Självständigt arbete på avancerad nivå
Independent degree project – second cycle

Huvudområde
MSc. in Electronics Design

Titel
Automatic recognition of tree trunks in images, Robotics in forest industry

Förnamn Efternamn
Tamrat Gebremedhin Gebre
Abstract

A novel machine vision system for the recognition of tree trunks in forest images and videos is proposed on this paper, which is based on linear multiclass Support Vector Machines (SVM) classification approach. The system solves problems that arise due to the complex backgrounds, non-homogeneous illumination, shadow, the variations in the bark’s colour and texture and the change in the appearance of the same tree with a change in season. First, forest images were captured by a digital camera and then the acquired images were grouped into training images and test images. Second, the training images were pre-processed to acceptable sizes, texture and colour feature extractions were done by using appropriately designed Gabor filters, mean and variance. Third, one versus all multiclass linear SVMs were constructed and trained with the training feature vectors. The performance of colour and texture descriptors alone and in fusion at different colour spaces have been evaluated using normalized Sum Absolute Difference (SAD). The empirical study demonstrated that the combination of mean, variance and Gabor filter features achieve the best classification performance of 95.33% in CbCr colour space. The study has also shown promising result for the identification of trees in the forest using multiclass linear SVMs that could be applied in real time environment.

Keywords: Machine vision system, linear multiclass SVM, tree trunk recognition, classification, feature extraction, Gabor filters, feature vectors, colour feature, texture feature, descriptors, CbCr colour space, normalized SAD.
Acknowledgements

I would like to pass my sincere gratitude to my supervisor Dr. Benny Thörnberg for supporting me throughout the project. His knowledge and way of assessing the quality of paper have been very useful for me to improve the paper in better way. On a personal level, I would like to thank the saviour Jesus Christ the son of Virgin Mary and my beloved parents for their encouragement, believe and support throughout my stay and studies in Sweden.

Tamrat G.
2014
# Table of Contents

Abstract ............................................................................................................ iii

Acknowledgements ....................................................................................... iv

Terminology .................................................................................................... vi

1  Introduction............................................................................................ 1
   1.1  Background and problem motivation ...................................... 1
   1.2  Overall aim ................................................................................... 2
   1.3  Scope ............................................................................................. 2
   1.4  Concrete and verifiable goals .................................................... 2
   1.5  Outline .......................................................................................... 3

2  Related Work.......................................................................................... 5

3  Fundamentals of Machine Vision ...................................................... 7
   3.1  2D Convolution ........................................................................... 7
   3.2  Statistical Colour Features ......................................................... 8
   3.3  Texture features and Gabor filter banks .................................. 9
   3.4  Multiclass Linear SVMs ............................................................ 12

4  Methodology ........................................................................................ 15
   4.1  Preprocessing ............................................................................. 17
   4.2  Feature extractions .................................................................... 18
   4.3  Multiclass Linear SVM ............................................................. 20
   4.4  Post processing .......................................................................... 21
   4.5  Performance Evaluation ........................................................... 24

5  Proposed Hardware Architecture ..................................................... 25
   5.1  Modified 2D Filters Convolution ............................................ 27

6  Results ................................................................................................... 33

7  Discussion and Conclusions ............................................................. 41
   7.1  Discussion.................................................................................... 41
   7.2  Conclusions ................................................................................ 42
   7.3  Future works .............................................................................. 43

References ........................................................................................................ 45
Appendix A: Classified images........................................................................49
Terminology

Acronyms / Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>SVMs</td>
<td>Support Vector Machines</td>
</tr>
<tr>
<td>RGB</td>
<td>Red, Green, Blue</td>
</tr>
<tr>
<td>CbCr</td>
<td>The blue-difference and red-difference chroma components</td>
</tr>
<tr>
<td>HSV</td>
<td>Hue, Saturation, Value</td>
</tr>
<tr>
<td>KNN</td>
<td>k-Nearest Neighbor</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>YCbCr</td>
<td>Luminance; Chroma: Blue; Chroma: Red.</td>
</tr>
<tr>
<td>SAD</td>
<td>Sum of Absolute Differences</td>
</tr>
<tr>
<td>FD</td>
<td>Feature Descriptor</td>
</tr>
<tr>
<td>2-D</td>
<td>Two Dimensional</td>
</tr>
<tr>
<td>3-D</td>
<td>Three Dimensional</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Arrays</td>
</tr>
<tr>
<td>VHDL</td>
<td>Verilog Hardware Description Language</td>
</tr>
<tr>
<td>Kbits</td>
<td>Kilo Bits</td>
</tr>
<tr>
<td>FIFO</td>
<td>First In First Out shift registers</td>
</tr>
</tbody>
</table>
### Mathematical notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_{\lambda,\theta,\varphi}(x,y)$</td>
<td>Gabor filtered image</td>
</tr>
<tr>
<td>$G_{\lambda,\theta,\varphi}(x,y)$</td>
<td>Gabor filter banks</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Standard deviation of Gaussian</td>
</tr>
<tr>
<td>$\Theta$</td>
<td>Orientation angle</td>
</tr>
<tr>
<td>$\varphi$</td>
<td>Phase offset</td>
</tr>
<tr>
<td>$b$</td>
<td>Band width</td>
</tr>
<tr>
<td>$\alpha_i$</td>
<td>Lagrange multiplier</td>
</tr>
<tr>
<td>$C$</td>
<td>Correction factor</td>
</tr>
<tr>
<td>$\xi_i$</td>
<td>Slack variables</td>
</tr>
<tr>
<td>$i^*$</td>
<td>combined classification output</td>
</tr>
</tbody>
</table>
1 Introduction

