Modelling and Implementation of Real time Texture Detection module for Mobility Assistant for Visually Impaired System (MAVI)

by

Prerna Agarwal

Under the Supervision of

Dr. Chetan Arora, IIIT Delhi
Prof. M. Balakrishnan, IIT Delhi

Indraprastha Institute of Information Technology, Delhi

July, 2017

Keywords: LBP, GLCM, Computer Vision, SegNet, ZedBoard

Modelling and Implementation of Real time Texture Detection module for Mobility Assistant for Visually Impaired (MAVI) System.

Prerna Agarwal
IIIT-D-MTech-CS-DE-15-045

Submitted in partial fulfillment of the requirements
for the Degree of M.Tech. in Computer Science

to

Indraprastha Institute of Information Technology Delhi

July, 2017

Certificate

This is to certify that the thesis titled **"Modelling and Implementation of Real time Texture Detection for Mobility Assistant for Visually Impaired (MAVI) System"** submitted by **Prerna Agarwal** to the Indraprastha Institute of Information Technology Delhi, for the award of the Master of Technology, is an original research work carried out by her under our supervision. In our opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree. The results contained in this thesis have not been submitted in part or full to any other university or Institute for the award of any degree/diploma.

Dr. Chetan Arora

Dept. of Computer Science

IIIT Delhi

Prof. M. Balakrishnan

Dept. of Computer Science

IIT Delhi

Abstract

As the volume of video data is increasing day by day, real-time video processing is becoming an important application to build any real time device based on image processing. An interesting task that is crucial for the safety of a person is safety from moving vehicles. This can be a crucial module for the people who are interested in developing an assisting device for visually impaired that can help in identifying the type of surface a person is walking on.

The thesis presents the design space exploration, implementation of texture detection module for MAVI (Mobility Assistant for Visually Impaired), an outdoor navigation system in Indian context, for helping visually impaired people by making them aware of the surface on which they are walking on. It will help them in identifying the sidewalks (pavement) to ensure safety from moving vehicles on a road.

The work involves exploring various models which will work in real time, based on different texture features like LBP and GLCM and using Computer Vision techniques, measuring their accuracies and performance by cross compiling their code on ZedBoard. It also explores the usage of deep learning model like SegNet for this task and its performance in real time. The major challenge for such a module is to perform well in a lot of diversity and complexities that arise in the Indian scenario due to non standard practices, therefore its scope is limited to three major texture classes i.e., road, pavement and grass.

Acknowledgements

I sincerely thank my supervisor Dr. Chetan Arora, IIIT Delhi for giving me the opportunity to work on this project and also for his constant supervision and valuable guidance during the course of thesis. I would also like to thank my co-supervisor Prof. M. Balakrishnan, IIT Delhi for his valuable insights and assistance. I extend my gratitude to Mr. Rajesh Kedia, IIT Delhi for providing us with his ideas and motivation.
A special thanks to my fellows, Richa Verma and Saurabh Agarwal for making my thesis journey fun and memorable and also for their help in cross compilation and profiling on Zedboard.
I would also like to thank my mother for providing me the motivation, support and encouragement during the entire duration of my thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview and Problem definition (MAVI)

The objective of MAVI System is to conceptualize and design a smart-camera based vision system, capable of extracting useful information, for example, faces, texture, signboard information from captured images.

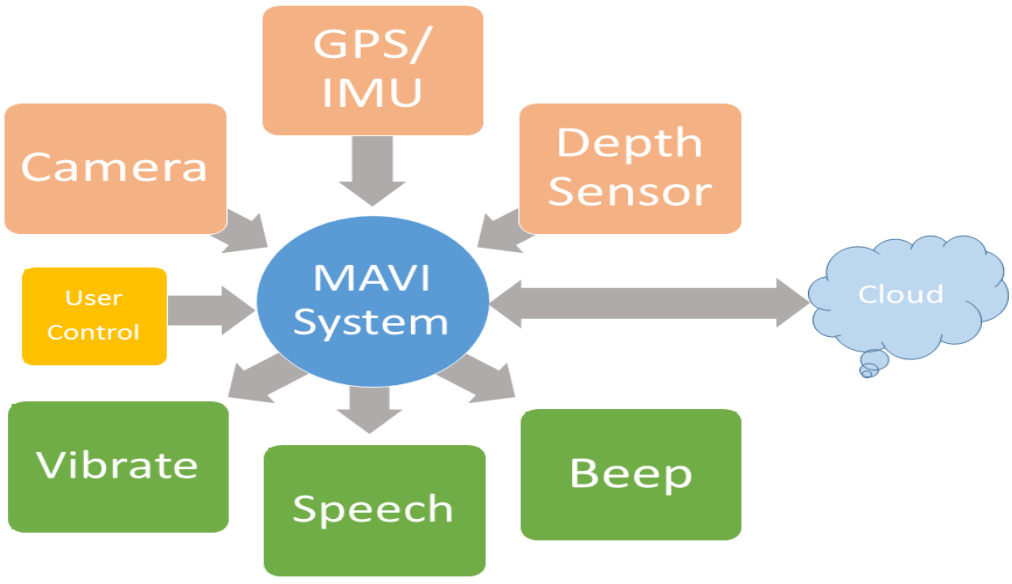

Figure 1.1: MAVI system

It consists of different modules integrated onto one platform, communicating with the user through a mobile interface and using cloud to translate the coordinates sent by Localization module into navigational information in a frame captured by the camera. The modules in MAVI System consist of Face Detection and Recognition, Texture Detection, Signboard Detection, Animal Detection and Localization [60,65].

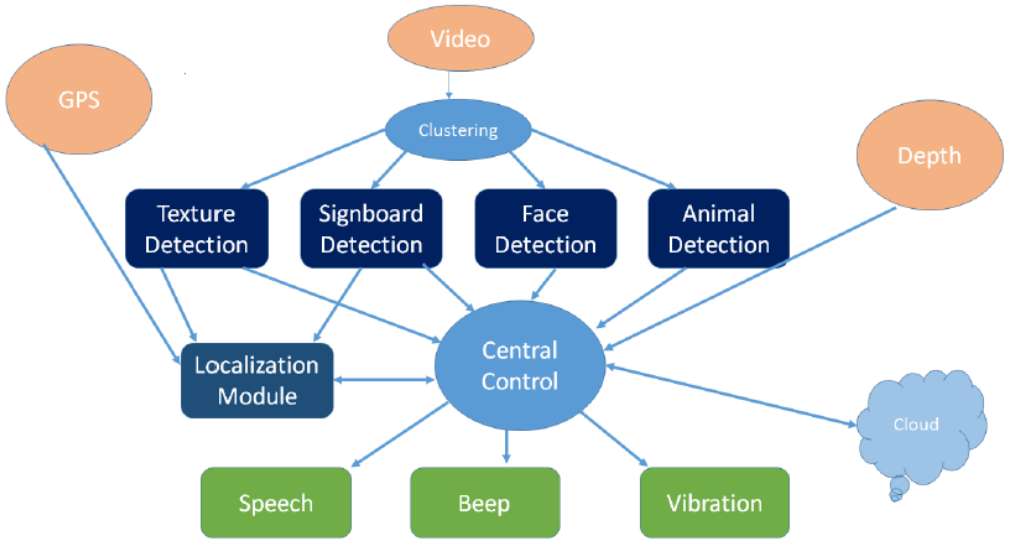

Figure 1.2: MAVI System and its modules

The objective of the texture detection module is to implement a real-time, online, energy-optimal texture classification algorithm by using texture features and Computer Vision techniques and further deep neural network techniques. The scope of this problem is limited to classify the texture in 3 classes i.e., road, pavement and grass.

1.2 Motivation

Visual impairment is a type of disability in which the sufferer needs continuous assistance for living. Traditional white cane has been used by a large number of blind people since decades despite several challenges. But white cane has its own limitation. It has a short range of detection, can detect

obstacles below certain height only. It also doesn't provide geographical information about its surrounding. Social inclusion, safety from stray dogs, cows (common in India), navigation are few of the several limitations of the traditional white cane. Moreover, many unmet needs with most globally available solutions are unaffordable. Many computer vision based visual aids are available but the problem with such solutions is affordability, acceptability, and usability. Computer-vision-based systems provide substantial advantages and help in addressing such limitations. Thus we would like to propose a cheap globally acceptable solution for to overcome the above-listed limitations traditional white cane.

1.3 Texture

Images generally consist of distinct parts representing different surfaces in the scene. Surfaces made of an optically homogeneous material and under smoothly varying illumination give rise to image parts with a smooth variation of image brightness, referred to as non-texture sub image. In contrast, surfaces with discontinuities in depth, material, illumination, etc., give rise to texture sub images. The spatial variation of intensity in each image part contains different information about the scene, often requiring different types of algorithms to be invoked on these parts. Depending on its purpose, an algorithm may apply to either texture or non-texture sub images, or both. For example, if a surface in the scene gives rise to the texture sub image it may be better to estimate the surface's shape by shape-from-texture algorithms than by shape-from-shading methods. Also, in order for image smoothing and compression algorithms to be applicable to the entire image they should account for differences among textured and non-textured image parts, and adjust their parameters accordingly.

1.4 Challenges

When choosing a texture feature for real time scenario, a number of aspects should be considered:

- *Illumination (gray scale) invariance*: how sensitive the algorithm is to changes in gray scale. This is particularly important in outdoor

environment, where lighting conditions may vary.

- *Spatial scale invariance*: Does the algorithm work, if the spatial scale of unknown samples to be classified is different from that of training data.

- *Rotation invariance*: Does the algorithm work, if the rotation of the images changes with respect to the viewpoint.

- *Projection invariance (3-D texture analysis)*: in addition to invariance wrt. spatial scale and rotation the algorithm may have to cope with the changes in tilt and slant angles.

