Pedestrian Crossing Guide based on Android-Cloud Platform for Blind People

By,

Anik Hasan	09101024
Nasrat Sharif	09201013

Supervised By:

Md. Zahangir Alam
Lecturer,
Dept. of Computer Science and Engineering,
BRAC University

Co- Supervisor:

Risul Karim
Lecturer,
Dept. of Computer Science and Engineering,
BRAC University

Signature:

Date:

GRATITUDE

We would like to thank all those who were involved with our research work directly and indirectly. Contribution of our friends, seniors, juniors and our respected faculty members cannot be forgotten. Our special thanks go to our honorary chairperson Professor Dr. Zahidur Rahman.

We would also like to thank our Supervisor Md. Zahangir Alam and Co-Supervisor Risul Karim who played the key role to make this dissertation come to an end. Their contributions and guidelines are beyond saying. We are grateful for their effort.

ABSTRACT

This paper presents an automatic system which can detect road traffic and let blind people know about that so that they can easily cross the road. In the recent years, Android-driven mobile phones have got popularity among users and developers as well. In the first place, we have selected Android-driven phone to get the desired output.

We have used real time image processing system to detect traffic condition of a road. To implement this, at first we needed a camera with good resolutions, a methodology which will give the decisions and last but not the least Internet connection. The captured image will run through the application and then the application automatically sends this information to the cloud. The cloud will automatically send information to the blind by internet whether it is safe to cross the road or not. Later, it became so complex that we had to change our path and decided to make a custom application which will automatically decide whether it is safe to cross the road or not by capturing instant images and process using OpenCV.

Key Words: Optical Flow, Real Time Image Processing, Object detection, Road sign identification, Object recognition

Table of Contents

List of figures

1. Introduction

Bangladesh is one of the densely populated countries of the world. The rate of vision impaired people is too high here whereas the solutions for their blindness are beyond their reach. As the world gets involved with technology day by day it is high time to mitigate the sufferings of blind people through technologies. An Android-driven mobile phone is all we need. Initially the mobile phone will capture images continuously on what comes next to a blind person. When that person needs to cross the road he/she will just stand beside the road and the mobile phone will detect the movement of traffic by using optical flow.

1.1 Motivations:

When people walk around they do not need to think how they can judge where to go, how the internal functions of their body works or how their brains are processing images captured by their vision. However, the situation is different for a blind person. As they cannot see, their brain also does not know how to take decision if there is no vision available. At this point our idea came up. We can call it an artificial process which is linked to human brain.

1.2 Related Works:

There are certainly several works have been done by different personnel in different countries. Few of the developed countries have implemented the solutions already and these are effective too. Few of the works are mentioned below

1.2.1 <u>Mobile-Cloud based pedestrian crossing guide:</u>

The most similar work has been done by a few international students. According to them, the total process needs a spectacular lens which is Bluetooth enabled, an Android device which can detect what the lens is passing through the Bluetooth. After receiving the images, the mobile then automatically sends the information to the cloud. The cloud then processes the image with its internal methods and give the result back to the device whether it is safe or not to cross the road.

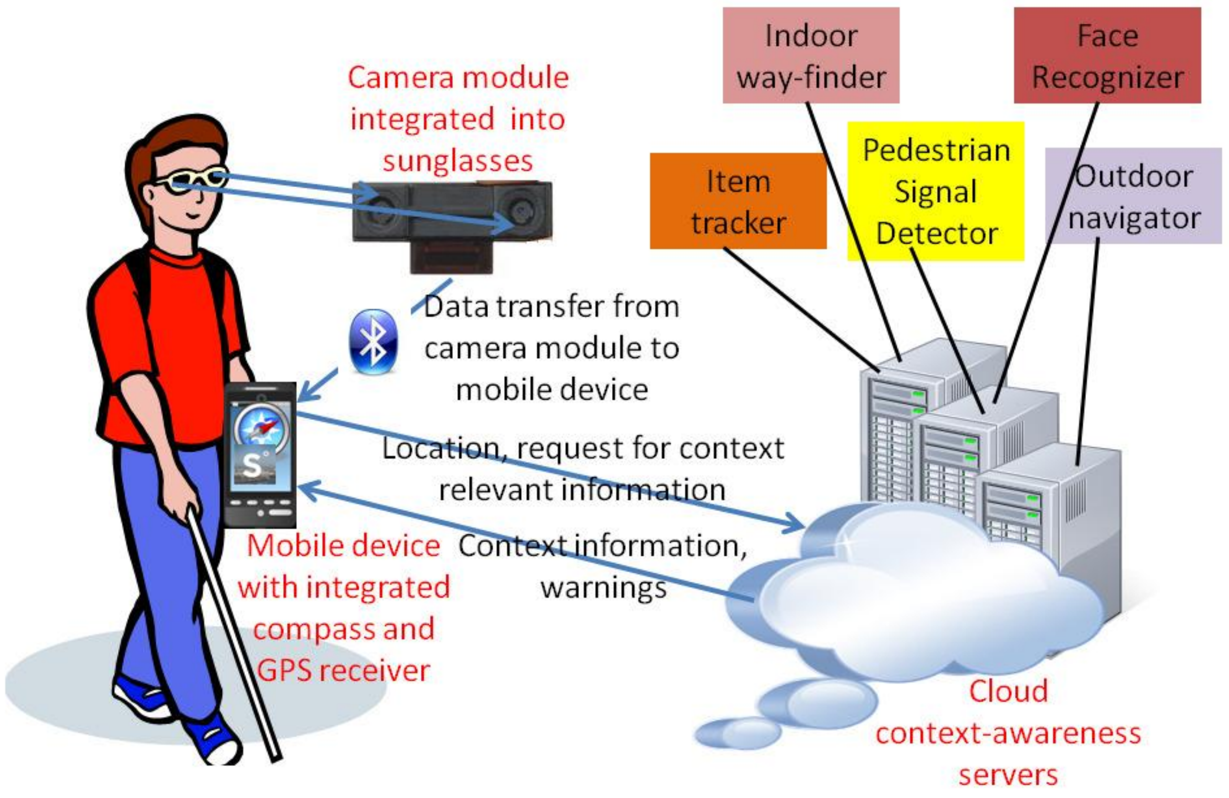
Indoor way-finder
Face Recognizer
Camera module integrated into sunglasses
Pedestrian Signal Detector
Item tracker
Outdoor navigator
Data transfer from camera module to mobile device
Location, request for context relevant information
Mobile device with integrated compass and GPS receiver
Context information, warnings
Cloud context-awareness servers

Figure 1: Mobile-cloud architecture for context-aware navigation of the blind and visually impaired

1.2.2 Real Time Mobile-Cloud computing for context-aware blind navigation

This research methodology is quite similar with the previous one. Here they have used Amazon EC2 server as a cloud. The decision making has been done here. In this process they have selected a suitable place like a zebra crossing where the traffic signal light is located. They detect the signal light with the camera placed on the cell phone and give the output by processing all lights. The selection process is quite complicated. They threshold the values by cutting all outside areas and then select the signal area only. Which is why, it becomes easier to detect current signal light.