The advancement of forestry production brought keen interest to incorporate autonomous obstacle detection system to its huge expensive machines. The Autonomous Navigation for Forest Machines research undergone at the department of computing science in Umeå University gives higher emphasis on developing autonomous forwarder. This machine holds piles of wood logs and transports them to an area close to a road for additional transportation [2]. The motivation of their project was to cut back costs of excessive man power while maximizing production.

GPS-coordinates can be an input for the forwarder’s navigation and drive itself from forest to wayside [2]. Still, such a machine would want ways in which to robustly recognise and estimate distance to trees near its driving path. The developed system will record its track while it is driven by human and it uses the recorded track to identify obstacles.

This paper explored the existing methodology in [1] for reliable detection of tree trunks in forest images. It presents a machine vision system that recognize tree obstacles based on their appearance on both sides of an autonomous forest vehicle. This system also identifies which types of trees are there and marks each tree with its own predefined category. Moreover it introduces the multiclass Supported Vector Machines (SVMs) technique for the classification of tree trunks from the background and applies reliable post processing mechanism.

1.1 Background and problem motivation

In recent days the recognition of trees in forest environment becomes an interesting research field for some scholars. The machine vision system applied in the forests for the detection of obstacles need a great deal of studies due to the complex backgrounds, non-homogeneous illumination, shadow, the variations in the bark’s colour and texture even in a single tree. Moreover, the appearance of the same tree changes with a change in season. The nature of the backgrounds that varies even in a very small place with little but complex differences made it difficult to
be applicable in the real environment. Higher achievements have been observed from Wajid [1]. Solutions were addressed by the classification of trees using different types of classifier. However, the problem arises on the robustness how the trees are segmented and the size of training and classification process also brought some misclassification in the output and problem on feasibility to real time application.

1.2 Overall aim
The project’s overall aim is to design a machine vision model for the autonomous forwarder machine that detects and identifies tree obstacles along its path. Furthermore, the project has also aim to investigate the reliability of the design to be applicable in real time.

1.3 Scope
The study has mainly focused on designing feature descriptors that extract texture and colour features of training images. Train these feature vectors to the multiclass classifiers offline. Then in real time environment acquired image’s feature vectors are extracted with the same descriptors and supplied to the trained classifiers for the classification of image in to foreground (trees trunk) part and background part. Finally, label the different categories of trees with colours based on predefined class.

1.4 Concrete and verifiable goals
The objective of the thesis is to explore the existing method in [1] and model a machine vision system for a forwarder machine. The details of the goals are to:

1. Repeat major parts of the experiments done in [1] by MatLab modelling of such vision system.
2. Use SVM for the linear classifier and expand to networks of binary classifiers for multiple classes.
3. Automatically label the different tree trunks with different colours.
4. Test the classification performance of the model with a forest images and check if it can classify forest videos too.
5. Estimate the hardware implementation cost on FPGA.

1.5 Outline

The organization of the paper is as follows. In section 2, related works and in section 3 fundamentals of computer vision system used in the design are given in detail. Section 4, presents the methodology and section 5 presents the hardware architecture based on the main hardware resources then experimental results are given in section 6. Finally, discussion and conclusion are explained in section 7.1 and section 7.2 respectively.
**Automatic recognition of tree trunks in images – Robotics in forest industry**

Tamrat Gebremedhin

<table>
<thead>
<tr>
<th>1 Introduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-01-23</td>
</tr>
</tbody>
</table>


