values of voltage levels. The number of quantization levels is defined by

Number of quantization levels 2^n

where n is the number of bits of the A/D converter. A large number of bits enables a signal to be represented more precisely. For example, an 8 bit converter would allow us to quantize at

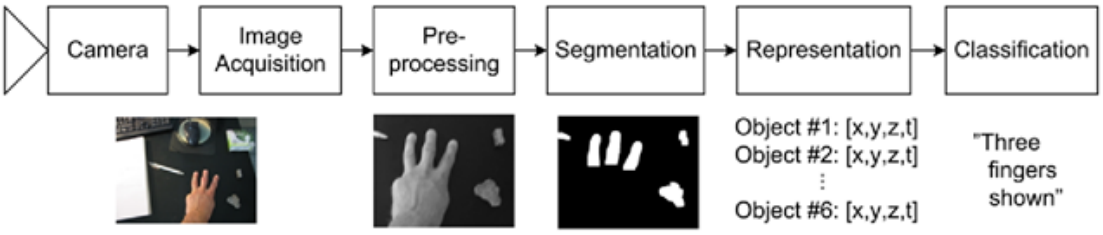
2^8 = 256 different values whereas 4 bits would allow only 2^4=16 different quantization levels.

Encoding The amplitude levels that are quantized must be changed into digital code. This process ned encoding, lves representing an amplitude level by a binary digit sequence. The ability of the encoding process to distinguish between various amplitude levels is a function of the spacing of each quantization level.

Given the full-scale range of an analog video signal, the spacing of each level would be defined by Quantization level spacing= Full-scale range/ $2^{\square}$

The quantization error resulting from the quantization process can be defined as Quantization error =±1/2 (quantization level spacing).

If we had a voltage signal which, for example, varied over a +5 V range, we could then assign the first bit to indicate polarity and use the succeeding 7 bits to identify specific voltage ranges. The number of quantization levels would be 2^7=128 and the quantization level spacing would be 5 V divided by 128 or 0.039 V.



Chapter 7

IMAGE PROCESSING AND ANALYSIS

The discussion in the preceding section described how images are obtained, digitized and stored in a computer. For use of the stored image in industrial applications, the computer must be programmed to operate on the digitally stored image. This is a substantial task considering the large amount of data that must be analyzed. Consider an industrial vision system having a pixel density of 350 pixels per line and 280 lines (a total of 98,000 picture elements), and a 6-bit register for each picture element to represent various gray levels; this would require a total of 98.000 x 6588000 bits of data for each 1/30 s. This is a formidable amount of data to be processed in a short period of time and has led to various techniques to reduce the magnitude of the image processing problem. These techniques include:

1. Image data reduction
2. Segmentation
3. Feature extraction
4. Object recognition

These techniques of image data analysis are discussed in the following subsections

The discussion in the preceding section described how images are obtained, digitized and stored in a computer. For use of the stored image in industrial applications, the computer must be programmed to operate on the digitally stored image. This is a substantial task considering the large amount of data that must be analyzed. Consider an industrial vision system having a pixel density of 350 pixels per line and 280 lines (a total of 98,000 picture elements), and a 6-bit register for each picture element to represent various gray levels; this would require a total of 98.000 x 6588000 bits of data for each 1/30 s. This is a formidable amount of data to be processed in a short period of time and has led to various techniques to reduce the magnitude of the image processing problem. These techniques include:

1. Image data reduction
2. Segmentation
3. Feature extraction
4. Object recognition

These techniques of image data analysis are discussed in the following subsections

7.1. Image Data Reduction

In image data reduction, the objective is to reduce the volume of data. As a preliminary step in the data analysis, the following two schemes have found common usage data reduction:

1. Digital conversion
2. Windowing

The function of both schemes is to eliminate the bottleneck that can occur from the large volume of data in image processing Digital conversion reduces the number of gray levels used by the machine vision system. For example, an 8-bit register used for each pixel would have $2^8 = 256$ gray levels. Depending on the requirements of the application, digital conversion can be used to reduce the number of gray levels by using fewer bits to represent the pixel light intensity. Four bits would reduce the number of gray levels to 16. This kind of conversion would significantly reduce the magnitude of the image processing problem.

Windowing involves using only a portion of the total image stored in the frame buffer for image processing and analysis. This portion is called the window. For example, for inspection of printed circuit boards, one may wish to inspect and analyze only one component on the board. A rectangular window is selected to surround the component of interest and only pixels within the window are analyzed. The rationale for windowing is that proper recognition of an object requires only certain portions of the total scene.