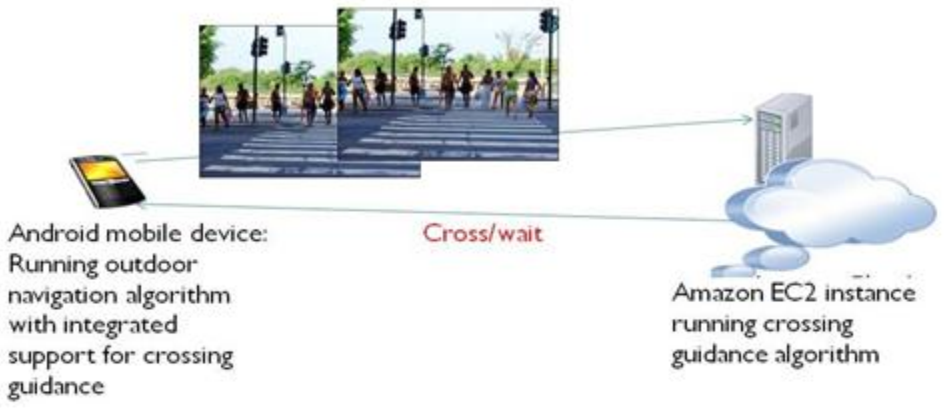
Android mobile device: Running outdoor navigation algorithm with integrated support for crossing guidance
Cross/wait
Amazon EC2 instance running crossing guidance algorithm

Figure 2: System architecture for pedestrian signal detector.

Disadvantages:

Both of the above systems are technically almost flawless. As we are implementing our research only for Bangladesh's perspective so in a sense there are some redundancies and are inappropriate to apply in our country. The first disadvantage is to use a Bluetooth enabled spectacular lens which is expensive and cannot be afforded by all classes of the people. Secondly, the image is processed and output comes from cloud server. Indeed it is efficient but the internet is mandatory here. That is why we have designed our system which processes images and gives decision using the available functions of the android-driven mobile phone. The Internet is not necessary here at all.

1.3 Research Methodology:

Our research aims to implement a system which is capable of identifying the current condition of traffic in roads. Using the image captured by the mobile phone, optical flow detects the motion of the road traffic. It sends a signal or sound to visual impaired people by earphone only if the motion is static. We have used OpenCV to implement optical flow which is available for android mobiles recently. Before, we were trying to detect traffic signal to get information about traffic but this is not efficient at all at our condition.

1.4 Outlines:

This thesis is organized in four chapters including this introduction. Chapter 2 attempts to find answer to the first question proposing and investigating the entire algorithm used here step by step to give a shape of our proposed system where an input image is taken which is processed to detect traffic. Chapter 3 is concerned about experimental results of this system and its discussions in detail. Chapter 4 summarizes the main contributions of this thesis. Finally, further research directions are suggested.

2. Overall System Implementation

2.1 Problem Definition:

Currently, Bangladesh has a vast number of blind people who are unable to buy necessary equipments which can help them to feel like they do not need to depend on any other person. At this point we have made a system to help them, so that they can at least depend on their own when it comes to cross the road. We have noted out what are the necessary solutions for the problems associated with this situation. We wanted to make sure that no blind will face out of money to get the facility.

2.2 Proposed Approach:

We divided our work in two segments. Based on this segmentation, we applied two different methodologies. We assumed a general place of human body where we can hang the mobile phone. After getting the image captured, we cropped the image in two parts known as the upper part and the lower part. The upper part generally detects the traffic signal light and pass information to the application the signal light represents a safe crossing for pedestrians. Meanwhile, the lower part detects whether there is any flow or movement of vehicle is going on or not.

To illustrate more, we have drawn an overall system implementation below.

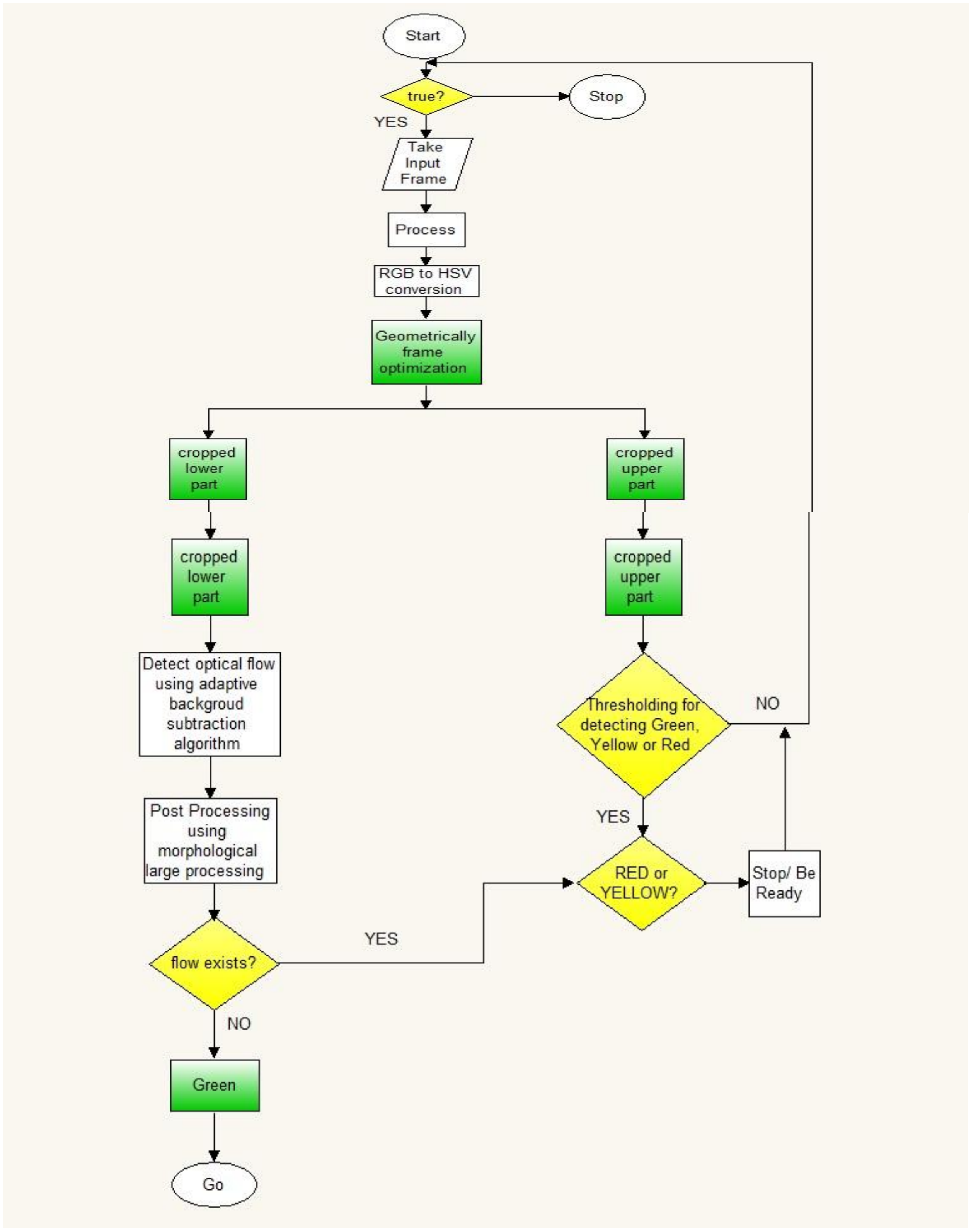
Start
true?
Stop
YES
Take Input Frame
Process
RGB to HSV conversion
Geometrically frame optimization
cropped lower part
cropped upper part
cropped lower part
cropped upper part
Detect optical flow using adaptive backgroud subtraction algorithm
Thresholding for detecting Green, Yellow or Red
NO
YES
Post Processing using morphological large processing
RED or YELLOW?
Stop/ Be Ready
YES
flow exists?
NO
Green
Go