- *Robustness w.r.t. noise*: how well the algorithm tolerates noise in the input images.

- *Robustness wrt. parameters*: the algorithm may have several built-in parameters; is it difficult to find the right values for them, and does a given set of values apply for a large range of textures.

- *Computational complexity*: many algorithms are so computationally intensive that they can not be considered for applications with high throughput requirements, e.g. real-time visual inspection and retrieval of large databases.

- *Window/Sample size*: how large sample the algorithm requires to be able to produce a useful description of the texture content.

1.5 Contributions

The main challenge to build up a real time module for MAVI which performs well in Indian context due to the non standard practices.
The key contributions are:

- Creating and benchmarking different texture dataset, titled Texture Version 1.1 for broader view of the scenario.

- Creating and benchmarking different texture dataset, titled Texture Version 1.2 for granular view of the scenario.

- Exploring various features and models to solve the problem as classification task into 3 main classes i.e., road, pavement and grass
- A hardware model which ports the GLCM features based algorithm for texture classification which works in real time and overcomes the major challenges.
- An android based application which ports the GLCM features based algorithm for texture classification which works in real time and overcomes the major challenges.
- A software solution to the problem using deep neural network (due to the limitation of it being ported on hardware).

1.6 Texture Classification

Texture Classification is the problem of distinguishing between different textures. Since many very sophisticated classifiers exist, the key challenge here is the development of effective features to extract from a given textured image. Many approaches have been defined to extract features, such as Gabor filters or wavelets, however a great deal of recent work has focused on patch-based methods, whereby a texture is classified strictly based on a set of small patches of pixels extracted from a given textured image.

The difficulty is that too small a patch fails to be sensitive to non-local features of the texture, whereas a very large patch leads to enormous feature vectors and computational problems associated with very high-dimensional pattern recognition.

1.7 Database

The data was collected in IIT campus and outside the campus to induce variety and complexity. The dataset has the following variations:

- Angle of capture
- Lighting conditions
- Types of pavement

- Noisy blocks
- Shadow

Texture Version 1.1 (Block size- 160*100)

This version is for the broader view of the scenario. The following amount of database was collected and annotated:

- Total number of images: 1518
- Total number of blocks: 13,662
- Total number of images used for training: 1184
- Total number of blocks used for training: 10656 (approximately 2500 of each class)
- Total number of images used for testing: 334
- Total number of blocks used for testing: 3006 (approximately 750 of each type)

Texture Version 1.2 (Block size- 40*40):
This version is for the granular view of the scenario. The following amount of dataset was annotated:

- Total number of images: 200
- Total number of blocks: 16,000
- Total number of images used for training: 150
- Total number of blocks used for training: 12000 (approximately 4000 of each class)
- Total number of images used for testing: 50
- Total number of blocks used for testing: 4000 (approximately 1000 of each type)

1.8 Hardware choice for Prototype: ZedBoard

ZedBoard is based on Zynq All-Programmable SoC (AP SoC). This product integrates a feature-rich dual-core ARMCortex- A9 MPCore based processing system (PS) and Xilinx programmable logic (PL) in a single device, built on a state-of- the-art, high-performance, low- power process technology. It is a joint venture by Xilinx (Zynq AP SoC), Digilent (board manufacturer) and Avnet (distributor) .

- Xilinx Zynq XC7Z020-1CSG484 EPP (extensible processor platform) containing Dual-core ARM Cortex-A9 (PS) running at 666 MHz and Artix-7 FPGA (PL)
- Memory: 512 MB DDR3 and 256 Mb QSPI Flash
- On-board oscillators: 33.333 MHz (PS), 100 MHz (PL)
- Interfaces like USB-JTAG Programming, Ethernet, USB OTG, USB-UART,SD Card, Pmod, LEDs, Dip switches, Push buttons etc.
- Display: HDMI output, VGA (12-bit colour)
- Power: 12 V @ 5A AC-DC regulator

Figure 1.3: ZedBoard

1.8.1 Linaro

For the software-only implementation, Linaro Ubuntu distribution was used. Linaro is an open-source complete Linux distribution based on Ubuntu. It supports graphical desktop through on-board HDMI port. For booting from SD card, Linaro less system has to be placed in different partition other than the partition where Kernel image,devicetree reside. It is a persistent OS, i.e. all changes are written to memory and it saves files after reboot or shutdown. OpenCV(version 3.1) was built on top of it. For booting Linux, 8 GB, class-10 SD card was used.

1.8.2 Cross Compilation for ZedBoard

Arm toolchain provided for linux ARM-linux-gnueabihf was installed and OpenCV 3.1 was source compiled using this tool chain for Linaro based systems. OpenCV was compiled both as a shared library and static libraries , stripped version of OpenCV was also compiled . The same tool chain is use to compile the application for Zedboard.

1.8.3 Pivot Head Camera

Pivot head camera is a wearable camera device, which will be used to stream live data and store it. Control system receives these frames and passes it further to different modules after doing some basic image processing task. All the modules execute predefined algorithm to extract required information from the received image. The extracted information is transmitted to the control system. Control system combines all the received information into a packet and transfers this information to the mobile Android application via the bluetooth protocol. The mobile application receives this information and informs user through the audio output.

Figure 1.4: Pivot Head Camera

1.9 Thesis Outline

In the next chapter, the background and its limitations is being discussed. In chapter 3, Gabor filter based algorithms are being discussed. In chapter 4, LBP features based algorithm, along with its hardware porting with its plots are being discussed. In chapter 5, GLCM and its features are discussed along with its porting on hardware and Android application. In chapter 6, a deep neural network approach, SegNet is being discussed. Finally, in the last chapter, conclusions along with the work is summarized with the future work.

Chapter 2

Background

The previous literature in this domain was surveyed keeping an eye on three different view points:

2.1 Safety on Road

From the point of view of safety on road, there has been a lot of work in the past. In [1] authors have presented a computer vision system whose aim is to detect and classify cracks on road surfaces. They first find the region of interest(ROI) using Hough Transform and then use Hough transform features (HTF) and local binary pattern (LBP) to identify the cracks in the road surface. They took an advantage of the fact that most of the roads can be characterized by a particular texture pattern. In [2] a multi resolution pyramid is built using morphological operators. Then the Co-Occurrence matrix and one local threshold are computed for every scale to detect the cracks on roads. This solution is highly dependent on the set of pairs chosen to build the Co-Occurrence matrix. In [3] Local Binary Pattern is combined with other techniques to classify the road pavement. In [4] a solution is presented in which, the image is divided in ROIs and depending on the intensity values of its neighborhood, a binary pattern is built. The shape of that pattern is studied to determine which class each patch belongs to. The limitation of this approach is its inability to define accurately the lines that compose the detected cracks. In [5] the authors have developed a system AMAC [6] for characterization of roads in France, one of its functions is acquisition of road images in high resolution and independently of lighting conditions. Data

acquisition is performed with two line scan cameras mounted on the vehicle with an angle of approximately 30 degrees, the road surface is illuminated by two laser illuminators , thus the time of day, varying light conditions and shadows over the road have almost no effect on the quality of the acquired data. Although, these kind of techniques cannot be used for a real time application which is required to be energy and cost efficient to be used by a large number of people.
Most of the previous works consisted of complex and expensive acquisition systems to identify road problems, whereas there is a need of a simple system composed of a single camera mounted and no additional image enhancing devices. In [7] authors have presented a new approach for defect detection using linear FIR filters with optimized energy separation. They have also evaluated the performance of different feature separation criterion with reference to fabric defects. In an early study [8], Tamura and co-authors introduced a set of texture features, including regularity, which were designed to correspond to human perception. Their regularity feature was defined via the spatial variation of four other features, such as coarseness and directionality. However, experiments indicated poor correlation between the feature and the human rankings. So, in [9] the author's approach exploits pairwise (second order) statistics of an image, such as the co-occurrence matrix and the greylevel difference histogram (GLDH) [10]. These standard tools of texture analysis describe statistical interactions between pixels separated by different spacing vectors. Traditionally, most model based and heuristic approaches to texture have used local neighbourhoods formed by a few small spacings. The dependence on the spacing vector, that is, the long-range spatial interactions have not been analysed.

2.2 Texture Classification

In the view of texture classification, a lot of classification techniques based on different texture features have been proposed in the past. A texture classification problem consists of different determining textures present in an image given as a set of texture patterns of interest. Many texture classification problems usually require the computation of a large amount of texture features in order to characterize their associated patterns. This implies that texture classifiers frequently combine big sets of features without taking into account their relevance and redundancy. Thus, lowering the dimensionality

of a feature set is necessary for preserving the most relevant features and reducing the computational cost derived from unnecessary features that do not contribute to increase the quality of the available information for each class [11, 12, 13]. As images of the same underlying texture can vary significantly, textural features must be invariant to (large) image variations and at the same time sensitive to intrinsic spatial structures that define textures. As there is no obvious feature common for all texture images, texture features are often proposed based on assumptions for mathematical convenience or task-specific heuristics [14, 15]. Existing texture analysis methods are broadly divided into two categories: statistical methods and structural methods [15, 16, 17]. The traditional statistical approaches to texture analysis such as Gray level Co-occurrence Matrices (GLCM), second order statistics, Gaussian Markov Random Fields (GMRF), local linear transforms [18, 19, 20, 21, 22, 23, 24, 25, 26, 27], and run length matrix [28] are restricted to the analysis of spatial interactions over relatively small neighborhoods on a single scale. As a consequence, their performance is best for the analysis of micro-textures only [27]. A number of techniques incorporating invariance with respect to both spatial scale and rotation have been presented [29, 30, 31, 32, 33, 34], but there exist only a limited number of examples of successful exploitation of texture. The local binary pattern (LBP) operator, first introduced by Ojala et al. [17], is a robust but theoretically and computationally simple approach. It brings together the separate statistical and structural approaches to texture analysis of both stochastic micro textures and deterministic macro textures simultaneously. Later, the original LBP operator [17] has been extended in several ways, such as neighborhoods with different sizes [35], multi-resolution [36], uniform patterns [35], etc. The extended LBP operator can be used for rotation invariant texture classification, and has a wide application, such as texture analysis and classification [37, 38, 39, 40, 41, 42, 35, 43, 44, 45, 46, 47, 48].

