ROAD DETECTION FOR AUTONOMOUS NAVIGATION

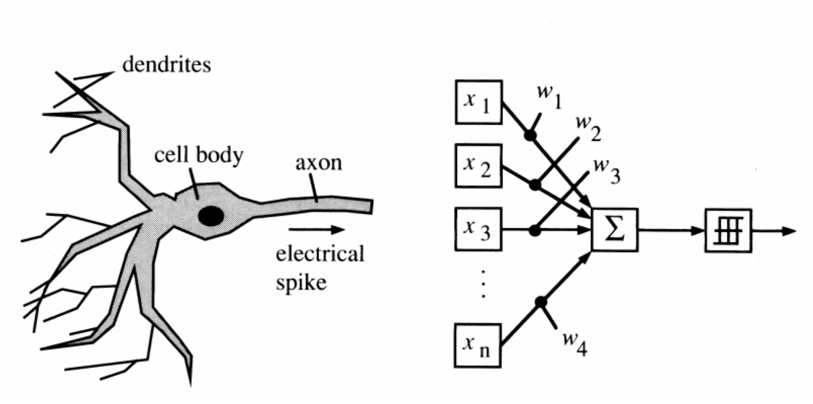
ROAD DETECTION USING MACHINE LEARNING

In this approach, neural networks are used for segmentation of road against the surrounding. The final output should contain the road as intensity zero (black) pixels and everything else in the surrounding as full intensity (white) pixels.

NEURAL NETWORKS

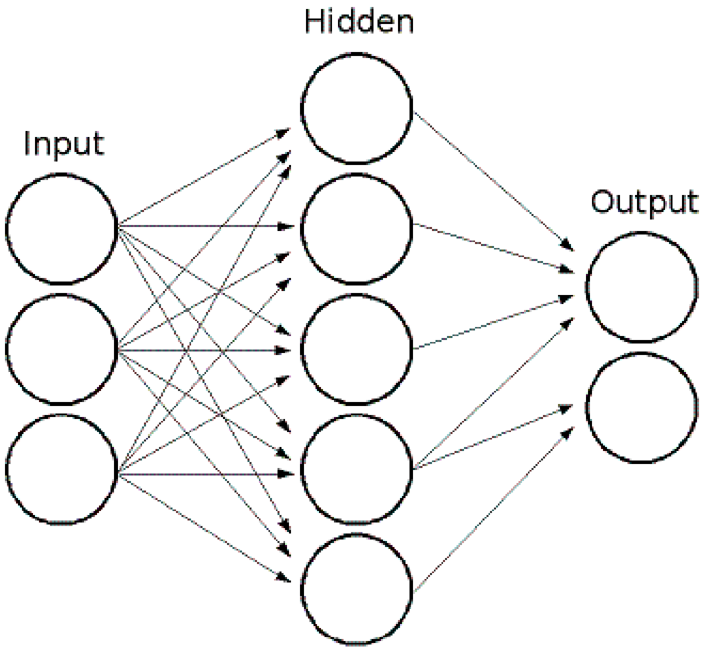
Artificial neural networks (**ANNs**) are computational models inspired by an animal's central nervous systems (in particular the brain) which is capable of machine learning as well as pattern recognition. Artificial neural networks are generally presented as systems of interconnected "neurons" which can compute values from inputs.

For example, a neural network for handwriting recognition is defined by a set of input neurons which may be activated by the pixels of an input image. After being weighted and transformed by a function (determined by the network's designer), the activations of these neurons are then passed on to other neurons. This process is repeated until finally, an output neuron is activated. This determines which character was read.



Structure of Neural Network

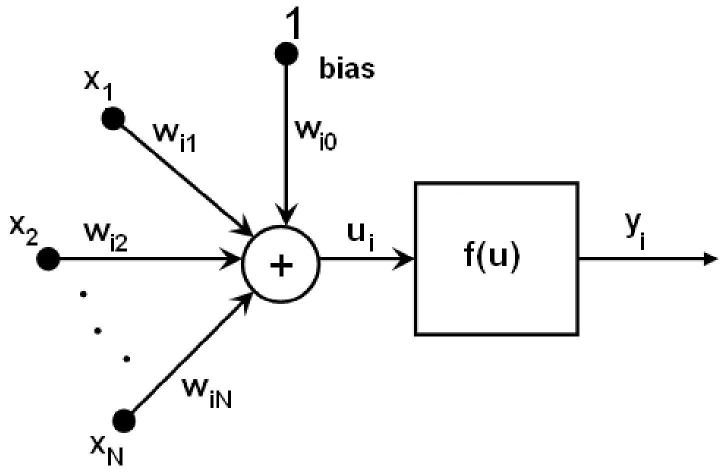
- It consists of an input layer, output layer and one or more hidden layers
- Different features if input data are fed into input layer.
- The output layer consists of neurons which may be suitably grouped to represent a particular class.