Figure 3: Proposed System Architecture at a glance

A. Acquisition and optimization

In front of a traffic signal, the camera then moved to the direction of signal light to detect the signal is green or not.

Acquisition

Since the camera is responsible for taking the picture, it must be accurate. Without having the expected image file it is impossible to find out where the process should start.

Geometrical optimization of input frame

This is the most important part of this segment where we are optimizing the location geometrically.

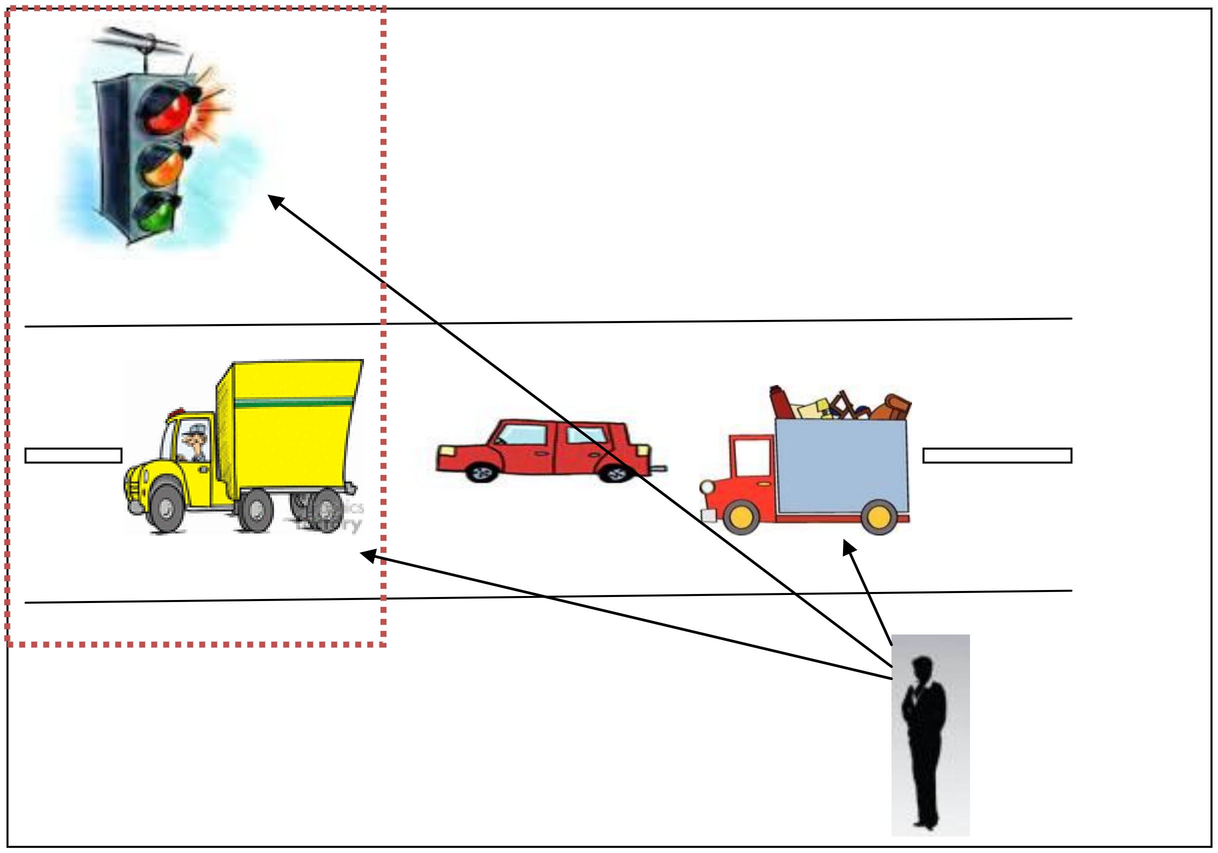

Figure 4: Geometrical optimization of input frame

In the above picture, the pedestrian is watching road traffic signal light and currently no vehicle is moving. From our system, we will be automatically notified that this is the right time to cross the road. What we have done is, we shaped a rectangular area (red dotted) where the signal light and few vehicles are included. The system is observing the signal light very carefully and using optical flow detection we are detecting the motion of the vehicles.

As we know, there are 3 signal (i.e: green, yellow and red) lights in a road side. Generally, the pedestrian is allowed to cross the road only when both the vehicle motion is static and signal is red.

B. Converting Colors from RGB to HSV

Assuming that the RGB values are normalized to be in the range [0,1], the hue angle H is measured with respect to the red axis in the range [0,360], S and V in the range [0,1]. The HSV components can be calculated from the RGB color space as follows [123]:

The value is given by

$$V=max(R,G,B)$$

The saturation component is calculated by:

$$S = \frac{\max(R,G,B) - \min(R,G,B)}{\max(R,G,B)} \qquad if \max(R,G,B) \neq 0$$

$$S = 0 \qquad if \max(R,G,B) = 0$$

The Hue is given by:

H is undefined if S=0;

$$H = \frac{G-B}{\max(R,G,B)-\min(R,G,B)} \qquad if\ R=\max(R,G,B)$$

$$H = 2.0 + \frac{B-R}{\max(R,G,B)-\min(R,G,B)} \qquad if\ G=\max(R,G,B)$$

$$H = 4.0 + \frac{R-G}{\max(R,G,B)-\min(R,G,B)} \qquad if\ B=\max(R,G,B)$$

$$H=60xH$$

$$if\ H<0\ then \qquad H=H+360$$

Where *H* is the hue angle measured from RED.