Structural approaches consider texture as repetition of some texture primitives often called textons, with a certain rule of placement that determines the nature of periodicity. These rules are described in a dictionary to model possible texture appearances. Often structural based methods are justified using psychophysical studies of the texture perception. Structural and geometrical features are usually more stable in overall illumination changes than statistical features, but rely strongly on primitive detection. The previous work is focussed on building texture clasification approaches, without taking computational time complexity much into consideration. The results in

these works are mainly on the datasets which are artificially synthesized and are ideal in nature. Whereas, the challenge in the task solved in this thesis is energy and computation efficient model which will perform good in real world scenario and under different conditions.

2.3 Deep Neural Networks

In order to achieve a near optimal accuracy, there is a need of deep neural networks which gives better results under different circumstances. Newer deep architectures [49, 50, 51, 52, 53] particularly designed for segmentation have advanced the state-of-the-art by learning to decode or map low resolution image representations to pixel-wise predictions. The encoder network which produces these low resolution representations in all of these architectures is the VGG16 classification network [54] which has 13 convolutional layers and 3 fully connected layers. This encoder network weights are typically pretrained on the large ImageNet object classification dataset [55]. The decoder network varies between these architectures and is the part which is responsible for producing multi-dimensional features for each pixel for classification. Each decoder in the Fully Convolutional Network (FCN) architecture [49] learns to upsample its input feature map(s) and combines them with the corresponding encoder feature map to produce the input to the next decoder. The predictive performance of FCN has been improved further by appending the FCN with a recurrent neural network (RNN) [51] and fine-tuning them on large datasets [56, 57]. The RNN layers mimic the sharp boundary delineation capabilities of CRFs while exploiting the feature representation power of FCN's. They show a significant improvement over FCN-8 but also show that this difference is reduced when more training data is used to train FCN-8. The results, although very encouraging, appear coarse [58]. This is primarily because max pooling and sub-sampling reduce feature map resolution. SegNet fulfills the need to map low resolution features to input resolution for pixel-wise classification. This mapping must produce features which are useful for accurate boundary localization. From a computational perspective, it is necessary for the network to be efficient in terms of both memory and computation time during inference. The ability to train end-to-end in order to jointly optimise all the weights in the network using an efficient weight update technique such as stochastic gradient descent (SGD) [59] is an additional benefit since it is more easily repeatable.

Chapter 3

Gabor filter based Algorithms

3.1 Gabor filters

Gabor filter is a linear filter used for edge detection at different frequency and orientation representations which are similar to those of the human visual system, and they have been found to be particularly appropriate for texture representation and discrimination. In the spatial domain, a 2D Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave [61].
A set of Gabor filters with different frequencies and orientations may be helpful for extracting texture oriented in a particular direction as shown in Figure 2.1.

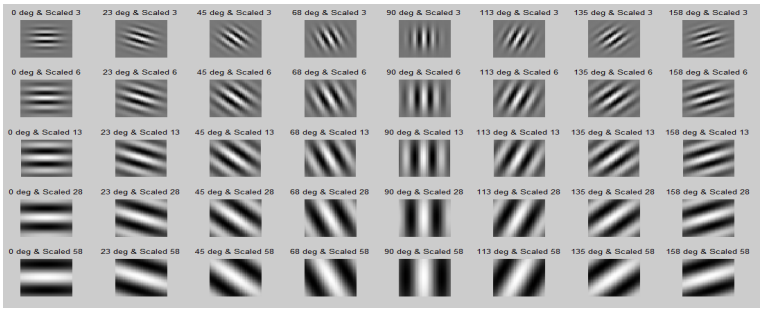

Figure 3.1: Gabor filters with different frequencies and orientation

3.2 Road Extraction using Gabor Filters and k-means

It is an unsupervised method to cluster the image using k-means using Gabor filter features.

3.2.1 Algorithm 1

- **Dataset:** KITTI dataset (1242 * 375)
- **Algorithm**:

  1. To use Gabor magnitude responses as features for use in clustering, some post-processing is required. This post processing includes Gaussian smoothing, adding additional spatial information to the feature set, reshaping feature set to the form expected by the PCA and k-means functions, and normalizing the feature information to a common variance and mean.
  2. Each Gabor magnitude image contains some local variations, even within well segmented regions of constant texture. A sigma is chosen which is matched to the Gabor filter that extracted each feature. When constructing Gabor feature sets for classification, it is useful to add a map of spatial location information in both X and Y. This additional information allows the classifier to prefer groupings which are close together spatially.
  3. Repeat k-means clustering five times to avoid local minima when searching for means that minimize objective function. The only prior information assumed in this algorithm is how many distinct regions of texture are present in the image being segmented.

3.2.2 Limitations

- Computation cost is high due to the necessity of using a large bank of filters.
- Shadow distorts the clusters.

3.2.3 Output

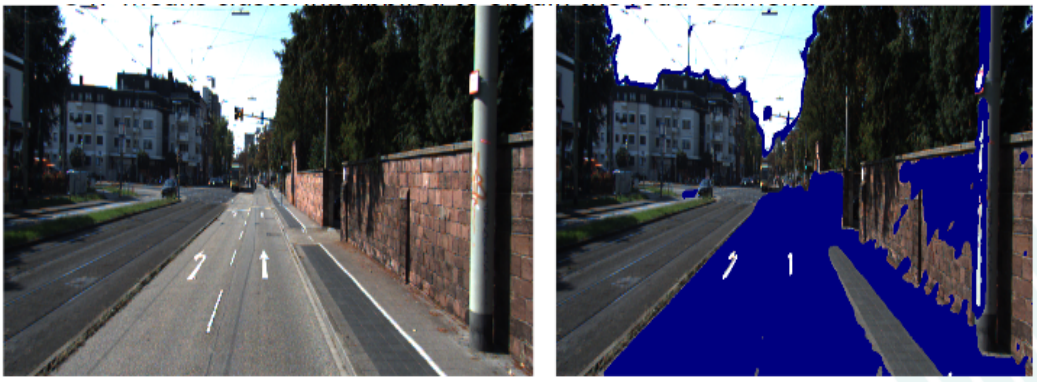

Figure 3.2: Output of Gabor filters to extract road segment

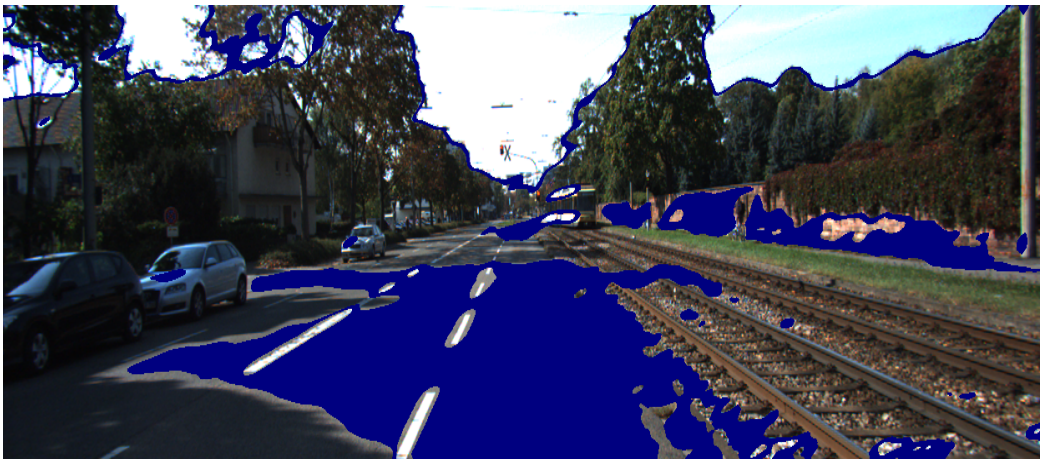

Figure 3.3: Effect of shadow on Gabor filters

3.3 Road Extraction using Gabor filters and SVM

This is a supervised method to predict each pixel of an image belonging to class road.

3.3.1 Algorithm 2

- **Dataset:** KITTI dataset (1242 * 375)
- **Algorithm**:
  - **Training:**
    1. Firstly, the pixel level color and texture features of the image are extracted and they are used as input to the SVM classifier. These features are extracted using the homogeneity model and Gabor Filter.
    2. With the extracted pixel level features, the SVM Classifier is trained by using FCM(Fuzzy C-Means).
  - **Testing:**
    Each pixel is classified as road or non-road segment using SVM classifier.

3.3.2 Limitations

- Time consumption is high because each pixel is classified into its corresponding class.
- Post processing step is required to remove the noise points from output.
- Pixels of shadows are mostly misclassified because the information is very low level to correctly classify them.

3.3.3 Output

The following images depict the output.