2 Related Work

Appearance based obstacle detection has got little emphasises compared to the extensive research done on the range based obstacle detection [11]. The pioneer on appearance based obstacle detection done on the first autonomous mobile robot Shakey by Nilsson [8] which operates on a smooth floor tiles. Shakey detects obstacles easily by using an edge detector from monochrome input image, but it only operates on smooth floor tiles whereas the outdoor environments have texture. An appearance based obstacle detection system developed by Ulrich and Nourobkhsh [11] classifies each pixel of the image based on its colour appearance either to the obstacles or to the background. Another system developed by K. Morioka and Hideki Hashimoto [4] also depends on the colour appearance for the identification of objects. Another system developed by O. Chapelle, P. Haffner and V. Vapnik [12] incorporates colour histogram as the feature vectors of the image and use Support Vector Machine (SVM) for the classification. Even though, the purpose of appearance based detection in [6] is different from obstacle detection. Their approach was using the most commonly used appearances that are shapes, textures and colours as features for leaf classification. Some obstacle detection systems incorporate both range and appearance based approach. For instance the algorithm developed by D. Maier and M. Bennewitz [5] is a hybrid technique of range and appearance based obstacles detection. It acquires training data from structural information, but reaches to conclusion based on colour and texture appearance. Another algorithm designed by fusion of laser scanner and camera used as obstacle detection for active pedestrian detection system [7]. This system uses the laser scanner to cluster and track range data and generate obstacle candidates. Then, the texture based vision system classifies the obstacles by SVM into pedestrian, vehicle and other.

The speed, simplicity, ability to detect smaller obstacles and foremost the lower cost made this project to stick on the idea of appearance based autonomous obstacle detection. Most of the appearance based obstacle detection systems discussed above are based on either colour or texture cues. Where colour based only doesn’t perform well due to variation in illumination and texture only has also problem of performing well for lower textured trees (young trees). Recent research made by W. Ali, F.
Georgsson and T. Hellström [1] on the tree detections for autonomous navigation in forest environment used fusion of colour and texture cues to classify trees from background. In their work different types of colour and texture descriptors are analyzed separately and in fusion form. Their approach was first to divide the test image to smaller blocks and extract features from each block, finally classify those blocks using the trained k-Nearest Neighbour (KNN) or Artificial Neural Network (ANN) classifiers [1]. The systems developed by [1] have faced misclassification of blocks to a tree or background because their work depends in a block based approach. So the motivation behind my project is to overcome this problem and analyze other mechanisms of classification approach.
3 Fundamentals of Machine Vision

3.1 2D Convolution

Convolution is a mathematical operation applied on two functions \( f \) and \( h \) that gives a new function \( g \). Where \( g \) is a filtered version of \( f \) and \( h \) is the filter [25]. The discrete 1D convolution in spatial domain is expressed in Eq. 3.1 below:

\[
g[x] = f[x] * h[x] = \sum_{k=-\infty}^{\infty} f[k] \cdot h[x - k]
\]

A 2D convolution is an extension of the 1D convolution that convolve both in horizontal and vertical directions on 2D spatial domain. Its formula for the discrete function is expressed as:

\[
g[x, y] = f[x, y] * h[x, y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1, n_2] \cdot h[x - n_1, y - n_2]
\]

In image processing, convolution is made between the input image and a window based filter for smoothing, edge detection, blurring or for the extraction of feature that are basics to the classification of images. For an image with a size of \((M \times N)\) pixels and the filter with a size of \((K \times K)\) pixels, then filtering one pixel of the image needs \(K^2\) multiplications and accumulations. For the whole image it needs \((M \cdot N \cdot K^2)\) multiplications and accumulations. However, these massive computation can be reduced to \((2 \cdot M \cdot N \cdot K)\) if the filter is separable [26].

Separable 2D filters

A 2D matrix is separable if it can be decomposed to two 1D vectors and the product of these two vectors can give back the original 2D matrix. From the following example we can see that matrix A is separated to B and C vectors.
A good example to these types of filters is \([K \times K]\) window based 2D mean filter, which is separable to two 1D filters.

\[
A = \frac{1}{K^2} \begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{bmatrix} = B \cdot C \quad B = [1 1 1 1 1], \quad C = \begin{bmatrix}
1 \\
1 \\
1 \\
1 \\
1 \\
\end{bmatrix}
\]

The 2D convolution is separated to two 1D convolutions convolve first horizontally then the result is convolved vertically to get the final result. The convolution between these types of filters with a kernel size \([K \times K]\) reduces the convolution computations from \(K^2\) to 2·\(K\) multiplication and addition operations per pixel of an image. For larger size kernels this method of convolution reduces more than 80% of the resource used by the computations.

### Symmetrical 2D filters

A 2D matrix is symmetrical if its transpose is the same as the original matrix. In case of 2D convolution it gives ease of adding the symmetrical values before the multiplication that reduces the resource demanding multiplication operations at least by half. Gabor filters are either symmetrical or anti-symmetrical filters so it can take advantage of reducing the 2D convolution computation based on symmetrical approach [27].