C. Traffic signal segmentation and extraction

The first part of our system is Traffic signal segmentation and extracts the desired area to get more accurate result. To do this, we need an android driven mobile along with a digital camera attached to it. To elaborate more, we can divide this part by two segments.

2.3 Algorithms used to implement this total Architecture:

We have used few algorithms here to implement the total system. All have specific work to do.

- Lucas & Kanade method
- Background subtraction method
- Hough circle transform
- Good features to track

i. **Lucas & Kanade method:**

The Lucas–Kanade method is a widely used differential method for optical flow estimation developed by Bruce D. Lucas and Takeo Kanade. It assumes that the flow is essentially constant in a local neighborhood of the pixel under consideration, and solves the basic optical flow equations for all the pixels in that neighborhood, by the least squares criterion.

By combining information from several nearby pixels, the Lucas–Kanade method can often resolve the inherent ambiguity of the optical flow equation. It is also less sensitive to image noise than point-wise methods. On the other hand, since it is a purely local method, it cannot provide flow information in the interior of uniform regions of the image.

Concept:

The Lucas–Kanade method assumes that the displacement of the image contents between two nearby instants (frames) is small and approximately constant within a neighborhood of the point *p* under consideration. Thus the optical flow equation can be assumed to hold for all pixels within a window centered at *p*.

Namely, the local image flow (velocity) vector (V_x, V_y) must satisfy

$$\begin{aligned}
I_x(q_1)V_x + I_y(q_1)V_y &= -I_t(q_1)\\
I_x(q_2)V_x + I_y(q_2)V_y &= -I_t(q_2)\\
&\vdots\\
I_x(q_n)V_x + I_y(q_n)V_y &= -I_t(q_n)
\end{aligned}$$

where $q_1, q_2, \ldots, q_n$ are the pixels inside the window,and $I_x(q_i), I_y(q_i), I_t(q_i)$ are the partial derivatives of the image I with respect to position *x*, *y* and time *t*, evaluated at the point $q_i$and at the current time.

These equations can be written in matrix form $Av = b$, where

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix}, \quad v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, \quad \text{and} \quad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix}$$

This system has more equations than unknowns and thus it is usually over-determined. The Lucas–Kanade method obtains a compromise solution by the least squares principle. Namely, it solves the 2×2 system

$A^T A v = A^T b$ or

$\mathrm{v} = (A^T A)^{-1} A^T b$

where A^T is the transpose of matrix A. That is, it computes

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i) I_y(q_i) \\ \sum_i I_y(q_i) I_x(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i) I_t(q_i) \\ -\sum_i I_y(q_i) I_t(q_i) \end{bmatrix}$$

with the sums running from *i*=1 to *n*.

The matrix $A^T A$ is often called the structure tensor of the image at the point *p*.

Weighted Window:

The plain least squares solution above gives the same importance to all *n* pixels q_i in the window. In practice it is usually better to give more weight to the pixels that are closer to the central pixel *p*. For that, one uses the weighted version of the least squares equation,

$$A^T W A v = A^T W b$$

or

$$\mathrm{v} = (A^T W A)^{-1} A^T W b$$

where W is an *n*×*n* diagonal matrix containing the weights $W_{ii} = w_i$ to be assigned to the equation of pixel q_i. That is, it computes

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i w_i I_x(q_i)^2 & \sum_i w_i I_x(q_i) I_y(q_i) \\ \sum_i w_i I_x(q_i) I_y(q_i) & \sum_i w_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i w_i I_x(q_i) I_t(q_i) \\ -\sum_i w_i I_y(q_i) I_t(q_i) \end{bmatrix}$$

The weight w_i is usually set to a Gaussian function of the distance between q_i and *p*.

ii. Background subtraction method:

Background subtraction, also known as Foreground Detection, is a technique in the fields of image processing and computer vision wherein an image's foreground is extracted for further processing (object recognition etc.). Generally an image's regions of interest are objects (humans, cars, text etc.) in its foreground. After the stage of image preprocessing (which may include image denoising etc.) object localisation is required which may make use of this technique. Background subtraction is a widely used approach for detecting moving objects in videos from static cameras. The rationale in the approach is that of detecting the moving objects from the difference between the current frame and a reference frame, often called "background image", or "background model". Background subtraction is mostly done if the image in question is a part of a video stream.

Approaches:

A robust background subtraction algorithm should be able to handle lighting changes, repetitive motions from clutter and long-term scene changes. The following analyses make use of the function of $V(x,y,t)$ as a video sequence where t is the time dimension, x and y are the pixel location variables. e.g. $V(1,2,3)$ is the pixel intensity at (1,2) pixel location of the image at $t = 3$ in the video sequence.

Using frame differencing:

Frame difference (absolute) at time $t + 1$ is

$$D(t+1) = |V(x,y,t+1) - V(x,y,t)|$$

The background is assumed to be the frame at time t. This difference image would only show some intensity for the pixel locations which have changed in the two frames. Though we have seemingly removed the background, this approach will only work for cases where all foreground pixels are moving and all background pixels are static.

A threshold "Th" is put on this difference image to improve the subtraction (see Image thresholding).

$$|V(x,y,t) - V(x,y,t+1)| > \mathrm{Th}$$

(this means that the difference image's pixels' intensities are 'thresholded' or filtered on the basis of value of Th)

The accuracy of this approach is dependent on the speed of movement in the scene. Faster movements may require higher thresholds.

<u>Mean filter:</u>

For calculating the image containing only the background, a series of preceding images are averaged. For calculating the background image at the instant *t*,

$$B(x,y) = \frac{1}{N}\sum_{i=1}^{N} V(x,y,t-i)$$

where *N* is the number of preceding images taken for averaging. This averaging refers to averaging corresponding pixels in the given images. *N* would depend on the video speed (number of images per second in the video) and the amount of movement in the video.[3] After calculating the background *B(x,y)* we can then subtract it from the image *V(x,y,t)* at time *t*=t and threshold it. Thus the foreground is

$$|V(x,y,t) - B(x,y)| > \mathrm{Th}$$

where *Th* is threshold. Similarly we can also use median instead of mean in the above calculation of *B*(*x*,*y*).