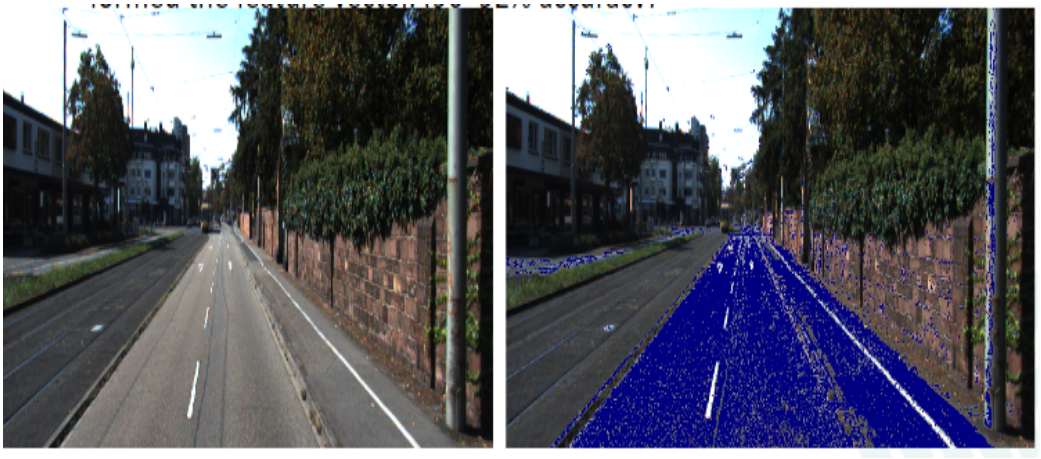

Figure 3.4: Output of SVM to extract road segment

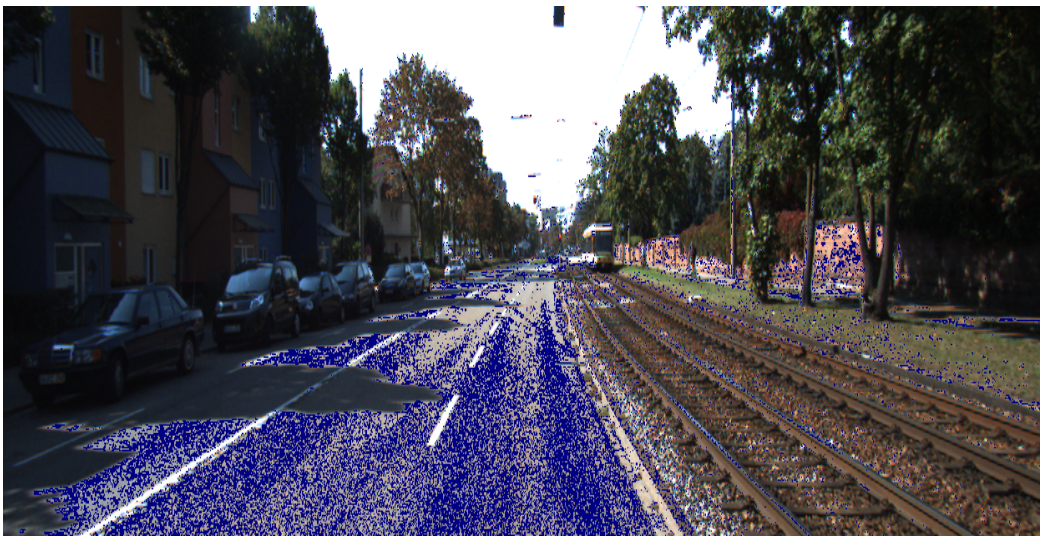

Figure 3.5: Effect of shadow on SVM

3.4 Texture Classification using Gabor filters and SVM

Gabor features will be extracted on the samples of each class and an SVM model will be trained over them.

3.4.1 Algorithm 3

- **Dataset:** Texture Version 1.1
- **Algorithm**:
  - **Training:**

1. *Preprocessing:*
   i) Noise is removed by applying Gaussian filter.
   ii) Contrast is increased so that edges are clear. iii) Image is divided into 3*3 grid blocks.
2. Each filter is convolved with each block to get different representations (response matrices) of the same block where each image gives a feature vector.
3. Response Matrices are converted into feature vectors by any one of the following method:
   i) Local Energy
   ii) Mean Amplitude
   iii) Orientation whose local has the maximum Energy.
4. Extract features of blocks of each class
5. SVM classifier trains Multi-class classifier on the extracted features.

- **Testing:**

1. *Preprocessing:*
   i) Noise is removed by applying Gaussian filter.
   ii) Contrast is increased so that edges are clear.
2. Image of size 640*480 is divided into 3*3 grid and each block is classified into its corresponding class.

3.4.2 Limitations

- Shadows cannot be handled by gabor filters.
- Low accuracy for grass vs road class.
- The time consumption is very high (approximately 2 seconds per frame).

3.4.3 Results

Method	Pavement(%)	Road(%)	Others(%)	Total(%)
P vs NP	87.66	66.5		75.57
hline 3 class	87.66	68.5	87.8	81.6

Table 3.1: Accuracy of classification on Gabor Filters

3.4.4 Output

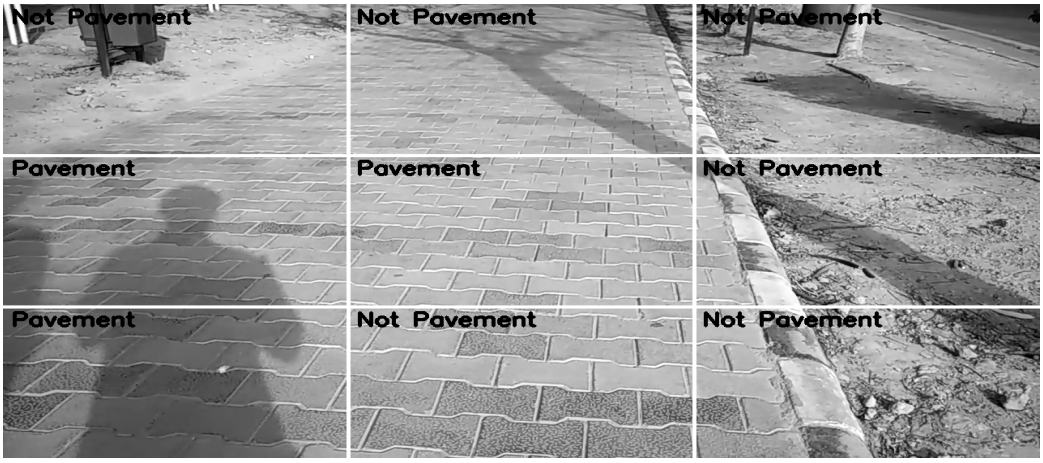

Figure 3.6: Output for Pavement vs Not-Pavement

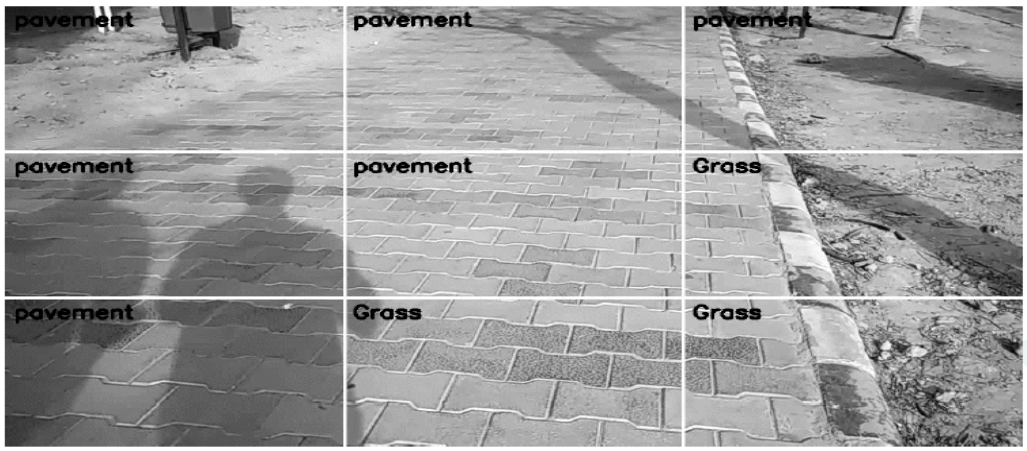

Figure 3.7: Output for 3 classes

Chapter 4

LBP features based Algorithm

4.1 LBP features

Local binary patterns (LBP) is a type of visual descriptor used for texture classification. For each pixel in a cell(which is defined by specifying radius of the circle), it compares the pixel to each of its neighbors. Then it follows the pixels along a circle, i.e. clockwise or counterclockwise. Where the center pixel's value is greater than the neighbor's value, it writes "0". Otherwise, writes "1". This gives a 8 digit binary number which is converted into its decimal value[62].
Histogram is computed over the cell, of the frequency of each "number" occurring. This histogram can be seen as a 256-dimensional feature vector. Normalized histograms are concatenated to give a feature vector for the entire window.

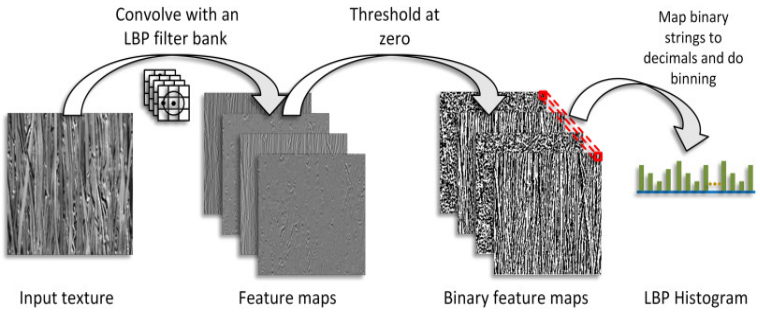

Figure 4.1: LBP feature extraction

4.2 Algorithm

- **Dataset:** Texture Version 1.1, Texture Version 1.2
- **Training:**
  - *Preprocessing:*
    i) Noise is removed by applying Gaussian filter.
    ii) Contrast is increased so that edges are clear.