7.2. Segmentation

Segmentation is a general term which applies to various methods of data reduction. In segmentation, the objective is to group areas of an image having similar characteristics or features into distinct entities representing parts of the image. For example, boundaries (edges) regions (areas) represent two natural segments of an image. There are many ways to segment an image.

Three important techniques that we will discuss are:

1. Thresholding
2 Region growing
3. Edge detection

In its simplest form, thresholding is a binary conversion technique in which each pixel is converted into a binary value, either black or white. This is accomplished by utilizing a frequency histogram of the image and establishing what intensity (gray level) is to be the border between black and white. This is illustrated for an image of an object shows a regular image with each pixel having a specific

gray tone out of 256 possible gray levels. The histogram plots the frequency (number of pixels) versus the gray level for the image. For histograms that are bimodal in shape, each peak of the histogram represents either the object itself or the background upon which the object rests. Since we are trying to differentiate between the object and background, the procedure is to establish a threshold (typically between the two peaks) and assign, for example, a binary bit 1 for the object and 0 for the background. The outcome of this thresholding technique is illustrated in the binary-digitized. To improve the ability to differentiate, special lighting techniques must often be applied to generate a high contrast.

It should be pointed out that the above method of using a histogram to determine threshold is only one of a large number of ways to threshold an image. It is however the method used by many of the commercially available robot vision systems today Such a method is said to use a global threshold for the entire image. In some cases this is not possible and a local thresholding method as described below may be employed.

When it is not possible to find a single threshold for an entire image (for example if many different objects occupy the same scene, each having different levels of intensity), one approach is to partition the total image into smaller rectangular areas and determine the threshold for each window being analyzed.

Thresholding is the most widely used technique for segmentation in industrial vision applications. The reasons are that it is fast and easily implemented and that the lighting is usually controllable in an industrial setting Once thresholding is established for a particular image, the next step is to identify particular areas associated with objects within the image. Such regions usually possess uniform pixel properties computed over the area. The pixel properties may be multidimensional; that is, there may be more than a single attribute that can be used to characterize the pixel (e.g, color and light intensity). We will avoid this region complication and confine our discussion to single pixel attributes (light intensity) of Region growing is a collection of segmentation techniques in which pixels are -grouped in regions called grid elements based on attribute similarities. Defined regions can then be examined as to whether they are independent or can be merged to other regions by means of an analysis of the difference in their average properties and spatial connectedness. To differentiate between the objects and the background, assign I for any grid element occupied by an object and 0 for background elements. It is common practice to use a square sampling grid with pixels spaced equally along each side of the grid. For the two-dimensional image of a key as shown, this would give the pattern indicated. This technique of creating runs of 1s and Os is often used as a first-pass analysis to partition the image into identifiable segments or 'blobs. Note that this simple procedure did not identify the hole in the key. This could be resolved by decreasing the distance between grid points and increasing the accuracy with which the original image is represented. For a simple image such as a dark blob on a light background, a runs technique can

provide useful information. For more complex images, this technique may not provide an adequate partition of an image into a set of meaningful regions. Such regions might contain pixels that are connected to each other and have similar attributes, for example, gray level. A typical region-growing technique for complex images could have the following procedure:

1. Select a pixel that meets a criterion for inclusion in a region. In the simplest case, this could mean select white pixel and assign a value of 1.
2. Compare the pixel selected with all adjacent pixels. Assign an equival. Value to adjacent pixels if an attribute match occurs.
3. Go to an equivalent adjacent pixel and repeat the process until no equivale pixels can be added to the region.

This simple procedure of growing regions around a pixel would be repeat until no new regions can be added for the image.