Usage of global and time-independent Thresholds (same Th value for all pixels in the image) may limit the accuracy of the above two approaches.

Running Gaussian average:

For this method, Wren et al. propose fitting a Gaussian probabilistic density function (pdf) on the most recent n frames. In order to avoid fitting the pdf from scratch at each new frame time t, a running (or on-line cumulative) average is computed.

The pdf of every pixel is characterized by mean μ_t and variance σ_t^2 . The following is a possible initial condition (assuming that initially every pixel is background):

$\mu_0 = I_0$

$\sigma_0^2 = <$ some default value $>$

where I_t is the value of the pixel's intensity at time t. In order to initialize variance, we can, for example, use the variance in x and y from a small window around each pixel.

Note that background may change over time (e.g. due to illumination changes or non-static background objects). To accommodate for that change, at every frame t, every pixel's mean and variance must be updated, as follows:

$\mu_t = \rho I_t + (1-\rho)\mu_{t-1}$

$\sigma_t^2 = d^2\rho + (1-\rho)\sigma_{t-1}^2$

$d = |(I_t - \mu_t)|$

Where ρ determines the size of the temporal window that is used to fit the pdf (usually $\rho = 0.01$) and d is the Euclidean distance between the mean and the value of the pixel.

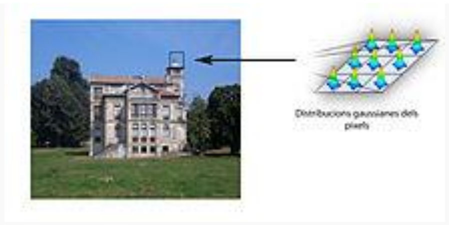

Figure 5: Running Gaussian Average

Gaussian distribution for each pixel.

We can now classify a pixel as background if its current intensity lies within some confidence interval of it's distribution's mean:

$$\frac{|(I_t - \mu_t)|}{\sigma_t} > k \longrightarrow Foreground$$
$$\frac{|(I_t - \mu_t)|}{\sigma_t} \leq k \longrightarrow Background$$

where the parameter k is a free threshold (usually $k = 2.5$). A larger value for k allows for more dynamic background, while a smaller k increases the probability of a transition from background to foreground due to more subtle changes.

In a variant of the method, a pixel's distribution is only updated if it is classified as background. This is to prevent newly introduced foreground objects from fading into the background. The update formula for the mean is changed accordingly:

$$\mu_t = M\mu_{t-1} + (1 - M)(I_t\rho + (1 - \rho)\mu_{t-1})$$

where $M = 1$ when I_t is considered foreground and $M = 0$ otherwise. So when $M = 1$, that is, when the pixel is detected as foreground, the mean will stay the same. As a result, a pixel, once it has become foreground, can only become background again when the intensity value gets close to what it was before turning foreground. This method, however, has several issues: It only works if all pixels are initially background pixels (or foreground pixels are annotated as such). Also, it cannot cope with gradual background changes: If a pixel is categorized as foreground for a too long period of time, the background intensity in that location might have changed (because illumination has changed etc.). As a result, once the foreground object is gone, the new background intensity might not be recognized as such anymore.

<u>Background mixture models:</u>

In this technique, it is assumed that every pixel's intensity values in the video can be modeled using a Gaussian mixture model. A simple heuristic determines which intensities are most probably of the background. Then the pixels which do not match to these are called the foreground pixels. Foreground pixels are grouped using 2D connected component analysis.

At any time t, a particular pixel (x_0, y_0)'s history is

$$X_1, \ldots, X_t = \{V(x_0, y_0, i) : 1 \leqslant i \leqslant t\}$$

This history is modeled by a mixture of *K* Gaussian distributions:

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} N\left(X_t \mid \mu_{i,t}, \Sigma_{i,t}\right)$$

where

$$N\left(X_t \mid \mu_{it}, \Sigma_{i,t}\right) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma_{i,t}|^{1/2}} \exp\left(-\frac{1}{2}(X_t - \mu_{i,t})^T \Sigma_{i,t}^{-1}\left(X_t - \mu_{i,t}\right)\right)$$

An on-line K-means approximation is used to update the Gaussians. Numerous improvements of this original method developed by Stauffer and Grimson have been proposed and a complete survey can be found in Bouwmans et al.

iii. Hough circle transform

The Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform.

The classical Hough transform was concerned with the identification of lines in the image, but later the Hough transform has been extended to identifying positions of arbitrary shapes, most commonly circles or ellipses. The Hough transform as it is universally used today was invented by Richard Duda and Peter Hart in 1972, who called it a "generalized Hough transform" after the related 1962 patent of Paul Hough. The transform was popularized in the computer vision community by Dana H. Ballard through a 1981 journal article titled "Generalizing the Hough transform to detect arbitrary shapes".

Implementations:

The linear Hough transform algorithm uses a two-dimensional array, called an accumulator, to detect the existence of a line described by $r = x\cos\theta + y\sin\theta$. The dimension of the accumulator equals the number of unknown parameters, i.e., two, considering quantized values of r and θ in the pair (r,θ). For each pixel at *(x,y)* and its neighborhood, the Hough transform algorithm determines if there is enough evidence of a straight line at that pixel. If so, it will calculate the parameters (r,θ) of that line, and then look for the accumulator's bin that the parameters fall into, and increment the value of that bin. By finding the bins with the highest values, typically by looking for local maxima in the accumulator space, the most likely lines can be extracted, and their (approximate) geometric definitions read off. (Shapiro and Stockman, 304) The simplest way of finding these *peaks* is by applying some form of threshold, but other techniques may yield better results in different circumstances - determining which lines are found as well as how many. Since the lines returned do not contain any length information, it is often necessary, in the next step, to find which parts of the image match up with which lines. Moreover, due to imperfection errors in the edge detection step, there will usually be errors in the accumulator space, which may make it non-trivial to find the appropriate peaks, and thus the appropriate lines.

The final result of the linear Hough transform is a two-dimensional array (matrix) similar to the accumulator -- one dimension of this matrix is the quantized angle θ and the other dimension is the quantized distance r. Each element of the matrix has a value equal to the number of points or pixels that are positioned on the line represented by quantized parameters (r, θ). So the element with the highest value indicates the straight line that is most represented in the input image.

Using the gradient direction to reduce the number of votes

An improvement suggested by O'Gorman and Clowes can be used to detect lines if one takes into account that the local gradient of the image intensity will necessarily be orthogonal to the edge. Since edge detection generally involves computing the intensity gradient magnitude, the gradient direction is often found as a side effect. If a given point of coordinates (*x,y*) happens to indeed be on a line, then the local direction of the gradient gives the ϑ parameter corresponding to said line, and the *r* parameter is then immediately obtained. (Shapiro and Stockman, 305) The gradient direction can be estimated to within 20°, which shortens the sinusoid trace from the full 180° to roughly 45°. This reduces the computation time and has the interesting effect of reducing the number of useless votes, thus enhancing the visibility of the spikes corresponding to real lines in the image.