  Given a training set of block of images obtained by dividing the image by the window size, extract LBP features from them:

  1. Choose number of points as 16 and the radius of the circle to be 2.
  2. Compute Histogram over the block, of the frequency of each number occuring for each window.
  3. Normalized histograms are apended and given as a feature vector to SVM Classifier for training.

- **Testing:**
  1. *Preprocessing:*
     i) Noise is removed by applying Gaussian filter.
     ii) Contrast is increased so that edges are clear.
  2. Image of size 640*480 is divided into 3*3 grid and each block is classified into its corresponding class.
  3. Image of size 640*480 is divided into 40*40 sized blocks and each block is classified into its corresponding class.
  4. The upper rows of the image is discarded because that mainly contains sky and trees.

4.3 Observations

1. Runs in real time i.e., can be ported on hardware.
2. Handles shadows well.
3. Overall performance is acceptable but needs improvements.

4.4 Output

4.4.1 Output: 160*100

Figure 4.2: Output for Pavement vs Not-Pavement for 160*100 block

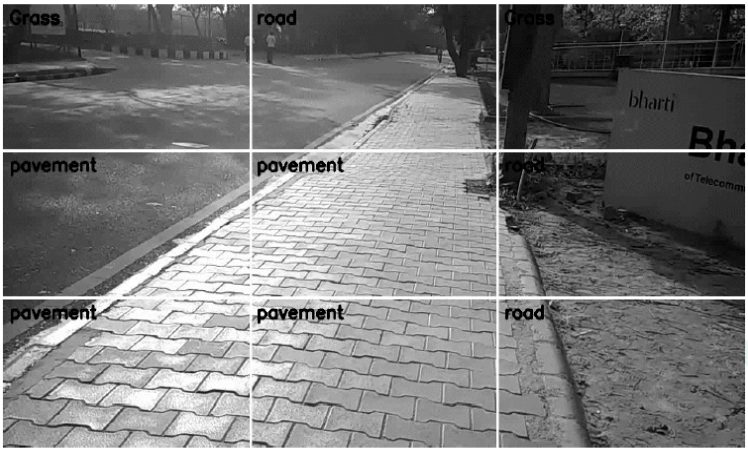

Figure 4.3: Output for 3 class for 160*100 block

4.4.2 Output: 40*40

Red color denotes the detected pavement blocks and blue color denotes the detected non pavement blocks.

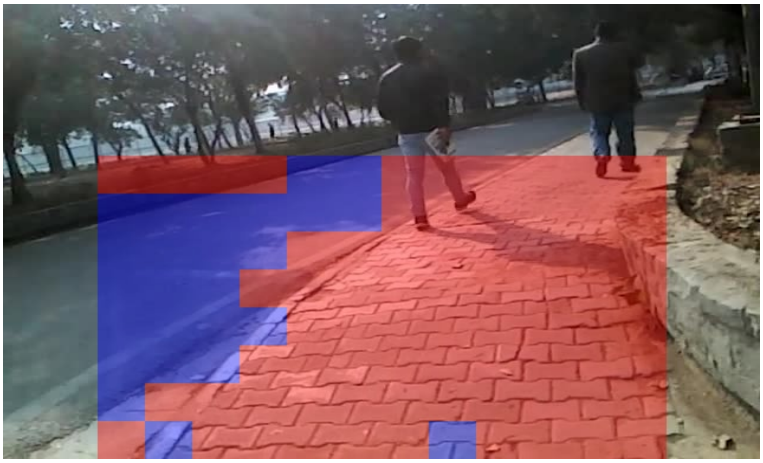

Figure 4.4: Output for Pavement vs Not-Pavement for 40*40 block

Red color denotes the detected pavement blocks and blue color denotes the detected road blocks. Green color denotes the detected 'others' blocks.

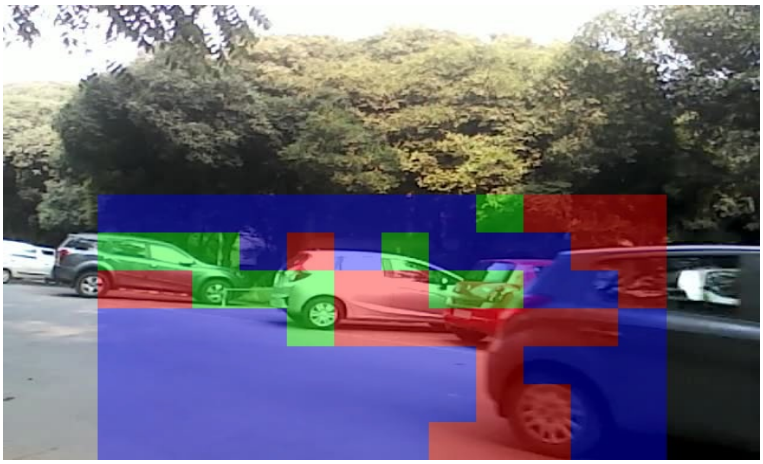

Figure 4.5: Output for 3 class for 40*40 block

4.5 Hardware Porting

- Since, only C++ (OpenCV) could be cross-compiled on the ZedBoard, the entire code was written from scratch in C++ along with feature extraction code to port it on ZedBoard.
- The ZedBoard is approximately 10 times slower than the normal computer.
- Running time on Desktop: 6 frames/sec
- Running time on ZedBoard: 5.2 sec/frame
- CDF of running time of LBP on Zedboard:

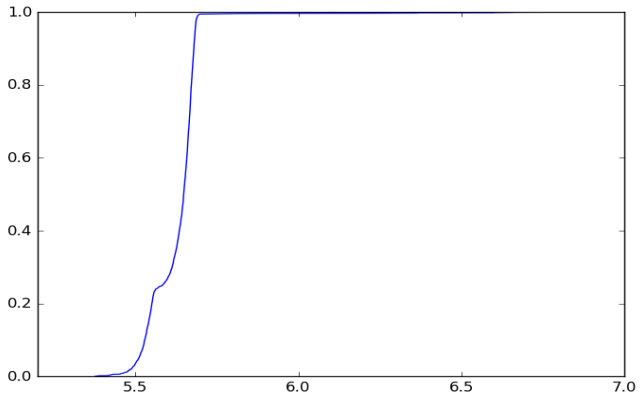

Figure 4.6: CDF for LBP based model

The Cumulative Distribution Function (CDF) plot depicts the time taken to compute LBP features for 40*40 blocks on Zedboard and to predict the class of each block. This plot gives us an estimate of the fraction of frames taking the highest run time. This information can be used to analyze the frames that takes high run time, thus will help in resolving the issue of high run time. The long tail depicts that most of the frames takes similar amount of time to compute LBP features. This is due to the fixed window size for prediction. Although, time taken on few frames to compute LBP features is different which might arise due to ZedBoard being heatup.

4.6 Results

- **Accuracy:**

Method	Pavement(%)	Road(%)	Others(%)	Total(%)
P vs NP	92.33	88.3		90.12
3 class	91.6	92.3	93.8	92.4

Table 4.1: Accuracy of classification of LBP model for 160*100

Method	Pavement(%)	Road(%)	Others(%)	Total(%)
P vs NP	93.6	94.3		92.8
3 class	93.6	94.3	92.8	93.4

Table 4.2: Accuracy of classification of LBP model for 40*40

- **Distance vs Accuracy plot:**

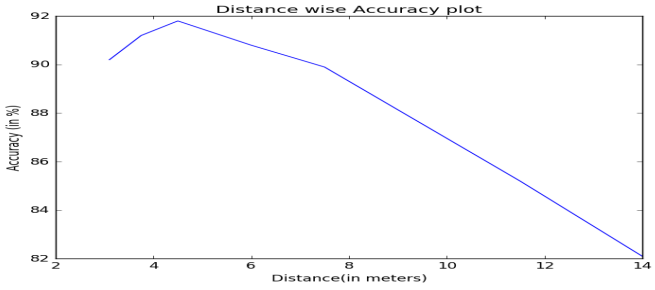

Figure 4.7: Distance vs accuracy for LBP

The accuracy increases until an optimal distance for pivot head i.e., around 3-4 meters and then the accuracy drops as the distance increases because texture is no longer visible with distinct features.

Chapter 5

GLCM based Algorithm

5.1 GLCM(gray-level co-occurrence matrices)

Whether considering the intensity or grayscale values of the image or various dimensions of color, the co-occurrence matrix can measure the texture of the image. Because co-occurrence matrices are typically large and sparse, various metrics of the matrix are often taken to get a more useful set of features. Features generated using this technique are usually called Haralick features. A co-occurrence matrix or co-occurrence distribution is a matrix that is defined over an image to be the distribution of co-occurring pixel values (grayscale values, or colors) at a given offset.
The offset, is a position operator that can be applied to any pixel in the image (ignoring edge effects).
An image with p different pixel values will produce a $p * p$ co-occurrence matrix, for the given offset. The (i, j) value of the co-occurrence matrix gives the number of times in the image that the i^{th} and j^{th} pixel values occur in the relation given by the offset [63].
The co-occurrence matrix captures numerical features of a texture using spatial relations of similar gray tones. Numerical features computed from the co-occurrence matrix can be used to represent, compare, and classify textures. The following are a subset of standard features derivable from a normalized co-occurrence matrix as shown in Equations:

- *Contrast*: Measures the local variations in the gray-level co-occurrence

matrix.

$$\sum_{n=0}^{N_g-1} n^2 \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} P[i,j]$$

- *Correlation*: Measures the joint probability occurrence of the specified pixel pairs.

$$\frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} (i,j) P[i,j] - \mu_x \mu_y}{\sigma_x \sigma_y}$$

- *Energy*: Provides the sum of squared elements in the GLCM. Also known as uniformity or the angular second moment.