### 3.2 Statistical Colour Features

The statistical colour features are quantified from the original pixel values without any dependence on the pixel’s region relation or use of matrix transforms. The statistical descriptors for example mean, variance, skewness, kurtosis, dispersion and entropy are categorized into this class [9][27].
Different types of these features were combined and tried in the work of [14] then it shows that the mean and variance filters had a significant impact on the performance. Moreover, they are easy to be implemented in the hardware as well [14]. The local mean and variance for a window size of N x N are described as shown below,

\[
\text{mean}(i,j) = \frac{1}{(N \cdot N)} \sum_{r=i-w}^{i+w} \sum_{c=j-w}^{j+w} I(r,c) 
\]

\[
\text{variance}(i,j) = \frac{1}{(N \cdot N)} \sum_{r=i-w}^{i+w} \sum_{c=j-w}^{j+w} (I(r,c) - \text{mean}(i,j))^2
\]

Where, \( I(r,c) \) is the pixel value of image I of size m x n and \( w = \frac{N-1}{2} \). For RGB colour space, the tree trunk features are extracted from each plane R, G, and B. and, in the YCbCr colour space, the two features are extracted considering only the blue (Cb) and red (Cr) channels while removing the luminance (Y).

### 3.3 Texture features and Gabor filter banks

Texture has the information that defines the structural arrangement of the surface with respect to the surrounding environment. However, this information is hard to be recognized and described in empirical terms with computer systems. Therefore, extracting these texture features are important to discriminate among different parts of the image data. Some of the extraction methods are Gabor filters, Local Binary Patterns (LBP) and Edge Histogram Detection (EHD)[30].

Gabor filters are mainly used as texture descriptors [13][18]. They are tuneable band pass filters with the best localization both in the spatial and frequency domains. Thus, when applied for tree feature extraction, they extract spatially local features of a specified frequency band with resolution invariant capability [7].

I selected the following Gabor function type (refer [18][20][13][17]).

\[
G_{\lambda,\theta,\phi}(x',y') = e^{-((x'^2+y'^2)/2\sigma^2)} \cos(2\pi \frac{x'}{\lambda} + \phi)
\]
Where

\[ x' = x \cos \theta + y \sin \theta, \quad y' = -x \sin \theta + y \cos \theta \quad (3.6) \]

The set of image points are denoted by \((x, y)\). The \(\sigma\) in the Gaussian factor is a standard deviation that determines the size of the Gabor function receptive field [18]. Its eccentricity of the receptive field ellipse is expressed by the parameter \(\gamma\), called the spatial aspect ratio. Its value varies between 0.23 (very elliptical) and 1 (circular) as it is described in [18]. I kept \(\gamma = 0.5\) throughout the experiment. The parameter \(\lambda\) in the cosine part is the wavelength that specifies the spatial frequency of the Gabor function’s receptive field [17]. The value of \(\lambda\) is expressed in pixels and \(\lambda \geq 2\) and \(\lambda < \frac{1}{5}\) *(the image size). The orientation \(\Theta\) is the angular measurement in counterclockwise between the horizontal plane and the normal line to the stripes of the Gabor function that is between \(0^\circ\) and \(360^\circ\). In the cosine factor, \(\varphi\) is the phase offset that corresponds to the symmetry of the strips measured in degrees and the ranges are \(-180^\circ \leq \varphi \leq 180^\circ\). The values \(0^\circ\) and \(180^\circ\) represents the center-symmetric ‘center-on’ and ‘center-off’ functions, respectively, while \(-90^\circ\) and \(90^\circ\) represent the anti-symmetric functions. The value of \(\varphi = 0^\circ\) center-symmetric and \(\varphi = 90^\circ\) anti-symmetric are selected for our Gabor function. They represent the real and imaginary part of the Gabor function respectively [18]. The half response spatial frequency bandwidth \(b\) is proportional to \(\sigma/\lambda\) and is expressed as follows:

\[ b = \log_2 \frac{\sigma}{\pi} \frac{\sqrt{\ln 2}}{\sqrt{\ln 2}}, \quad \frac{\sigma}{\lambda} = \frac{1}{\pi} \sqrt{\frac{\ln 2}{2}} \cdot \frac{2^b + 1}{2^b - 1}. \quad (3.7) \]

A two dimensional Gabor function with particular orientation, wavelength, aspect ratio, bandwidth and symmetry is shown in Figure 3.1(a) and the corresponding spatial frequency response is shown in Figure 3.1(b).
Let $I(x,y)$ denote a gray scale forest image whose $(x,y) \in \mathbb{P}$ where $\mathbb{P}$ is the set of image points. Let the 2-Dimensional Gabor filter banks are expressed by $G_{\lambda,\theta,\varphi}(x,y)$. Then, the feature extracted image $r_{\lambda,\theta,\varphi}(x,y)$ is a convolution of the gray scale image and the Gabor filter banks.