Kernel-based Hough transform

Fernandes and Oliveira suggested an improved voting scheme for the Hough transform that allows a software implementation to achieve real-time performance even on relatively large images (e.g., 1280×960). The Kernel-based Hough transform uses the same (r, θ) parameterization proposed by Duda and Hart but operates on clusters of approximately collinear pixels. For each cluster, votes are cast using an oriented elliptical-Gaussian kernel that models the uncertainty associated with the best-fitting line with respect to the corresponding cluster. The approach not only significantly improves the performance of the voting scheme, but also produces a much cleaner accumulator and makes the transform more robust to the detection of spurious lines.

Hough transform of curves, and its generalization for analytical and non-analytical shapes:

Although the version of the transform described above applies only to finding straight lines, a similar transform can be used for finding any shape which can be represented by a set of parameters. A circle, for instance, can be transformed into a set of three parameters, representing its center and radius, so that the Hough space becomes three dimensional. Arbitrary ellipses and curves can also be found this way, as can any shape easily expressed as a set of parameters.

The generalization of the Hough transform for detecting analytical shapes in spaces having any dimensionality was proposed by Fernandes and Oliveira. In contrast to other Hough transform-based approaches for analytical shapes, Fernandes' technique does not depend on the shape one wants to detect nor on the input data type. The detection can be driven to a type of analytical shape by changing the assumed model of geometry where data have been encoded (e.g., euclidean space, projective space, conformal geometry, and so on), while the proposed formulation remains unchanged. Also, it guarantees that the intended shapes are represented with the smallest possible number of parameters, and it allows the concurrent detection of different kinds of shapes that best fit an input set of entries with different dimensionalities and different geometric definitions (e.g., the concurrent detection of planes and spheres that best fit a set of points, straight lines and circles).

For more complicated shapes in the plane (i.e., shapes that cannot be represented analytically in some 2D space), the generalized is used, which allows a feature to vote for a particular position, orientation and/or scaling of the shape using a predefined look-up table.

Circle detection process:

The process of identifying possible circular objects in Hough space is relatively simple,

- First we create our accumulator space which is made up of a cell for each pixel; initially each of these will be set to 0.
- For each (edge point in image (i, j)): Increment all cells which according to the equation of a circle ($(i-a)^2 + (j-b)^2 = r^2$) could be the center of a circle, these cells are represented by the letter 'a' in the equation.
- For all possible value of a found in the previous step, find all possible values of b which satisfy the equation.
- Search for the local maxima cells, these are any cells whose value is greater than every other cell in its neighborhood. These cells are the one with the highest probability of being the location of the circle(s) we are trying to locate.

Note that in most problems we will know the radius of the circle we are trying to locate beforehand, however if this is not the case we can use a 3 dimensional accumulator space, this is much more computationally expensive. This method can also detect circles that are partially outside of the accumulator space if enough of its area is still present within it.

Using weighted features:

One common variation detail is, finding the bins with the highest count in one stage can be used to constrain the range of values searched in the next.

D. Thresholding and feature extraction

We have followed few internal strategies to accomplish our goal. Thresholding is one of these. It is done basically to consider a standard. By comparing with that value, we can detect our next step based on that value.

E. Signal detection and output generation

Signal detection is the next work we have done to achieve our goal. In this phase, we actually wanted to detect the signal to understand digitally whether it is safe to cross the road or not.

By getting the image, we made a decision with the text initially by notifying that the road is safe to cross. Our target is to make the output as Audio.