MECHANISM OF MULTI-LAYER PERCEPTION

- Every single unit (neuron) has several input links and several output links.
- The values retrieved from the previous layer are summed up with certain weights, individual for each neuron, plus the bias term.
- Given the outputs of previous layer, the inputs to next layer are computed as:

$$u_i = \sum_j (w_{i,j}^{n+1} * x_j) + w_{i,bias}^{n+1}$$



Training Multi-Layer Perception using Backpropogation algorithm

Phase 1: Propagation

- Each propagation involves the following steps:

- Forward propagation of a training pattern's input through the neural network in order to generate the propagation's output activations.
- Backward propagation of the propagation's output activations through the neural network using the training pattern target in order to generate the deltas of all output and hidden neurons.

Phase 2: Weight update

- For each weight-synapse follow the following steps:
- Multiply its output delta and input activation to get the gradient of the weight.
- Subtract a ratio (percentage) of the gradient from the weight.

The steps 1 and 2 are repeated for specified number of iterations or optimal difference in weights between successive iterations.

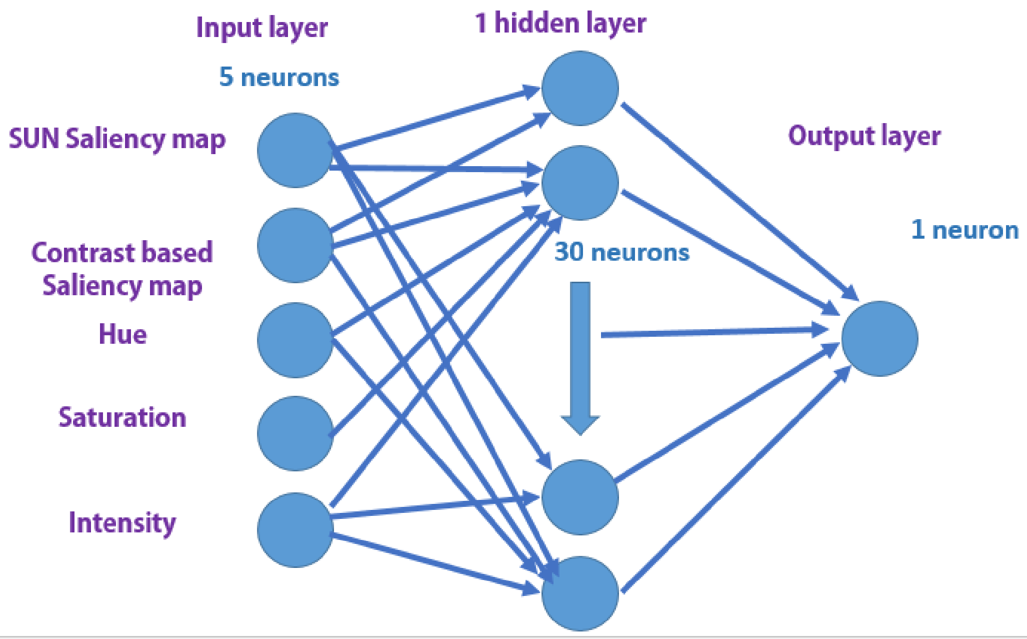
Benefits of Neural Networks

1. **Input-output mapping:** Each sample consists of a unique input signal and the corresponding desired response. The network is presented a sample picked at random from the set, and the synaptic weights (free parameters) of the network are modified so as to minimize the difference between the desired response and the actual response of the network produced by the input signal in accordance with an appropriate criterion. The training of the network is repeated for many samples in the set until the network reaches a steady state, where there are no further significant changes in the synaptic weights.
2. **Adaptivity**: Neural networks have a built-in capability to adapt their synaptic weights to changes in the surrounding environment. In particular, a neural network trained to operate in a specific environment can be easily retrained to deal with minor changes in the operating environmental conditions.

3. **Contextual information**: Knowledge is represented by the very structure and activation state of a neural network. Every neuron in the network is potentially affected by the global activity of all other neurons in the network. Consequently, contextual information is dealt with naturally by a neural network.
4. **Nonlinearity:** A neuron is basically a nonlinear device. Consequently, a neural network, made up of an interconnection of neurons, is itself nonlinear.

Neural Network Structure in present context

- ❖ Input layer consists of all 5 features of single pixel.
- ❖ Output layer gives probability of pixel belonging to the road.
- ❖ It consists of single hidden layer with 30 neurons.