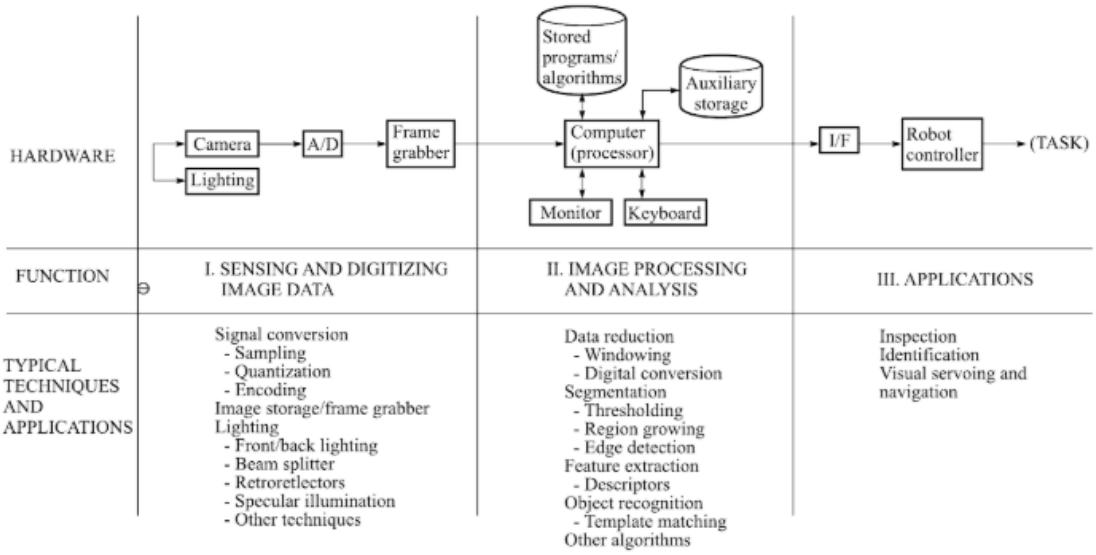
The region growing segmentation technique described here is applicable when images are not distinguishable from each other by straight thresholding or edge detection techniques. This sometimes occurs when lighting of the scene cannot be adequately controlled. In industrial robot vision systems, it is common practice to consider only edge detection or simple thresholding. This is due to the fact that lighting can be a controllable factor in an industrial setting and hardware/computational implementation is simpler.

Edge detection considers the intensity change that occurs in the pixels at the boundary or edges of a part. Given that a region of similar attributes has been found but the boundary shape is unknown, the boundary can be determined by a simple edge following procedure. This can be illustrated by the schematic of a binary image. For the binary image, the procedure is to scan the image until a pixel within the region is encountered. For a pixel within the region, turn left and step, otherwise, turn right and step. The procedure is stopped when the boundary is traversed and the path has returned to the starting pixel. The contour-following procedure described can be extended to gray level images.

7.3. Feature Extraction

In machine vision applications, it is often necessary to distinguish one object from another. This is usually accomplished by means of features that unique characterize the object. Some features of objects that can be used in machine vision include area, diameter, and perimeter. A feature in the context of vision systems, is a single parameter that permits ease of comparison and identification. The techniques available to extract feature values for two-dimensional cases can be roughly categorized as those that deal with boundary features and those that deal with area features. The various features can be used to identify the object or part and determine the part location and/or orientation.

The region-growing procedures described before can be used to determine the area of an object's image. The perimeter or boundary that encloses a specific area can be determined by noting the difference in pixel intensity at the boundary and simply counting all the pixels in the segmented region that are adjacent to pixels not in the region, that is, on the other side of the boundary. An important objective in selecting these features is that the features should not depend on position or orientation. The vision system should not be dependent on the object being presented in a known and fixed relationship to the camera.The preceding measures provide some basic methods to analyze images in a two-dimensional plane. Various other measures exist for the three-dimensional case as well.



Chapter 8

HAAR CASCADE MODEL

Haarcascade is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. It is well known for being able to detect faces and body parts in an image but can be trained to identify almost any object.
Let's take face detection as an example. Initially, the algorithm needs a lot of positive images of faces and negative images without faces to train the classifier. Then we need to extract features from it.
Here, according to our project, the Haar cascade Model is trained to determine the road signs by classifying between a lot of positive images of signs and negative images without signs.
The cascade classifier consists of a collection of stages. Each stage of the classifier labels the region defined by the current location of the sliding window as either positive or negative. Positive indicates that an object was found and negative indicates no objects were found. If the label is negative, the classification of this region is complete, and the detector slides the window to the next location. If the label is positive, the classifier passes the region to the next stage. The detector reports an object found at the current window location when the final stage classifies the region as positive.
The stages are designed to reject negative samples as fast as possible.

- A true positive occurs when a positive sample is correctly classified.
- A false positive occurs when a negative sample is mistakenly classified as positive.
- A false negative occurs when a positive sample is mistakenly classified as negative.

So, basically, a Haar Cascade is a classifier which is used to detect the object for which it has been trained for, from the source. The Haar Cascade is trained by superimposing the positive image over a set of negative images.

8.1. Positive Images

The "positive" images are images that contain the object you want to find. This can either be images that just mainly have the object, or it can be images that contain the object, and you specify the ROI (region of interest) where the object is. With these positives, we build a vector file that is basically all of these positives put together.

8.2. Negative Images

The negative images can be anything, except they cannot contain your object.