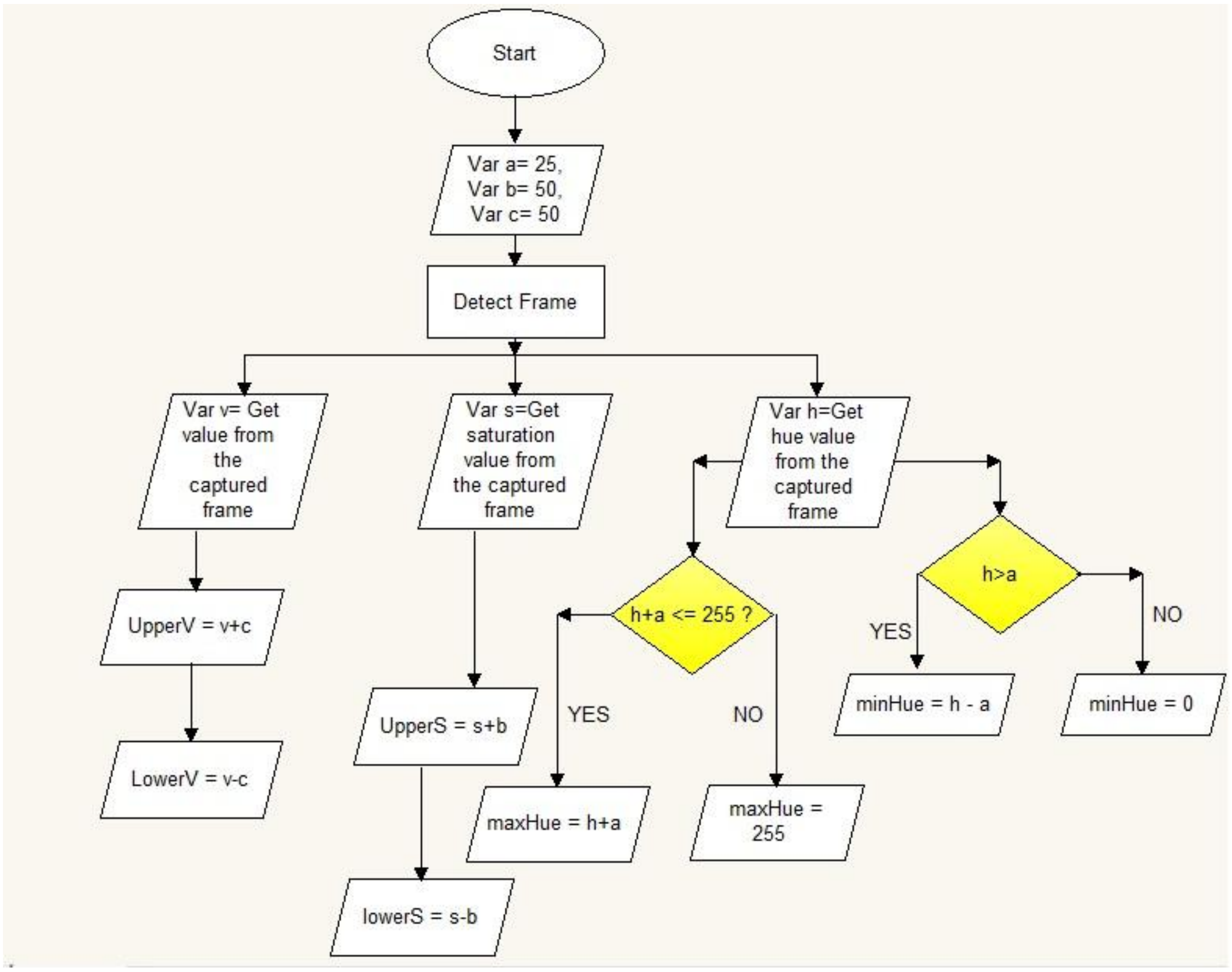

Figure 6: Getting the Upper and Lower HSV values

To detect the H, S and V value, we have considered a constant value for each of the values. In the above flow chart, we have assumed variable a, b and c as the constant values which will be compared with the values from captured frame. Let's consider the situation for finding minimum hue and maximum hue. After capturing the hue value from the captured frame, if the captured value is greater than the value we considered constant, then minimum hue will become the subtracted result between captured value and constant value. Otherwise, minimum hue will become 0.

Then again, to find out maximum value for hue, we add the captured hue value and constant hue value and compare the added value with 255. If the added value is greater than 255, then maximum hue will become 255 otherwise maximum hue will be the addition of these two values.

To get the Saturation value from the captured image, we simply add constant value and captured frame value to get the upper saturation value and to get the lower saturation value we simply subtract the constant saturation value from captured saturation value.

Similarly, to get the Upper ‘value’, we simply add constant value and captured value and subtract to get the lower one.

2.4 Android Application:

Our application actually is doing all the things we need to do to get our desired output. It is measuring every necessary step and performs operations like we have instructed. We have tested this application in an upgraded tab where android version is ice-cream sandwich (4.0.4). Starting from capturing the image of a road it then divides that image into upper and lower part. After getting two separated parts, it takes necessary area to detect signal light and movement of the vehicles.

3. Results and Discussions

We have successfully implemented our system for blind people from the perspective of our country. We had faced a lot of obstacles while implementing our system. When we started our work we assumed traffic system in Bangladesh will not be that much difficult that can create barrier in any type of solutions. As we have differentiated two phases to complete our work, our goal was to merge these two phases to complete the whole process successfully. Individually, we have done these two phases but when the point came out to merge, we could not do that properly due to the image acquisition problem.

With the proper hardware associated with our operations and algorithms attached to it, it definitely can be a real good system for the blind people of our country.

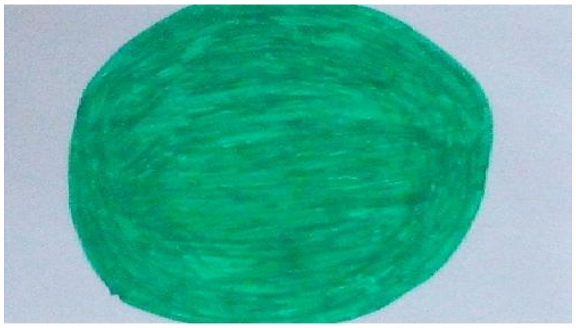
Figure 7: Sample Green

Figure 8: Sample Red

We have used our device camera to detect signal lights, but to get accurate result in an artificial environment we have used two hand drawn red and green circles on papers.

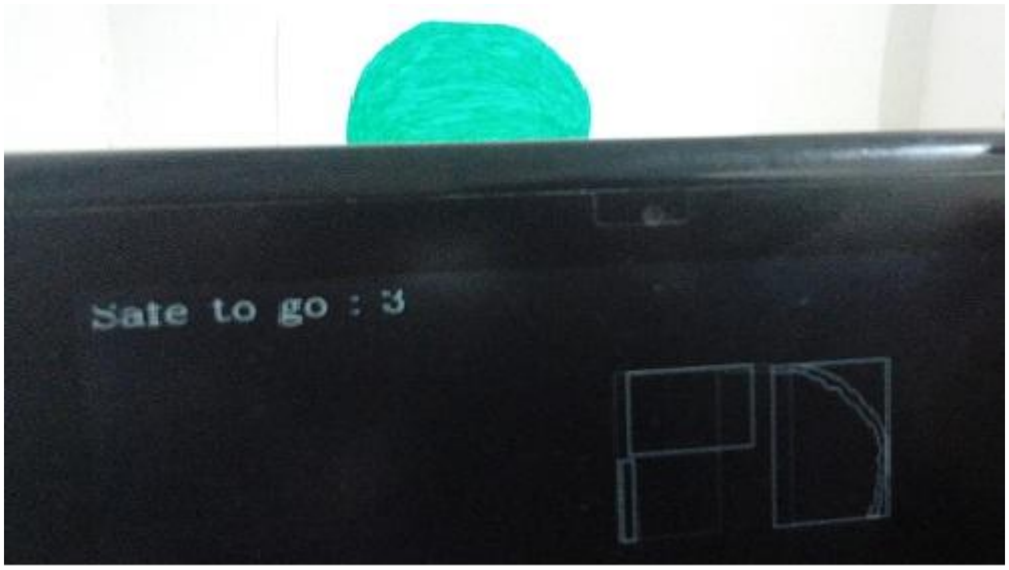

Figure 9: First Output

We can see from the image above that it is successfully detecting the green image and showing a message that it is safe to cross the road.