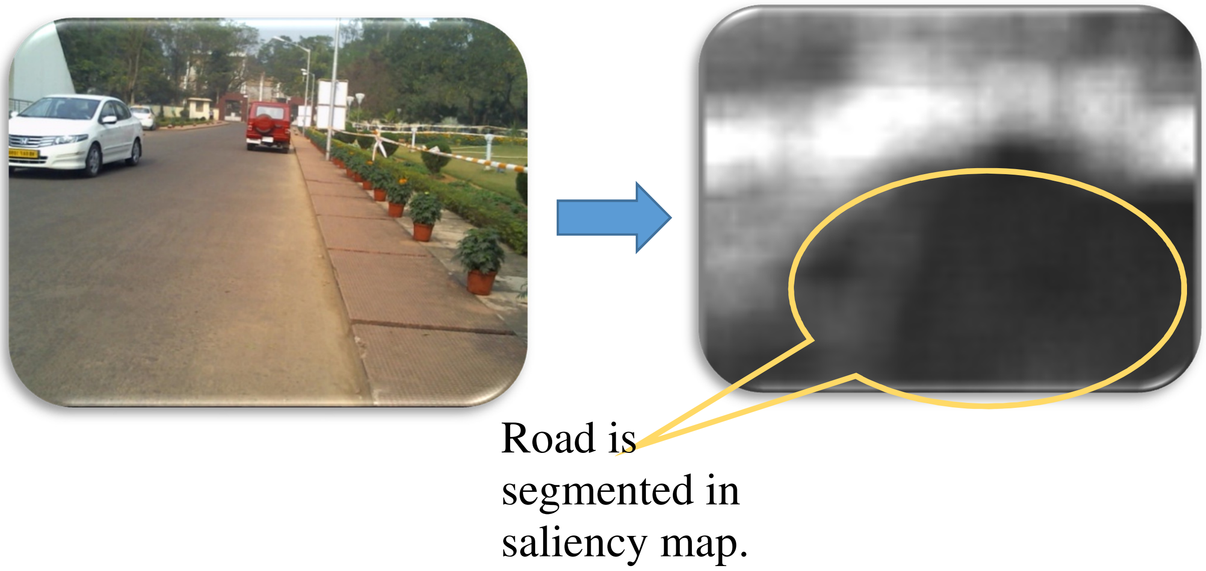
TRAINING PARAMETERS

- *Algorithm used*: BACKPROPOGATION
- *Maximum number of Iterations*=1000;
- *Target error in weights*=0.00001
- *Activation function*: SIGMOID_SYM (α=0.6, β=0.4)
- *Strength of the weight gradient term (bp_dw_scale)* =0.1
- *Strength of the momentum term (bp_moment_scale)* =0.1

INPUT FEATURES TO NEURAL NETWORK

a) Saliency Map

Saliency maps are a successful and biologically plausible technique for modelling visual attention. Any visual scene can be best analysed by paying more attention to fixate or salient locations in a scene. Typically these algorithms produce maps that assign high saliency to regions with rare features or features that differ from their surroundings. What constitutes "rare" varies across the models. One way to represent rare features is to determine how frequently they occur. By fitting a distribution P (F), where F represents image features, rare features can be immediately found by computing $P(F)^{-1}$ for an image.



SALIENCY MAP IMPLEMENTATION

Method 1: Difference of Gaussians filters

Let r, g and b denote the red, green, and blue components of an input image pixel. The intensity (I), red/green (RG), and blue/yellow (B) channels are calculated as

$$I = r + g + b, \qquad RG = r - g, \qquad BY = b - \frac{r+g}{2} - \frac{\min(r,g)}{2}.$$

The DoG filters are generated by

$$DoG(x,y) = \frac{1}{\sigma^2}\exp\left(-\frac{x^2+y^2}{\sigma^2}\right) - \frac{1}{(1.6\sigma)^2}\exp\left(-\frac{x^2+y^2}{(1.6\sigma)^2}\right).$$

where (x,y) is the location in the filter. These filters are convolved with the intensity and colour channels (I, RG, and BY) to produce the filter responses. We use four scales of DoG (σ= 4, 8, 16 or 32 pixels) on each of the three channels, leading to 12 feature response maps.