$$r_{\lambda,\theta,\varphi}(x,y) = I(x,y) \ast G_{\lambda,\theta,\varphi}(x,y)$$

(3.8)

In the above expression $r(x, y)$ is a complex function which has real part for phase offset $\varphi=0^0$ and imaginary part $\varphi=90^0$. However, filter outputs by default are not appropriate for identifying key texture features therefore, feature extractions are made by the magnitude response Eq(3.9) expressed as follows:

$$r_{\lambda,\theta}(x,y) = \sqrt{r_{\lambda,\theta,(\varphi=0^0)}(x,y)^2 + r_{\lambda,\theta,(\varphi=90^0)}(x,y)^2}$$

(3.9)

To enhance the performance of the segmentation processes spatial smoothing were applied. Since, it suppresses large variations in the feature map in areas which belong to the same texture. However, excess smoothing can have a negative effect on the localization of texture region edges. The output of each filter is smoothed using a Gaussian smoothing function that goes with the corresponding filter spatial smoothing function that goes with the corresponding filter spatial smoothing function that goes with the corresponding filter spatial smoothing function that goes with the corresponding filter spatial smoothing function that goes with the corresponding filter spatial smoothing function that goes with the corresponding filter spatial smoothing function that goes with the corresponding filter spatial smoothing function that goes with the corresponding filter spatial smoothing function that goes with the corresponding filter spatial
Gaussian curve. The selected Gaussian smoothing must have larger window than the matched Gabor modulated Gaussian [19].

Another method to extract texture features is the LBP. As the name indicates LBP defines texture in terms of patterns and its strength [31]. First it divides the gray scale image into smaller square blocks of images. Then each pixel in a block is thresholded with respect to the center pixel and changes the result to binary number [31]. Further descriptions are found in [31].

The other most commonly used texture extraction method is Edge Histogram Descriptor (EHD). The EHD begins its work by dividing the gray scale image into 16 sub-images. Each sub-image is divided into a fixed number of blocks. Then for each image block a local edge histogram is quantified.

### 3.4 Multiclass Linear SVMs

In machine learning theory there are different advanced classification techniques among them support vector machine (SVM) is one [12]. Compared with other methods such as ANN, KNN, and Bayesian networks, SVM has significant advantages because of its good mathematical tractability, greater accuracy and simple geometric explanation. Moreover, SVM requires only a small number of training samples that avoid over fitting [3].

In this section very concise reviews of SVM is presented and refer the details to [16] [21] [23]. Consider N training samples: \{x_1, y_1\},...,\{x_N, y_N\}, where \(x_i \in \mathbb{R}^m\) is a m-dimensional feature vector indicating the \(i^{th}\) training sample, and \(y_i \in \{-1,1\}\) is the class label of \(x_i\). In the feature space a hyperplane can be explained by Eq.(3.10)

\[
F(x) = w^T x + b = 0,
\]

\[
\vec{w} = \sum_{i=1}^{N} y_i \alpha_i \vec{s}_i
\]
Minimize: $L(w, \xi) = \frac{1}{2} ||w||^2 + C \sum_{i=1}^{N} \xi_i$  
Subject to: $y_i(w^T x_i + b) \geq 1 - \xi_i, \; i = 1, \ldots, N.$

Where $\mathbf{w}$ is the weight vector, which is normal to the hyperplane, $S_i \in \mathbb{R}^m$ are support vectors and the Lagrange multiplier $(\alpha_i) \neq 0, \; N_s$ is the total number of support vectors and $b$ is a bias, which is a scalar offset value. When the training samples are linearly separable, SVM produces the optimal hyperplane that divides two classes with no training error, and maximizes the margin from the training samples to the hyperplane. It is simple to search out that the parameter pair $(w, b)$ equivalent to the maximized margin classifier, which is the solution to the following optimization problem:

Minimize: $L(w) = \frac{1}{2} ||w||^2$  \hspace{1cm} (3.11)  
Subject to: $y_i(w^T x_i + b) \geq 1, \; i = 1, \ldots, N.$

The above assumption fulfils if all the samples are linearly separable. However, for samples that are not linearly separable, there is not any hyperplane, which is capable of classifying these training samples with no error. Therefore, the above optimization problem is extended by using the idea of soft margin and the new optimization becomes:

Minimize: $L(w, \xi) = \frac{1}{2} ||w||^2 + C \sum_{i=1}^{N} \xi_i$  \hspace{1cm} (3.12)  
Subject to: $y_i(w^T x_i + b) \geq 1 - \xi_i, \; i = 1, \ldots, N.$