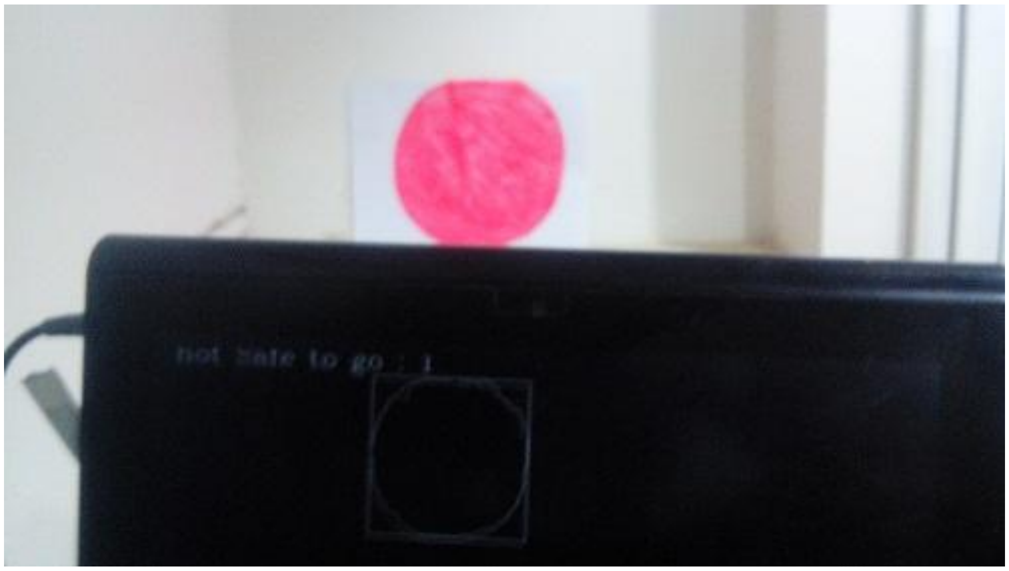

Figure 10: Second Output

This above image is also successfully detecting the red image and showing a message that it is not safe to cross the road right now.

4. Conclusion and Future Work

4.1 Conclusion:

In this paper we presented a complete system for road signal recognition and recommendation for blind people. Experiment consists of images with different size, lighting, background and distance etc. Due to the variation of light the acquisition of video from outside environment is very challenging. Moreover, feature extraction from those video signals challenging as well. The experimental results show that, road signal are extracted truly with higher success and instructions respect to the signal have been displayed on the screen.

4.2 Features for further implementations:

When we went to implement our work, we found many other things which we could not do because of the lack of the time. Few of them are enlisted below-

i. The first thing we would do if we would get more time is, we would merge two steps together to get accurate output. Detecting vehicle movement is the first priority always where RED traffic signal is a must as well to cross road safely. Since we could not merge these two steps together, we can not say our system is hundred percent efficient for crossing the road. According to our system, merging these two is a must.

ii. Initially, when we started our journey we had no idea how to actually complete everything so efficiently. Because of the lack of the idea, we did not notice that what will happen if we face multiple signal location. To find out a suitable place to detect road traffic we found many important places where signal is indicating multiple road traffic. According to our system, we can only consider a situation where only one way traffic signal exists. In future, we are planning to make this system for multiple traffic signal point.

iii. According to our system, we can detect two way movement of traffic to determine whether it is safe to cross the road or not. We have already tested that. But the time allocated was not sufficient to make it for the multiple road traffics. As we are a developing country, the multiple traffic ways is going to take place very soon. So to make this system more efficient and to fulfill the demand of age, it is very important to implement this idea in our system.

iv. When we went to capture the video, we suddenly noticed that things do not work like we assumed based on the theory. While getting practical experience, we became failed several times because of the lack of the performance of our application. Although it was not our fault but this is a must work for us to upgrade the performance of our application through the best combinations of hardware and software.

References

1. B. D. Lucas and T. Kanade (1981), An iterative image registration technique with an application to stereo vision. Proceedings of Imaging Understanding Workshop, pages 121—130
2. Charette, R. and Nashashibi, F. Real Time Visual Traffic Lights Recognition Based on Spot Light Detection and Adaptive Traffic Lights Templates. In *World Congress and Exhibition on Intelligent Transport Systems and Services* (2009).
3. Crandall, W., Brabyn, J., Bentzen, B.L., and Myers, L. Remote Infrared Signage Evaluation for Transit Stations and Intersections. *Journal of Rehabilitation Research and Development*, 36, 4 (1999), 341-355.
4. Ivanchenko, V., Coughlan, J., and Shen, H. Detecting and locating crosswalks using a camera phone. In *Computer Vision and Pattern Recognition Workshops* (2008).
5. Kim, Y.K., Kim, K.W., and Yang, X. Real Time Traffic Light Recognition System for Color Vision Deficiencies. In *IEEE International Conference on Mechatronics and Automation* (2007).
6. Shioyama, T., Wu, H., Nakamura, N., and Kitawaki, S. Measurement of the length of pedestrian crossings and detection of traffic lights from image data. *MEASUREMENT SCIENCE AND TECHNOLOGY*, 13 (2002), 1450-1457.
7. Uddin, M. and Shioyama, T. Detection of Pedestrian Crossing using Bipolarity and Projective Invariant. In *IAPR Conference on Machine Vision Applications* (2005).
8. Bohonos, S., Lee, A., Malik, A., Thai, C., and Manduchi, R. Universal real-time navigational assistance (URNA): An urban bluetooth beacon for the blind. In *1st ACM SIGMOBILE International Workshop on Systems and Networking Support for Healthcare and Assisted Living Environments* (2007).
9. Freund, Y. and Schapire, R.E. A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55 (1997), 119-139.

10. Lienhart, R. and Maydt, J. An Extended Set of Haar-Like Features for Rapid Object Detection. In *IEEE International Conference on Image Processing* (2002).
11. Pleis, J.R. and Lethbridge-Çejku, M. Summary health statistics for U.S. adults: National Health Interview Survey, 2006. National Center for Health Statistics, 2007.
12. Giudice, N.A. and G.E. Legge. Blind Navigation and the Role of Technology. In A. Helal, M. Mokhtari and B. Aldulrazak, ed., The Engineering Handbook of Smart Technology for Aging, Disability and Independence. John Wiley & Sons, Hoboken, New Jersey, 2008.
13. Angin, P., Bhargava, B., and Helal, S. A Mobile-Cloud Collaborative Traffic Lights Detector for Blind Navigation. In *Mobile Data Management*, 2010.
14. Gallo, O., Manduchi, R., and Rafii, A. Robust Curb and Ramp Detection for Safe Parking using the Canesta TOF camera. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* (2008).
15. Grundmann, T., Eidenberger, R., Zoellner, R.D., Zhixing, X., Ruehl, S., Zoellner, J.M., Dillmann, R., Kuehnle, J., and Verl, A. Integration of 6D Object Localization and Obstacle Detection for Collision Free Robotic Manipulation. In *IEEE/SICE International Symposium on System Integration* (2008).
16. Bostelman, R.V., Hong, T.H., and Madhavan, R. Towards AGV Safety and Navigation Advancement - Obstacle Detection using a TOF Range Camera. In *International Conference on Advanced Robotics* (2005).
17. Tombari, F., Stefano, L., Mattoccia, S., and Zanetti, A. Graffiti Detection Using a Time-of-Flight Camera. In *10th International Conference on Advanced Concepts for Intelligent Vision Systems* (2008).
18. Ringbeck, T., Moller, T., and Hagebeuker, B. Multidimensional Measurement by Using 3-D PMD sensors. *Advances in Radio Science*, 5 (2007), 135-146.