To parameterize this estimated distribution for each feature F, we used an algorithm proposed by Song (2006) to fit a zero-mean generalized Gaussian distribution, also known as an exponential power distribution, to the filter response data

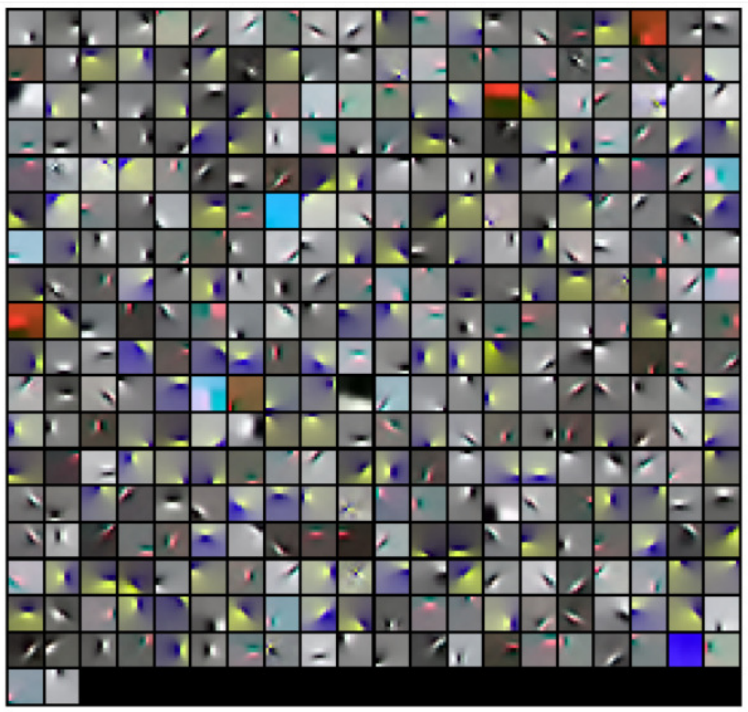
$$p(f;\sigma,\theta) = \frac{\theta}{2\sigma\Gamma(\frac{1}{\theta})}\exp\left(-\left|\frac{f}{\sigma}\right|^{\theta}\right).$$

Ѓ is the gamma function, θ is the shape parameter, σ is the scale parameter, and f is the filter response. . Hence the total bottom-up saliency of a point takes the form:

$$\log s = -\log p(F = f) = \sum_{i=1}^{12}\left|\frac{f_i}{\sigma_i}\right|^{\theta_i} + const.$$

Method 2: Linear ICA Filters

The features learned using DoG filters are not independent of each other.To overcome this drawback, we use Independent Component Analysis. FastICA algorithm (Hyvarinen & Oja, 1997) is applied to 11-pixel x 11-pixel colour natural image patches drawn from the Kyoto image dataset (Wachtler, Doi, Lee, & Sejnowski, 2007). This resulted in 11*11*3 – 1 = 362 features.The ICA feature responses to natural images can be fitted very well using generalized Gaussian distributions, and we obtain the shape and scale parameters for each ICA filter by fitting its response to the ICA training images.

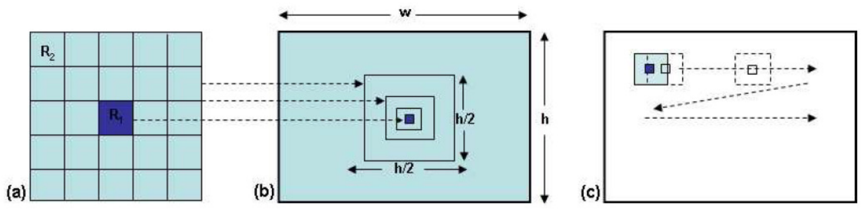


The Learned ICA Filters

b)Contrast based Saliency Map

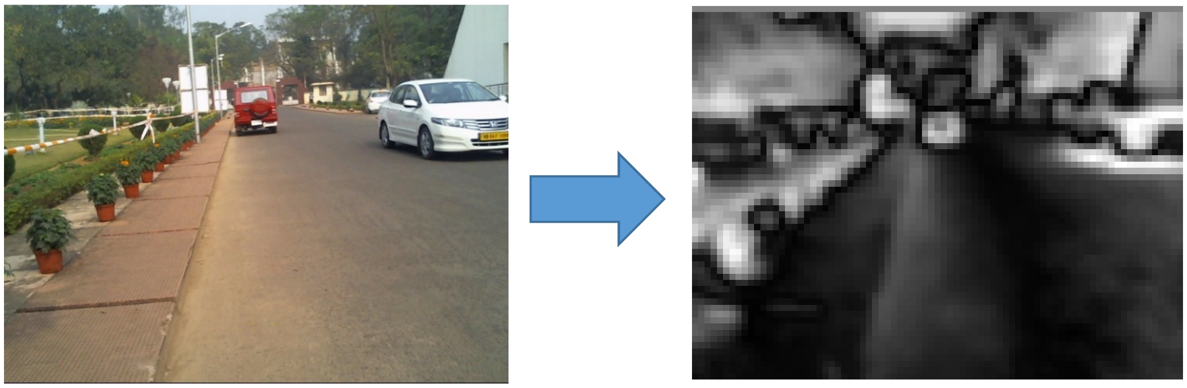
Saliency is determined as the local contrast of an image region with respect to its neighbourhood at various scales. This is evaluated as the distance between the average feature vectors of the pixels of an image sub-region with the average feature vector of the pixels of its neighbourhood. This allows obtaining a combined feature map at a given scale by using feature vectors for each pixel, instead of combining separate saliency maps for scalar values of each feature. At a given scale, the contrast based saliency value c_{ij} for a pixel at position (*i, j*) in the image is determined as the distance *D* between the average vectors of pixel features of the inner region *R*1 and that of the outer region *R*2.