Figure 3.2: Binary linear SVM classifier
As it can be seen in Figure 3.2 for a sample data that satisfies the $W^T x + b > 1$ it will be grouped in class 2 or if the sample data satisfies the $W^T x + b < -1$ it will be grouped in class 1. The hyperplane is the middle line that separates the two classes. However, some data samples could be misclassified as it is seen in Figure 3.2. The distance between the misclassified sample with its respective separating plane are called slack variables $\xi_i$, which are associated to the soft margin and $C$ is the correction factor that balances the margin and the training error. For the tree classifier application a combination of binary linear SVMs are recommended. The approach is to use the one versus all classifiers [16][21][23]. Let an $M$-class problem, where we have $N$ training samples: $\{x_1, y_1\}, \ldots, \{x_N, y_N\}$. Here $x_i \in \mathbb{R}^m$ is a m dimensional feature vector and $y_i \in \{1, 2, \ldots, M\}$ is the equivalent class label. One against all technique makes $M$ binary SVM classifiers, each of them divides one class from all the rest of the classes. All the training samples of the $i^{th}$ class with positive labels, and all the other with negative labels are trained to the $i^{th}$ SVM. Then, the $i^{th}$ SVM solves the following problems that provide the $i^{th}$ decision

\[
\text{Minimize : } (w, \xi_j) = \frac{1}{2} ||w||^2 + \sum_{i=1}^{N} \xi_i^i \\
\text{Subject to : } \tilde{y}_j (w_i^T x_j + b_i) \geq 1 - \xi_i^i, \xi_i^i \geq 0,
\]

Where $\tilde{y}_j = 1$ if $y_i = i$ and $\tilde{y}_j = -1$ otherwise.

At the classification stage, a sample $x$ is classified as in class $i^*$ whose $f_{i^*}$ creates the biggest value

\[
i^* = \arg \max_{i=1,\ldots,M} f_i(x) = \arg \max_{i=1,\ldots,M} (w_i^T x_j + b_i). \quad (3.14b)
\]
4 Methodology

Extracting features of images and using these feature vectors for classification purpose are very common in the machine vision system. The proposed model is designed to work in a real-time environment where the tree harvester machine has cameras mounted at all sides. These cameras take videos while the autonomous machine travels from the forest to the transportation site. The drivers of this machine are the cameras by interpreting the captured videos in a series of images. These images are segmented using trained classifiers in order to check whether the path has trees or not. If there is a tree on the path the autonomous machine tries to look for another path otherwise it notifies the person for assistance. The following two block diagrams depict how the acquired images are classified by feeding their extracted features to the multi-class SVMs. The SVMs compares it with the stored training feature vectors. Then assign their respective class for each pixel of the test image. Finally post processing and performance evaluation undergoes.

Figure 4.1: Block diagram of training the classifier from stored dataset.
The block diagram in Figure 4.1 shows how the SVMs are trained with the feature vectors of stored training images. The second block diagram in Figure 4.2 shows the model working at real time environment. Series of images are captured as test images or video without the effect of blurring, since, the intention of this model is not to address a solution regarding to the effect of motions.

Figure 4.2: Block diagram of classifier in a real time environment.
4.1 Preprocessing

The image dataset was made after months of on-site capturing via digital camera. Since the sizes of the images in the dataset are too big for processing them directly, each image is down sampled to acceptable size while keeping the aspect ratio. The dataset have separate images for the training set and test set. As Figure 4.3 depicts, the training data consists of six different categories which are leaves, snows and bushes those belongs to the background and black tree, brown tree and white tree which belongs to the tree trunks.

![Image of categories](image)

<table>
<thead>
<tr>
<th>Bushes</th>
<th>Leaves</th>
<th>Snow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black trees (fir)</td>
<td>White tree (Birch)</td>
<td>Brown tree (pine)</td>
</tr>
</tbody>
</table>

Figure 4.3: the six categories of tree trunks and backgrounds.

Every class has 10 training images which are cropped from the original training images to sizes of either 10 x 10 for the backgrounds or 25 x 4 for the tree trunks. The regions selected for the tree trunks are mostly considering larger rows over the columns.
The above Figure 4.4 shows this clearly that a training region in a red box on the tree trunk has a size of 25 x 4 or 100 pixels. For the training region in blue box on the background has a square size of 10 x 10 or 100 pixels. Since, there are 10 trainings from 10 different images; each class will have 1000 pixels for training. Now the training sets are ready for the feature extraction steps. For the test images only down sampling is done and passes to the feature extraction steps.