$$\sum_i \sum_j P[i,j]^2$$

- *Entropy*: Measures the degree of randomness in the pixels

$$\sum_i \sum_j P[i,j](lnP[i,j])$$

- *Dissimilarity*: Measures the differences of the distribution of elements in the GLCM to the GLCM diagonal.

$$\sum_{n=0}^{N_g-1} n \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} P[i,j]$$

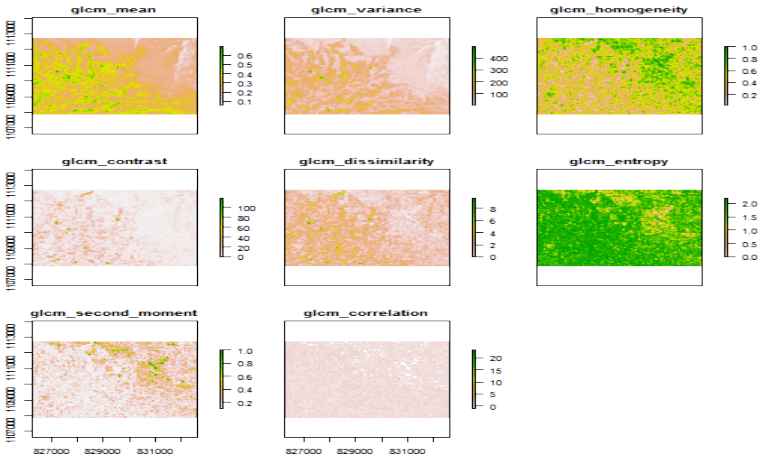

Figure 5.1: GLCM features

5.2 Algorithm

- **Dataset:** Texture Version 1.1, Texture Version 1.2
- **Training:**
  - *Preprocessing:*
    i) Noise is removed by applying Gaussian filter.
    ii) Contrast is increased so that edges are clear.

  Given a training set of block of images, extract GLCM features from them:

  1. Set the offset between the pixels to 1 to compute the GLCM matrix.
  2. Extract the values of Contrast, Correlation, Energy, Homogeneity, Dissimilarity to create a feature vector.
  3. These feature vectors are given to SVM Classifier for training.

- **Testing:**

  1. *Preprocessing:*
     i) Noise is removed by applying Gaussian filter.
     ii) Contrast is increased so that edges are clear.

2. Image of size 640*480 is divided into 3*3 grid and each block is classified into its corresponding class.
3. Image of size 640*480 is divided into 40*40 blocks and each block is classified into its corresponding class.
4. Few rows in the upper portion of the image are removed because it mainly contains sky and trees.

5.3 Observations

1. Gives acceptable accuracy to be working in real life scenario.
2. Handles variations in data.
3. Runs fast due to matrix operations.

5.4 Output

5.4.1 Output: 160*100

Figure 5.2: Output of GLCM for Pavement vs Not-Pavement for 160*100 block

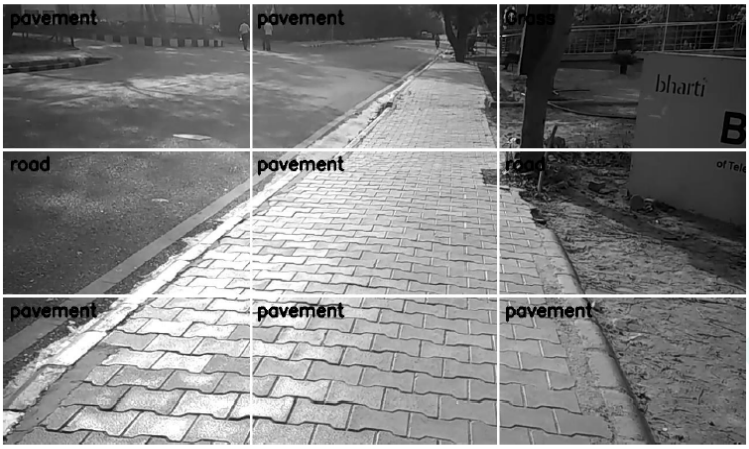

Figure 5.3: Output of GLCM for 3 class for 160*100 block

5.4.2 Output: 40*40

The red color blocks depicts the detected pavement blocks and blue color blocks depicts the detected non pavement blocks.

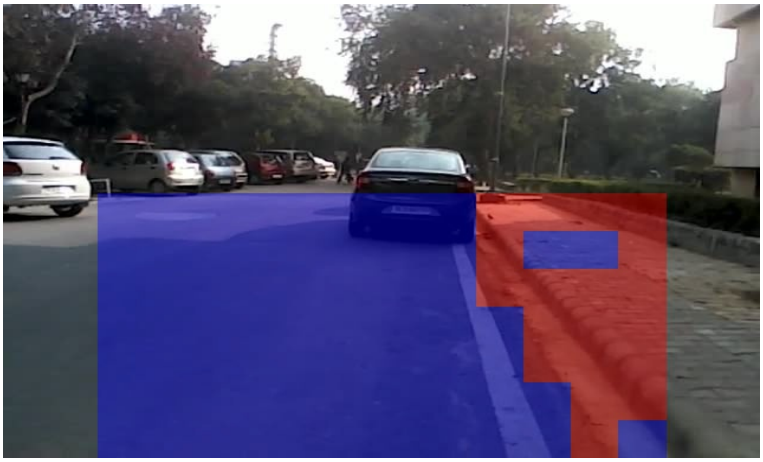

Figure 5.4: Output of GLCM for Pavement vs Not-Pavement for 40*40 block

The red color blocks depicts the detected pavement blocks and blue color blocks depicts the detected road blocks. Green color denotes the 'others' blocks.

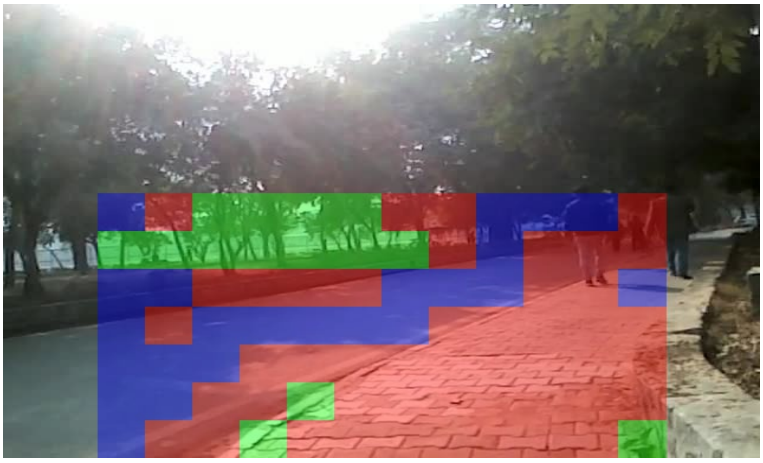

Figure 5.5: Output of GLCM for 3 class for 40*40 block

5.5 Hardware Porting

- Since, only C++ (OpenCV) could be cross-compiled on the ZedBoard, the entire code was written from scratch in C++ along with feature extraction code to port it on ZedBoard.
- The ZedBoard is approximately 10 times slower than the normal computer.
- Running time on Desktop: 9 frames/sec
- Running time on ZedBoard: 3.2 sec/frame
- **CDF of running time of GLCM on Zedboard:**

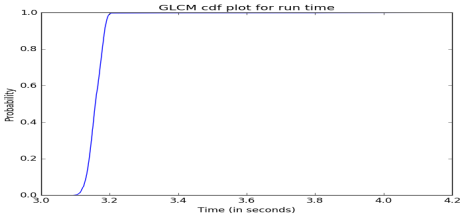

Figure 5.6: CDF for GLCM based model

The CDF plot depicts the time taken to compute GLCM features for 40*40 blocks on Zedboard and to predict the class of each block. This plot gives us an estimate of the fraction of frames taking the highest run time. This information can be used to analyze the frames that takes high run time, thus will help in resolving the issue. The long tail depicts that most of the frames takes similar amount of time to compute GLCM features. This is due to the fixed window size for prediction. Although, time taken on few frames to compute GLCM features is different which might arise due to ZedBoard being heatup.

5.6 Android Porting

Since, the ZedBoard Hardware is difficult to carry due to space, energy consumption, an android app was created which process real time video through phone's camera. It processes frames at a normal walking speed of 1 frame/sec and gives the color coded output to guide the user by speaking out whenever it identifies sidewalks/pavement.

Figure 5.7: GLCM Model on Android

5.7 Results

- **Accuracy:**

Method	Pavement(%)	Road(%)	Others(%)	Total(%)
P vs NP	96.8	94.9		95.7
3 class	92.8	94.3	93.2	93.8

Table 5.1: Accuracy of classification of GLCM Model on 160*100

Method	Pavement(%)	Road(%)	Others(%)	Total(%)
P vs NP	96.8	95.3		95.2
3 class	96.8	95.3	95.2	95.8

Table 5.2: Accuracy of classification of GLCM Model on 40*40

- **Distance vs Accuracy plot:**

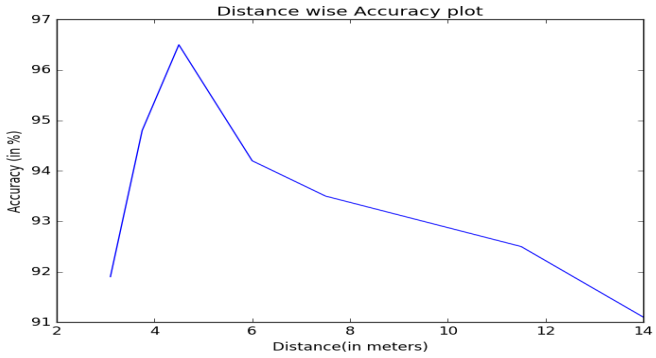

Figure 5.8: Distance vs accuracy for GLCM

The accuracy increases until an optimal distance for pivot head i.e., around 3-4 meters and then the accuracy drops as the distance increases because texture is no longer visible with distinct features.