$$c_{i,j} = D\left[\left(\frac{1}{N_1}\sum_{p=1}^{N_1}\mathbf{v}_p\right),\left(\frac{1}{N_2}\sum_{q=1}^{N_2}\mathbf{v}_q\right)\right]$$



$$c_{i,j} = \left\|\mathbf{v}_1 - \mathbf{v}_2\right\|$$

For CIELab colour space, the distance D is approximately Euclidian.The final saliency map is obtained as the sum of saliency values across scales.



Other features fed to neural network are

c) **Hue**:

The "attribute of a visual sensation according to which an area appears to be similar to one of the perceived colours: red, yellow, green, and blue

d) **Saturation**:

The "colorfulness of a stimulus relative to its own brightness".

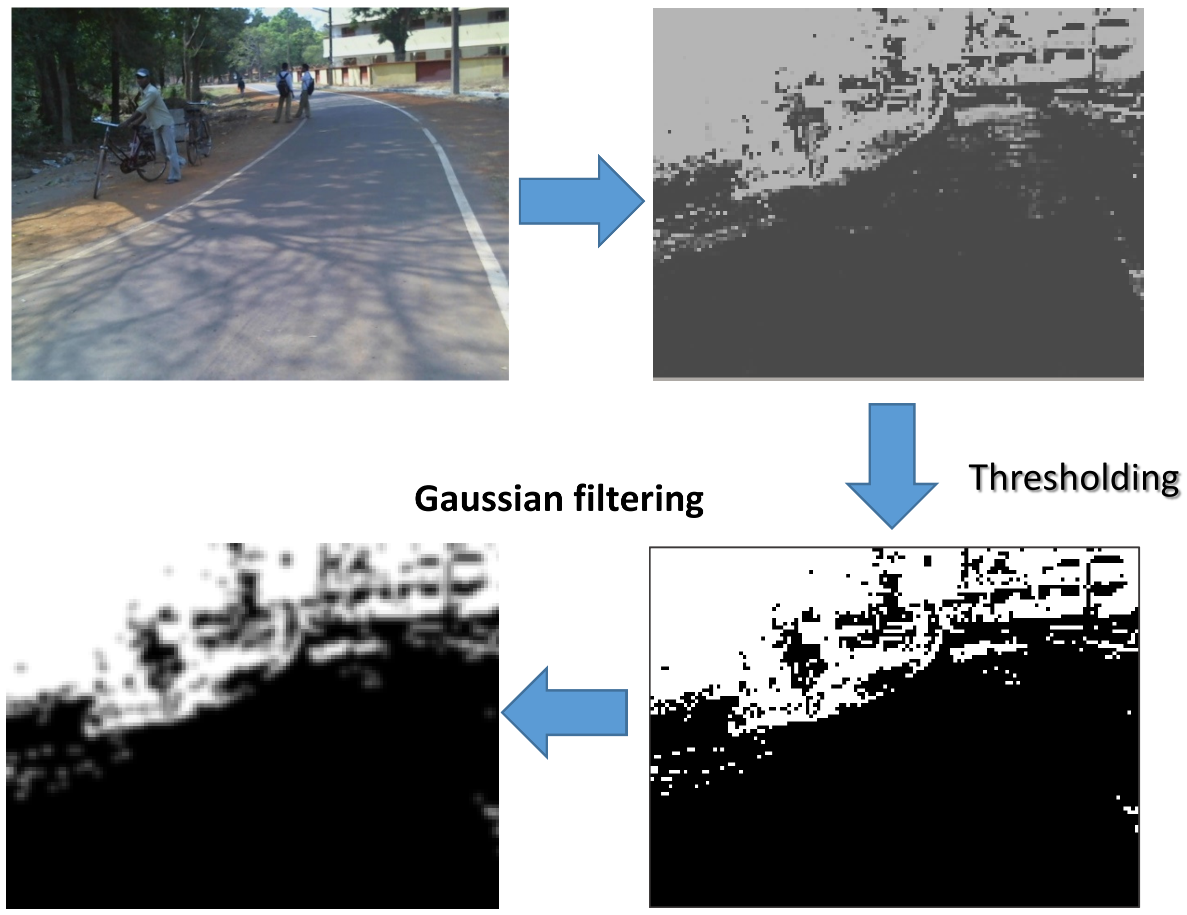
e) **Intensity**:

The total amount of light passing through a particular area.