Chapter 9

TRAINING THE VISION SYSTEM

The purpose of vision system training is to program the vision system with known objects. The system stores these objects in the form of extracted feature values which can be subsequently compared against the corresponding feature values from images of unknown objects. Training of the vision system should be carried out under conditions as close to operating conditions as possible. Physical parameters such as camera placement aperture setting, part position, and lighting are the critical conditions that should be simulated as closely as possible during the training session. Vision system manufacturers have developed application software for each
individual system marketed. The software is typically based on a high-level programming language. For example, Object Recognition Systems Inc. uses the 'C language, and Automatix Inc. uses their internally developed language called RAIL (for Robot Automatic Incorporated Language). There are two versions of RAIL, one for automated vision systems and the other for robot programming.

9.1 Image dataset

A data set is a collection of related, discrete items of related data that may be accessed individually or in combination or managed as a whole entity.
Here our dataset will be comprising of collection of similar images.
We will be using the positive image as our dataset for training the HAAR CASCADE model with the negative images.

9.2 Image Labeller application in MATLAB

The Image Labeler app provides an easy way to mark a rectangular region of interest (ROI) labels, polyline ROI labels, pixel ROI labels, and scene labels in a video or image sequence. This example gets you started using the app by showing you how to:

- Manually label an image frame from an image collection.
- Automatically label across image frames using an automation algorithm.
- Export the labeled ground truth data.

9.3. Training HAAR CASCADE model using MATLAB

```
// variable storing the directory of positive images
pos_dir=fullfile('positives');

// variable storing the directory of negative images
neg_dir=fullfile('negative');

//Model
trainCascadeObjectDetector('fileName.xml',NP,neg_dir,'Num
CascadeStages',10,'FeatureType','Haar')
```

trainCascadeObjectDetector(outputXMLFileName,
positiveInstances,
negativeImages)

writes a trained cascade detector xml file with name outputXMLFileName, which must have a .xml extension.

Chapter 10

OBJECT DETECTION

The next step in image data processing is to identify the object the image represents. This identification problem is accomplished using the extracted feature information described in the previous subsection. The recognition algorithm must be powerful enough to uniquely identify the object. Object recognition techniques used in industry today may be classified into two major categories:

1. Template-matching techniques
2 Structural techniques

Template-matching techniques are a subset of the more general statistical pattern. Recognition techniques that serve to classify objects in an image into predetermined categories. The basic problem in template matching is to match the object with a stored pattern feature set defined as a model template. The model template is obtained during the training procedure in which the vision system is programmed for known prototype objects. These techniques are applicable if there is not a requirement for a large number of model templates. The procedure is based on the use of a sufficient number of features to minimize the frequency of errors in the classification process
The features of the object in the image (e.g., its area, diameter, aspect ratio, etc.) are compared to the corresponding stored values. These values constitute the stored template. When a match is found, allowing for certain statistical variations in the comparison process, then the object has been properly classified. Structural techniques of pattern recognition consider relationships between textures or edges of an object. For example, it the image of an object can be subdivided into four straight lines (the lines are called primitives) connected at their endpoints, and the connected lines are at right angles, then the object is a rectangle. This kind of technique, known as syntactic pattern recognition, is the most widely used structural technique. Structural techniques differ from decision-theoretic. Techniques in that the latter deals with a pattern on a quantitative basis and ignores for the most part interrelationships among object primitives. A detailed discussion of pattern recognition techniques is the subject of complete books and is beyond the
scope of this text It can be computationally time consuming for complete pattern recognition. Accordingly, it is often more appropriate to search for simpler regions or edges within an image. These simpler regions can then be used to extract the required features, The majority of commercial robot vision systems make use of this approach to the recognition of two-dimensional objects. The recognition algorithms are used to identify each segmented object in an image and assign it to a classification (e.g nut, bolt, flange, etc.)