4.2 Feature extractions

Feature extraction is the most important task in the classification process after preprocessing. In order to classify images first there must be dimensionally reduced relevant features from the image. These features must describe the image with a higher accuracy while reducing the complexity of the image. There are various mechanisms of extracting features in image processing. However, the model has preferred the texture and colour feature descriptors for tree classification. Since trees have unique texture information and adding the colour features could give higher accuracy of describing the relevant information. Accordingly, in the proposed algorithm the texture feature vectors are extracted using Gabor filter banks. Because, the 2-D Gabor elementary functions are suggested by Daugman as an accurate models of the cortical simple
cells in the mammalian visual system. Many successful applications such as face recognition, finger print identification, and character recognition were done by applying Gabor filters as a lower level computer vision system [31]. To get good result on the classification colour feature vector extraction must be included. This is done by using the first order statistical features which are described as follows.

**Gabor filter banks**

Once the image is pre-processed, then the image is filtered with the two dimensional Gabor filter banks to extract the texture features. Selection of Gabor filter parameters are based on the application to be used. After a series of experimental analysis, the parameters of Gabor filter banks are chosen. This carefully design is important to extract essential features from the complex image. Those features can define this complex image with a higher accuracy. The parameters used to design the Gabor filter banks are wavelength of 8 pixels, bandwidth of 1 octave, angular orientations that start from 0\(^\circ\) to 360\(^\circ\) with a separation angle of 30\(^\circ\), phase offset of 0\(^\circ\) and 90\(^\circ\), aspect ratio sets to 0.5 values. The filter kernel size is computed from sigma which is the standard deviation of the Gaussian factor expressed in pixels. From Eq. 3.7 and for bandwidth of one octave, sigma is expressed as \(\sigma=0.56\lambda\), where \(\lambda\) wavelength. The kernel size is \(2\sigma+1 \times 2\sigma+1\), substituting the above parameters; we calculated the kernel size to be 11 x 11. Since, the Gabor filters don’t extract the key features by themselves, magnitude response are applied on the filters output that gives us useful information. Briefly describing, for each orientation there are two convolutions due to the phase offset values. Superposition are done for each orientation by squaring the values of the convolution results for each phase offset values and then add them pixel wise and followed by a pixel wise square root computation[18]. Finally, applying a Gaussian smoothing filter where the envelope size is 1.7 times the Gabor filter. At this point the texture feature vectors are ready for further steps.

**Mean and variance**

Mean and variance is another statistical feature descriptors used as a colour feature extraction mechanisms in the classification of trees from the complex backgrounds. Colour feature vectors are extracted for each image on different colour spaces (RGB, YCrCb) so as to present which colour space favours the classification process. For RGB colour space
there will be 18 feature vectors where the six colour feature vectors are from mean and variance the rest 12 texture feature vectors are from the 12 Gabor filters output. In case of YCrCb colour space the model only considers the CrCb information so as to suppress the problem occurring due to the variation in illumination. Therefore, in the CrCb colour space we have a total of 16 feature vectors where four are for the mean and variance each having two feature vectors and the other 12 texture feature vectors are from the 12 Gabor filters output.

### 4.3 Multiclass Linear SVM

SVM is primarily designed for binary classification. A number of methods have been proposed for multi-class SVMs, and the foremost approach is to split the single multiclass problem into multiple binary classification problems [23]. While training the classifiers, random sequence of class has been made. The linear SVMs designed on this model works as multi-classifier that uses the commonly known one-vs-the rest approach where for each class a classifier is trained for that class against the rest of the classes. Each classifier defines a discrimination function that should assume positive values when the cases belong to the single class and negative values to the rest. These values are then matched up together; and the output of the combined classifier is that class where the value of the discriminating function is the largest [3].

With this principle the trainings are done with the extracted feature vectors of the training images selective region. Training images are chosen from the most frequently found tree trunk types (black, brown and white) and for the complex backgrounds (bushes, leaves and snow). Since there are six different classes, the multi-classifier trained six binary linear classifiers those estimates the linear coefficients (weight vectors and bias or threshold to the margin), scaling factors and shifts for each feature vector per binary classifier.

The test images’ feature extractions are done in the same fashion as the training images. Then these feature vectors are fed to the multiclass classifiers to classify the pixels of the image to the tree types or backgrounds. The outputs of the classifiers are labelled to binary images. Where the first is 2-dimensional image and the second is 3-dimensional image. In the 2-D image, pixels that belongs to the tree trunk are labelled as one and those belongs to the backgrounds are labelled to 0. In the 3-D
image colour labelling are done on each tree categories to distinguish them among each other, whereas the backgrounds are labelled to black colour. The test, classified and colour labelled images are shown in Figure 4.5 (a), (b) and (c) respectively.

![Figure 4.5](image)

Figure 4.5: Classified image containing both the tree and misclassified backgrounds

There could be misclassification on some part of the images and these are reduced by applying post processing that gives acceptable accuracy.