OpenCV Implementation

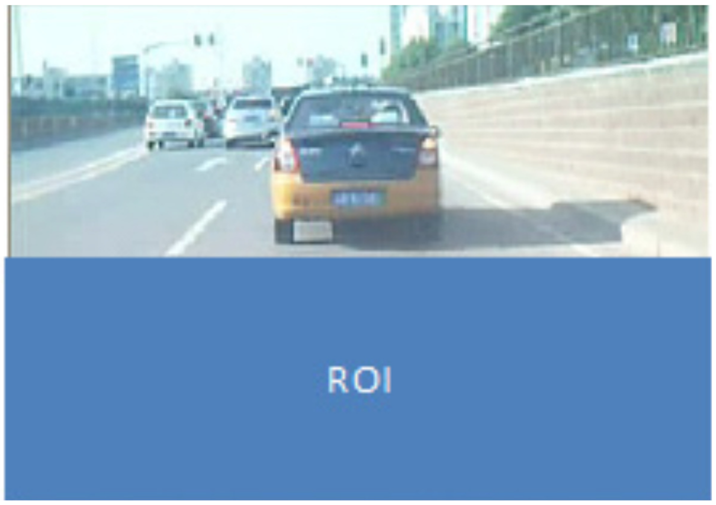
- Each image is loaded and rescaled from 480x640 to 100x133.
- All 5 features corresponding to a pixel of particular image are loaded into single row of Matrix ‘training_set’.
- The number of rows is total number of pixels in all the images.
- The binary pixel value corresponding to ground-truth is stored in matrix ‘training_set_classifications’.
- The neural network is trained and saved into .xml file which can later be used for prediction of output for a given input.

RESULTS



ROAD WIDTH AND OBSTACLE DETECTION

(without machine learning)



A Region of Interest (ROI) is selected as bottom half of the image. Among all the pixels present in ROI, pixels are filtered by following criteria:

1. Road colours are mainly gray, white or luminous yellow.
2. Road colours usually are not red or green.

$$P_f = \left\{ p \;\middle|\; \begin{array}{l} r_p - \max(g_p, b_p) < T_{red} \\ g_p - \max(r_p, b_p) < T_{green} \\ \frac{r_p + g_p + b_p}{3} > T_{gray} \\ p \in P_{all} \end{array} \right\}$$

Co-ordinates of eligible pixels are stored in vector 'P_f '.

where, (r_p, g_p, b_p) is the color value of pixel *p*. *Tred, Tgreen, Tgray* are thresholds chosen empirically based on the above priori knowledge to exclude non-road pixels from P_f as many as possible.Then, we calculate the distribution of each pixel in P_f on *RG* plane. Count the appearance *time*(*r,g*) of each color value (*r, g*) as: *time*(*r,g*) = *count*((r_p, g_p) $\in P_f$) where $r_p = r$ and $g_p = g$.

Probability Map Calculation

The biggest value among all $time_{(r,g)}$ is denoted as $time_{max}$. The closer a colour's count is to $time_{max}$, the more likely it stands for the road region. Hence, we regard a pixel p as road point if its $time_{(rp,gp)}$ are no less than $k \times time_{max}$ and define its probability of being road region $prob_p$ as 1. For other pixels, $prob_p$ is computed with the following formula:

$$prob_p = \frac{time_{(r_p,g_p)}}{k \times time_{max}}$$

In our implementation, value of k=0.7.Pixels with $prob_p$ =1.0 are added in another set P_l.

Tracking prediction

The probability map is updated each time a new frame is read and contains equal contribution from all frames at any instant. The pixels contained in set P_l are used as seed points for region growing segmentation.