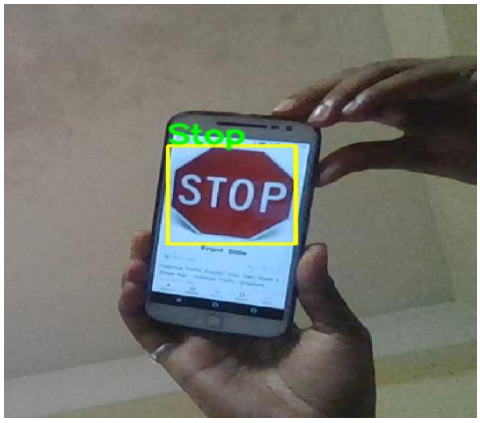
10.1. Python code for road signs recognition

```
import cv2
import numpy as np

NEntry_cascade=cv2.CascadeClassifier('NEntry.xml')
NUTurn_cascade=cv2.CascadeClassifier('NUTurn.xml')
Stop_cascade=cv2.CascadeClassifier('stop.xml')
SL_cascade=cv2.CascadeClassifier('SpdLmt.xml')
RL_cascade=cv2.CascadeClassifier('RedLight.xml')
GL_cascade=cv2.CascadeClassifier('green.xml')

cap=cv2.VideoCapture(0)

while True:
    ret, img=cap.read()
    gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    NUTurn=NUTurn_cascade.detectMultiScale(gray,1.2,40)
    Stop=Stop_cascade.detectMultiScale(gray,1.2,10)
    NEntry=NEntry_cascade.detectMultiScale(gray,1.4,30)
    SL=SL_cascade.detectMultiScale(gray,1.2,30)
    RL=RL_cascade.detectMultiScale(gray,1.3,30)
    GL=GL_cascade.detectMultiScale(gray,1.3,20)

    for (x,y,w,h) in NUTurn:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        font=cv2.FONT_HERSHEY_SIMPLEX #font
        cv2.putText(img,'No U-Turn', (x,y-3), font, 0.8,
(0,255,0), 2, cv2.LINE_AA)

    for (x,y,w,h) in Stop:
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,255),2)
        font=cv2.FONT_HERSHEY_SIMPLEX #font
        cv2.putText(img,'Stop', (x,y-3), font, 0.8,
(0,255,0), 2, cv2.LINE_AA)

    for (x,y,w,h) in NEntry:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,0),2)
        font=cv2.FONT_HERSHEY_SIMPLEX #font
        cv2.putText(img,'No Entry', (x,y-3), font, 0.8,
(0,255,0), 2, cv2.LINE_AA)

    for (x,y,w,h) in SL:
```

```
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,255),2)
        font=cv2.FONT_HERSHEY_SIMPLEX #font
        cv2.putText(img,'Speed Limit', (x,y-3), font, 0.8,
(0,255,0), 2, cv2.LINE_AA)

    for (x,y,w,h) in RL:

cv2.rectangle(img,(x,y),(x+w,y+h),(255,100,255),2)
        font=cv2.FONT_HERSHEY_SIMPLEX #font
        cv2.putText(img,'Red Light', (x,y-3), font, 0.8,
(0,255,0), 2, cv2.LINE_AA)

    for (x,y,w,h) in GL:
        cv2.rectangle(img,(x,y),(x+w,y+h),(25,100,255),2)
        font=cv2.FONT_HERSHEY_SIMPLEX #font
        cv2.putText(img,'Green Light', (x,y-3), font, 0.8,
(0,255,0), 2, cv2.LINE_AA)

    cv2.namedWindow("img", cv2.WND_PROP_FULLSCREEN)

cv2.setWindowProperty("img",cv2.WND_PROP_FULLSCREEN,cv2.W
INDOW_FULLSCREEN)
    cv2.imshow("img", img)

    if cv2.waitKey(1)==ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

RESULT OF THE PROJECT

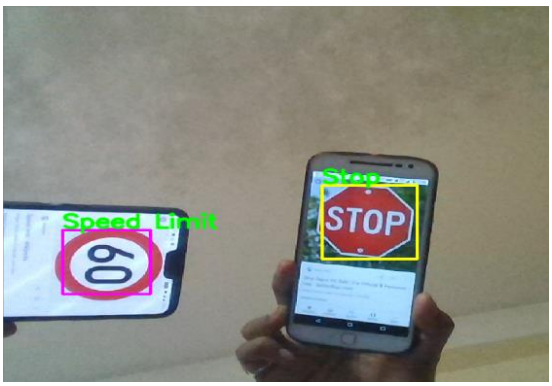
Pink square detects red light in the frame



Yellow square detects stop sign in the frame

Blue square detects no-uturn sign in the frame

Multiple detection of stop sign in the frame



Detection of different road sign in the frame

APPLICATION AND ADVANTAGES

1. Traffic Sign Recognition (TSR) is used to regulate traffic signs, warn a driver, and command or prohibit certain actions.
2. A fast real-time and robust automatic traffic sign detection and recognition can support and disburden the driver and significantly increase driving safety and comfort.
3. Automatic recognition of traffic signs is also important for automated intelligent driving vehicle or driver assistance systems.

REFERENCES

1. https://www.numpy.org/
2. https://docs.python.org/
3. https://www.wikipedia.org/
4. https://docs.opencv.org/2.4/doc/tutorials/tutorials.html/
5. Industrial Robotics by Mikkel P Groover(ISBN 10: 1259006212)
6. https://www.mathworks.com/help/vision/ug/get-started-with-the-image-labeler.html