Chapter 6

SegNet: Deep Neural Network

6.1 SegNet

SegNet is a Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling. One key ingredient of the SegNet is the use of max-pooling indices in the decoders to perform upsampling of low resolution feature maps. This has the important advantages of retaining high frequency details in the segmented images and also reducing the total number of trainable parameters in the decoders. This core trainable segmentation engine consists of an encoder network, a corresponding decoder network followed by a pixel-wise classification layer. The architecture of the encoder network is topologically identical to the 13 convolutional layers in the VGG16 network [54]. The role of the decoder network is to map the low resolution encoder feature maps to full input resolution feature maps for pixel-wise classification. It is designed to be efficient both in terms of memory and computational time during inference. It is also significantly smaller in the number of trainable parameters than other competing architectures like FCN and CRF and it can be trained end-to-end using stochastic gradient descent. The raw SegNet predictions tend to be smooth even without a CRF based post-processing [64].

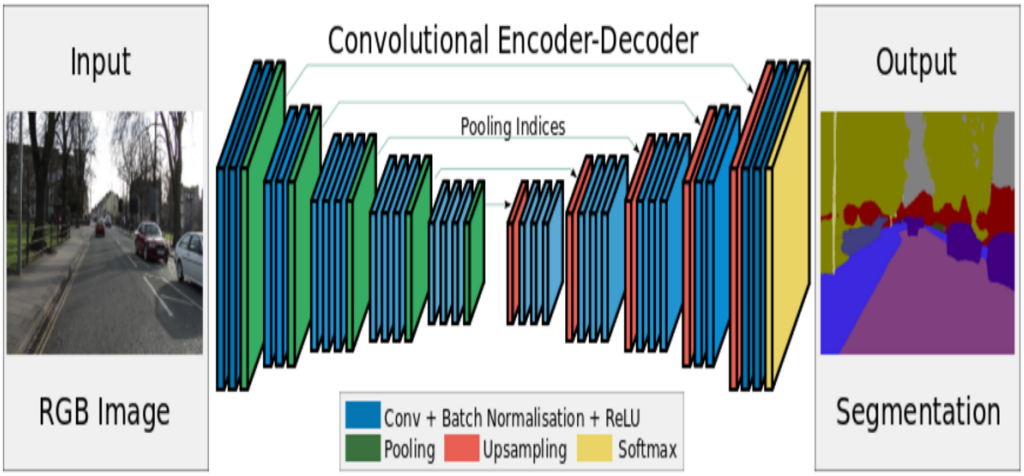

Figure 6.1: SegNet Architecture

6.2 Pre-trained Model

The pre-trained model available parse the road scene into twelve classes mainly road, pavement, buildings etc. The pre-trained model doesn't perform well in Indian scenario due to the difference in textures.

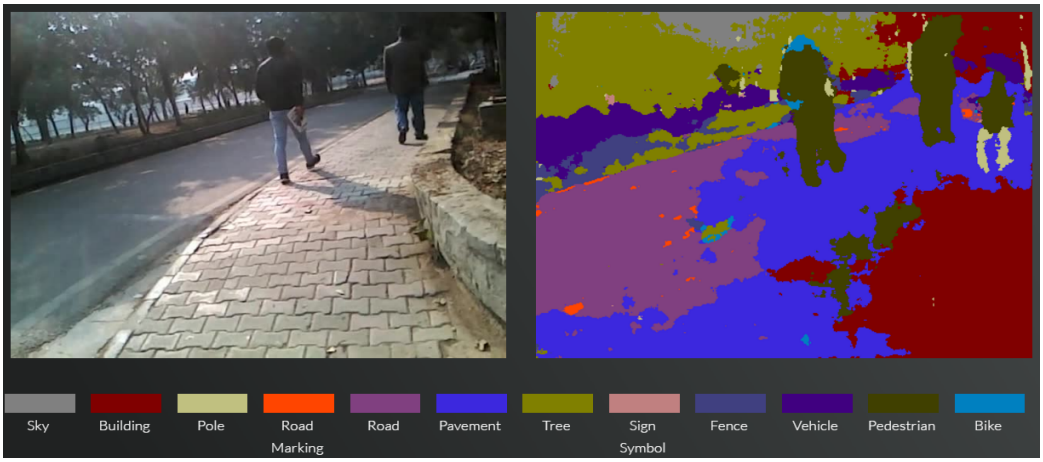

Figure 6.2: Output of pre-trained model in Indian Scenario

6.3 Fine Tuning of SegNet

The original SegNet model is trained on a total of 373 images of size 480*360. Median frequency balancing is used so that the classes which are dominant like Sky doesn't dominate while training. It reduces its weight.
In order for SegNet to use for our purpose a total of 150 images were annotated pixel wise to fine tune the model for our purpose.
Median frequency balancing was done on the annotated images of size 480*360 and the model was learned until 10,000 iterations.

6.3.1 Output: Pavement vs Not-Pavement

Figure 6.3: Output of Fine Tuned model for Pavement vs Not-Pavement

Legands:
Blue- Non-Pavement pixels **Pale Yellow**- Pavement pixels

6.3.2 Output: Road, Pavement, Others

The following is the output for 3 class i.e., Road, Pavement and Others. Grass is being merged with Others.

Figure 6.4: Output of Fine Tuned model for 3 class

Legands:
Grey color-Road **Pale Yellow-** Pavement
Red- Background

6.3.3 Observations

- The number of parameters in the basic SegNet model is 1.425 million.
- High Memory consumption at run-time (in Gb's).
- Due to the high number of parameters of the trained model, it cannot be ported on hardware.

6.3.4 Results

Method	Pavement(%)	Road(%)	Others(%)	Total(%)
P vs NP	90.3	96.8		93.3
3 class	90.2	95.5	94.3	93.5

Table 6.1: Accuracy of classification of SegNet Model on 160*100

Method	Pavement(%)	Road(%)	Others(%)	Total(%)
P vs NP	90.2	98.5		94.3
3 class	90.2	98.5	97.3	95.2

Table 6.2: Accuracy of classification of SegNet Model on 40*40

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The problem is currently modelled as both classification and segmentation task. Different techniques are explored and their accuracies, time consumption and run-time memory requirement are reported. Based on the experiments, GLCM outperforms in terms of time, and accuracy as GLCM features can be extracted just by matrix operations which is robust in different scenarios and makes it faster to run on hardware. In real time, high accuracy is required so as the visually challenged person have a clear and correct idea about the surface and can make decision accordingly. Deep neural network such as SegNet performs better than GLCM in terms of accuracy but still lacks behind for hardware porting because of extremely high requirement of memory at run time and lack of support for its code to port on hardware. Classification task was performed on two different window sizes so that a person using the device can change the mode accordingly to get a broader view (160*100) and granular view (40*40) of the scenario. The accuracy for different techniques for broader view is shown in Table 7.1 and for granular view is shown in Table 7.2.

Method	Pavement(%)	Road(%)	Others(%)	Total(%)
Gabor	87.66	68.5	87.8	81.6
LBP	91.6	92.3	93.8	92.4
GLCM	92.8	94.3	93.2	**93.8**
SegNet	90.2	95.5	94.3	93.5

Table 7.1: Accuracy Comparison for 160*100 block

Method	Pavement(%)	Road(%)	Others(%)	Total(%)
LBP	93.6	94.3	92.8	93.4
GLCM	96.8	95.3	95.2	**95.8**
SegNet	90.2	98.5	97.3	95.2

Table 7.2: Accuracy Comparison for 40*40 block

7.2 Future Work

This work can be extended further as follows:

- Hardware acceleration is required to be done so that these algorithms can run in real time, matching the walking speed of a person.
- Hardware porting of deep neural network has to be done so that it can be used in real time devices while retaining its accuracy.
- A light model like SqeezeNet which is approximately 300 times lighter than SegNet can also be created for this problem but its porting is again a big question to be answered.
- The database should be updated to capture more variations and diversity.
- The model training needs to be updated after few days so that it does not get obsolete. This has to be taken care of.

Bibliography

[1] M. Quintana, J. Torres and J. M. Menéndez, *A Simplified Computer Vision System for Road Surface Inspection and Maintenance*, in IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 3, pp. 608-619, March 2016.

[2] S. Paquis, V. Legeay, and H. Konik, *Road surface classification by thresholding using morphological pyramid*, in Proc. IEEE ICPR, Barcelona, Spain, 2000, vol. 1, pp. 1334–1337.

[3] M. Gavilán et al., *Adaptive road crack detection system by pavement classification*, Sensors, vol. 11, no. 10, pp. 9628–9657, Oct. 2008.

[4] Y. Hu and C. Zhao, *A local binary pattern based methods for pavement crack detection journal of pattern recognition research*, J. Pattern Recognit. Res., vol. 1, no. 2013, pp. 140–147, 2010.

[5] N. T. Sy, M. Avila, S. Begot and J. C. Bardet, Detection of defects in road surface by a vision system, MELECON 2008 - The 14th IEEE Mediterranean Electrotechnical Conference, Ajaccio, 2008, pp. 847-851.

[6] Vectra's AMAC specifications: `http://www.vectra.fr/AMAC.pdf`

[7] A. Kumar and G. K. H. Pang, Defect detection in textured materials using optimized filters, in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 32, no. 5, pp. 553-570, Oct 2002.

[8] H. Tamura, S. Mori, T. Yamawaki, *Textural features corresponding to visual perception*, IEEE Transactions on Systems, Man, and Cybernetics, 460–473.

[9] D. Chetverikov, *Pattern regularity as a visual key*, Image and Vision Computing, Volume 18, Issue 12, September 2000, Pages 975-985.