Region Growing Segmentation

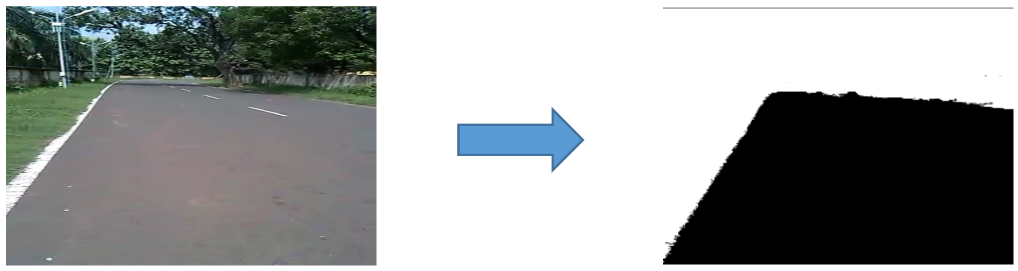
For each seed point belonging to P_l, we compare it with the neighbouring pixels (in 8-connectivity) using certain similarity rules. Pixels satisfying the requirement are added to list of seed points. For implementation, Breadth First search (BFS) is applied to each seed point using a queue. The combination of pixels visited by each seed point constitute the segmented region.

Our similarity rules with intensity feature representing likelihood between seed point *s* and its neighbour *neighbor*(*s*) are as follows:

$$\begin{cases} \left| I_s - I_{neighbor(s)} \right| < \gamma \\ \left| I_{neighbor(s)} - \mu_n \right| < \sigma_n \end{cases}$$

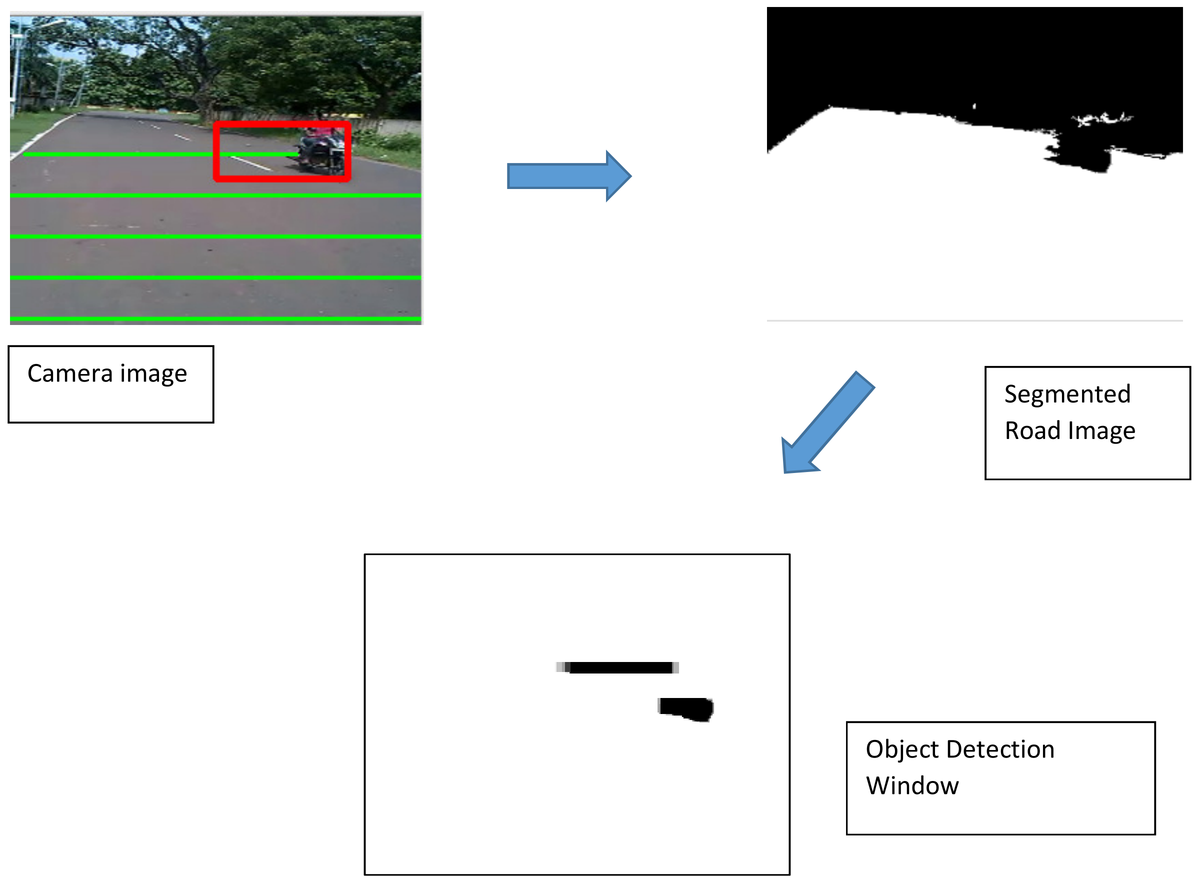
μ_n and σ_n can be obtained by maximum likelihood method as follows:

$$\begin{cases} \mu_n = \frac{1}{m-1} \sum\limits_{p \in \Omega} I_p \\ \sigma_n = \frac{1}{m-2} \sum\limits_{p \in \Omega} (I_p - \mu_n)^2 \end{cases}$$



Obstacle Detection

Once path is thresholded out, only objects within the path is considered as obstacle. Hence, anything out of the path detection window is turned into white pixels with obstacle only as black. Scanning is done in x-direction and y-direction, through every row and column to obtain the presence, the number, the size and the position of the obstacles.



Pseudo code for obstacle detection

```
for (i=image height;i>=0;i--)

for (j=0; j<image width;j++)

{

        if (image(i,j) ==0) image(i,j)=1;

        if (image(I,j) ==1) break from inner loop;

}

for (j=image width;j>=0;j--)

{

        if (image(i,j) ==0) image(i,j)=1;

        if (image(I,j) ==1) break from inner loop;

}
```

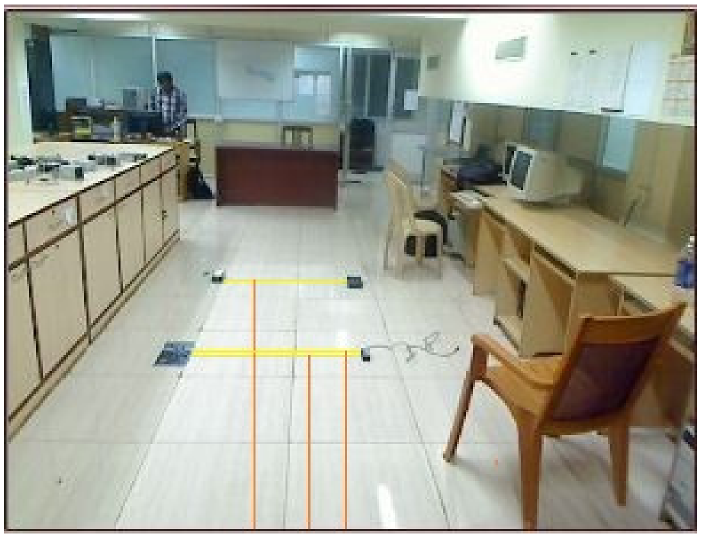
```
while (until an object is detected completely and no. of objects < 10)
{
z=20;
for (j=z+5;j<(image width-25);j++)
for (i=70;i<image height;i++)
{
        if (image(i,j)==0) record the left, right, top and bottom extremity
of the total blob.
}
}
if (object is found out) z=right extremity of previous object;
cvRectange(image,(left extremity, top extremity),(right extremity,
bottom extremity));
width in pixels = right extremity – left extremity;
height in pixels = bottom extremity – top extremity;
Calculate width and height in centimeters and display them;
        }
```

Distance Calibration

If the width of the path and the width of the obstacle are found out, the safe portion of the path for navigation can be easily found out. For distance calibration, 2 pairs of markers are kept at a distance of 1 m each at different distance from the camera. Considering an average user height of 5 feet 7 inch and the camera being at a total horizontal position, the image shown below is taken. Then the distance between the markers (D_1, D_2, D_3) are measured in pixels. Followed by that, the distance (A_1, A_2, A_3) are also measured in pixels. What was found out was that for all i, A_i * D_i = constant = 9000. Hence in this way, it was concluded that if an object of width of D pixels is at a distance of A pixels from the bottom of the image frame, the width of the object is tentatively given by the equation,

Width in meters = a*d/9000

<u>Width in centimeters = a*d/90</u>



Setup for distance calibration

DIRECTION PREDICTION

Using the knowledge of road width and width of obstacles present, the system can guide the user to walk safely by following the given directions. For each frame, the leftmost boundary among all obstacles and the rightmost one are calculated. If the rightmost boundary lies to the left of midline (assumed to be walking path), the person is directed to right. When the leftmost boundary lies to the right of midline, the person is directed to left. If no obstacle is present, the leftmost boundary is initialised greater than rightmost boundary, so the person should move forward. Else the person is directed to wait till the path gets cleared of obstacles.

References

1. Lingyun Zhang, Matthew H. Tong & Garrison W. Cottrell (2009) *SUNDAy*: *Saliency Using Natural Statistics for Dynamic Analysis of Scenes*
2. Yuan Gao, Yixu Song, and Zehong Yang *A Real-Time Drivable Road Detection Algorithm in Urban Traffic Environment*
3. Sebastian Montabone , Alvaro Soto *Human Detection Using a Mobile Platform and Novel Features Derived From a Visual Saliency Mechanism*