[10] R.M. Haralick, L.G. Shapiro, Computer and Robot Vision, vols. I–II, Addison-Wesley, Reading, MA, 1992–1993.

[11] Dash M., Liu H., *Feature selection for classification*, Intelligent data analysis, Elsevier, pp: 131–156, 1997.

[12] Koller D., Sahami M., *Toward optimal feature selection*, Proc. of the13th Int.conf on machine learning, pp. 284-292, 1996.

[13] Zhang P., Peng J., Buckles B., *Learning optimal filter representation for texture classification*, Int.conf. on Pattern Recognition, vol.2, pp. 1138-1141, 2006.

[14] Chen C. H., Pau L. F., and Wang P. S. P., *Texture analysis: in Handbook of Pattern Recognition and Computer Vision*, Eds, Singapore: World Scientific, pp. 235-276, 1993.

[15] Reed T. R., and Du Buf J. M. H., *A review of recent texture segmentation and feature extraction techniques*, Comput. Vis., Graph, Image process.: Image understand., vol. 57, no. 3, pp. 359-372, 1993.

[16] Haralick R.M., *Statistical and structural approaches to texture*, Proceedings of the IEEE, vol.67, pp: 786-804.

[17] Ojala, T., Pietikinen M., Harwood D., *A comparative study of texture measures with classification based on feature distributions*, Pattern Recognition, vol.29(1), pp: 51-59, 2006.

[18] Chandra Mohan M., Vijaya Kumar V., Sujatha B., *Classification of child and adult based on geometric features of face using linear wavelets*, IJSIP, vol.1, Iss.3, pp:211-220, 2010

[19] Chellappa R., Chatterjee S., *Classification of textures using Gaussian Markov Random Fields*, IEEE Trans. Acoustics Speech Signal Process, vol. 33(4), pp:959-963.

[20] Chen P.C., Pavlidis T., *Segmentation by texture using correlation*, IEEE Trans. PAMI, vol.5, pp: 64-69.

[21] Davis L.S., Johns S.A., Aggarwal J.K., *Texture analysis using generalized co-occurrence matrices*, IEEE Trans. PAMI, vol.1, pp:251-259.

[22] Derin H., Elliot H., *Modeling and segmentation of noisy and textured images using Gibbs Random Fields*, IEEE Trans. PAMI, vol.9, pp: 39–59.

[23] Haralick R.M., Shanmugan K. and Dinstein I., *Textural features for image classification*, IEEE Trans. Sysr. Man. Cybern., vol. 3, pp:610-621.

[24] Julesz B., *A theory of pre-attentive texture discrimination based on first order statistics of textons*, Biol. Cybernet., vol.41, pp:131-138.

[25] Raju U.SN., Eswara Reddy B., Vijaya Kumar V. , Sujatha B., *Texture classification based on extraction of skeleton primitives using wavelets*, Information Technology journal, vol.7, Iss.6, pp: 883-889, 2008.

[26] Unser M., *Local linear transforms for texture measurements*, Signal Process., vol.11, pp: 61-79.

[27] Unser M., *Texture classification and segmentation using wavelet frames*, IEEE Trans. Image Process., vol. 4, pp: 1549-1560, 1995.

[28] Ramana Reddy B.V., Sujatha B., Vijaya Kumar V., *Texture classification based on random threshold vector technique*, IJMUE, vol. 5, Iss.1, January, 2010.

[29] Alata O., Cariou C., Ramananjarasoa C., and Najim M., *Classification of rotated and scaled textures using HMHV spectrum estimation and the Fourier-Mellin Transform*, Proc. IEEE Int'l Conf. Image Processing, vol. 1, pp: 53-56, 1998.

[30] Cohen F.S., Fan Z., and Patel M.A., *Classification of rotated and scaled texture images using Gaussian Markov Random Field Models*, IEEE Trans. PAMI, vol. 13, pp: 192-202, 1991.

[31] Leung M.M., and Peterson A.M., *Scale and rotation invariant texture classification*, Proc. 26th Asilomar Conf. Signals, Systems and Computers, vol.1, pp: 461-465, 1992.

[32] Manian V. and Vasquez R., *Scaled and rotated texture classification using a class of basis functions*, Pattern Recognition, vol. 31, pp:1937-1948, 1998.

[33] Wu Y. and Yoshida Y., *An efficient method for rotation and scaling invariant texture classification*, Proc. IEEE Int'l Conf. Acoustics Speech and Signal Processing, vol. 4, pp: 2519-2522, 1995.

[34] You J. and Cohen H. A., *Classification and segmentation of rotated and scaled textured images using texture tuned masks*, Pattern Recognit., vol. 26, pp. 245-258, 1993.

[35] Ojala T., Pietikainen M., *Multiresolution gray-scale and rotation invariant texture classification with local binary patterns*, IEEE Trans., vol. 24, no. 7, pp: 971–987, 2002.

[36] Mäenpää T., Pietikäinen M., *Multi-scale binary patterns for texture analysis*, Proc. of 13th Scandinavian Conf. on Image Analysis, 2003.

[37] Khouzani K.J., Hamid S.Z., *Radon transform orientation estimation for rotation invariant texture analysis*, IEEE Trans. PAMI, vol. 27(6), pp: 1004-1008, 2005.

[38] Mäenpää T., *The local binary pattern approach to texture analysisextensions and applications*, Academic Dissertation, Infotech Oulu and Department of Electrical and Information Engineering, 2003.

[39] Mäenpää T., Ojala T., Pietikäinen M., Soriano M., *Robust texture classification by subsets of local binary patterns*, in: 15th Int. conference of Pattern Recognition, vol. 3, pp:947-950, 2000.

[40] Mäenpää T., Pietikäinen M., Ojala T., *Texture classification by multi-predicate local binary pattern operators*, Proceedings of 15th Int. conference on Pattern Recognition, Barcelona, pp. 951-954, 2000.

[41] Ojala T., Mäenpää T., Pietikäinen M., Viertola J., Kyll Outex J., *A new framework for empirical evaluation of texture analysis algorithms*, Proceedings of 16th Inter. Conf. on Pattern Recognition, pp:701-706, 2002.

[42] Ojala T., Pietikäinen M. and Mäenpää T.. *A generalized local binary pattern operator for multi-resolution gray scale and rotation invariant texture classification*, Machine Vision and Media Processing Unit, Infotech Oulu, University of Oulu, Finland.

[43] Ojala T., Pietikäinen M., *Unsupervised texture classification using feature distributions*, Pattern Recogn., vol.32, pp:477-486, 1999.

[44] Ojala T., Pietikäinen M., Mäenpää T., *Gray scale and rotation invariant texture classification with local binary patterns*, Proceedings of Sixth European conference on Computer Vision, pp:404-420, 2000.

[45] Ojala T., Valkealahti K., Oja E., Pietikäinen M., *Texture discrimination with multidimensional distributions of signed gray level differences*, Pattern Recogn., vol.34, pp:727-739, 2001.

[46] Petrou M., Garcia-Sevilla P., *Image processing: Dealing with texture*, Wiley, 2006.

[47] Pietikäinen M., *Image analysis with local binary patterns*, Image Analysis, SCIA Proceedings, Lecture Notes in Computer Science, vol. 3540, pp: 115-118, 2005.

[48] Pietikäinen M., Ojala T., Xu Z., *Rotation-invariant texture classification using feature distributions*, Pattern Recogn., vol. 33 , pp:43-52, 2000.

[49] J. Long, E. Shelhamer, and T. Darrell, *Fully convolutional networks for semantic segmentation*, in CVPR, pp. 3431–3440, 2015.

[50] H. Noh, S. Hong, and B. Han, *Learning deconvolution network for semantic segmentation*, in ICCV, pp. 1520–1528, 2015.

[51] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, *Conditional random fields as recurrent neural networks*, in Proceedings of the IEEE International Conference on Computer Vision, pp. 1529–1537, 2015.

[52] D. Eigen and R. Fergus, *Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture*, in ICCV, pp. 2650–2658, 2015.

[53] S. Hong, H. Noh, and B. Han, *Decoupled deep neural network for semisupervised semantic segmentation*, in NIPS, pp. 1495–1503, 2015.

[54] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, arXiv preprint arXiv:1409.1556, 2014.

[55] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, *ImageNet Large Scale Visual Recognition Challenge*, International Journal of Computer Vision (IJCV), pp. 1–42, April 2015.

[56] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, *The pascal visual object classes challenge: A retrospective*, International Journal of Computer Vision, vol. 111, no. 1, pp. 98–136.

[57] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, *Microsoft coco: Common objects in context*, in Computer Vision–ECCV 2014, pp. 740–755, Springer, 2014.

[58] C. Liang-Chieh, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, *Semantic image segmentation with deep convolutional nets and fully connected crfs*, in ICLR, 2015.

[59] L. Bottou, *Large-scale machine learning with stochastic gradient descent*,in Proceedings of COMPSTAT'2010, pp. 177–186, Springer, 2010.

[60] MAVI: `http://www.cse.iitd.ac.in/mavi/`

[61] Gabor Filters: `https://en.wikipedia.org/wiki/Gabor_filter`

[62] LBP Features: `https://en.wikipedia.org/wiki/Local_binary_patterns`

[63] GLCM Features: `https://en.wikipedia.org/wiki/Co-occurrence_matrix`

[64] V Badrinarayanan, Alex Kendall, Roberto Cipolla *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*, arXiv:1511.00561.

[65] R. Kedia, K. K. Yoosuf, P. Dedeepya, M. Fazal, C. Arora and M. Balakrishnan, *MAVI: An Embedded Device to Assist Mobility of Visually Impaired*, 30th International Conference on VLSI Design and 2017 16th International Conference on Embedded Systems (VLSID), Hyderabad, 2017, pp. 213-218.