### 4.4 Post processing

In this model the post processing is done by two measures based on mathematical morphology. The measures are the “Area” and “Bounding box” which can be extracted directly from the object. Areas are the actual number of pixels inside each connected components and the bounding boxes are the smallest possible rectangles that inbound each component in the image [15].

For each connected component on the classified image, the area and bounding box are calculated. To remove the smaller components in the image as seen in Figure 4.7, two thresholds are selected one for the area of connected components by applying histogram on the whole areas of the classified image as shown in Figure 4.6 and the second one is calculating ratio \( r \) of height \( h \) over width \( w \) for the bounding boxes of each component in the image.

\[
r = \frac{h}{w} \quad (4.1)
\]
Components in the image that have greater area and ratio compared to the two thresholds are the interesting part of the image and displayed as foreground (tree trunks) where as the rest are considered as background. It can be seen on Figure 4.8 more clearly.

![Figure 4.6: Histogram of Areas found in the image](image)

As shown in Figure 4.7 below there are more than 72 connected components. However, the connected components that show trees are eleven in number. After post processing step as seen in Figure 4.8, it is able to remove all the unwanted components or the misclassified background as tree parts. Out of eleven trees only seven trees are retrieved, where the rest of very small trees or trees very far from the machine are considered as background. Therefore, a thread off are placed in order to keep most of the trees while removing the misclassified background.
Finally, distinguishing the different classes of tree trunk is done by labeling them with colours. Each tree trunk category has specific colour where as the backgrounds are consolidated to be one colour. The general labeling rule is stated in Table 4.1 below.
Classes of tree trunks and backgrounds | Labeling colours
---|---
Black tree (Firs) | Red
Brown tree (Pine) | Blue
White tree (Birch) | Green
Bushes, Leaves and Snow | Black

Table 4.1: Colour labelling for different classes of tree trunks and background

### 4.5 Performance Evaluation

The performance of the segmentation is evaluated by taking some samples of the test images and segmenting them manually with higher accuracy. After segmentation, the image is changed to binary image only to have 0 for the background and 1 for the foreground (tree trunks). Then, the Sum of Absolute Differences (SAD) is calculated by subtracting the binary pixel values within a square window between the manually segmented image $I_1$ and the classified image $I_2$ followed by the summation of absolute differences within the square window. If the classified output matches the manually classified image, the resultant will be zero. This implies perfect classification [10]. Normalization is done by dividing the SAD with respect to area of the image as shown in Eq. (4.2)

$$SAD_{Error} = \frac{1}{m \times n} \sum_{i,j=1}^{m,n} |I_1(i,j) - I_2(i,j)|,$$  \hspace{1cm} (4.2)

The overall performance of the model is quantified by averaging the entire N test images $SAD_{Error}$ which is expressed below:

$$\text{Overall performance} = (1 - \frac{1}{N} \sum_{k=1}^{N} SAD_{Error_k}) * 100,$$  \hspace{1cm} (4.3)
5 Proposed Hardware Architecture

Recently, hardware systems that incorporate FPGA have been used widely for fast prototyping platforms, developing coprocessors and custom computing machines. FPGAs are appropriate for accelerating tasks that need processing of data with non-standard formats and cyclical execution of small operations. A system that runs on reconfigurable FPGA hardware has got several advantages. Hardware based executions are orders of magnitude faster than the corresponding software systems that carry out the same task for some applications. FPGAs reconfigurable nature has the advantage of running different applications at different period, that reduces the total number of components required in a given hardware platform. Implementing new versions of design are simply downloading configuration bit streams. More functions can be added and maintenance can be done as needed. Similarly, the systems can be made scalable [32].

The tree recognition system modelled on this paper is scaling and rotation invariant that takes advantage of the Gabor filter robustness. However, to run this system at real time environment the hardware architecture design must consider the massive computations arise from these filter banks, the dot product of the linear multi-classifiers to classify the trees from the backgrounds and post processing to remove misclassified backgrounds and label different tree types. Currently, there are different types of hardware processors available based on the application to be used, among them FPGA is selected as real time hardware processor for this model due to its capacity of handling large amount of arithmetic operations on dedicated DSP slices, have on chip memory, programmability at hardware level for higher parallelization and the customization of data transfer through pipelining [27][28].

The hardware architecture is shown in Figure 5.1. A camera acquires video data in the form of frames and each frame (image) is buffered as input in row by row basis to the on-chip memory of the FPGA where ease of access to the main process can be achieved. To match the size of the filter window K x K, the buffering is made in a window based approach where K rows have K-1 cascaded registers followed by a block
RAM configured as FIFO. The stored pixels in the FIFO used for later windows until they reach the end of the bottom row of the window [26]. For image size of 512 x 512 x 3 and filter kernel of 11 x 11 window, the FIFO must have a depth of 501 byte per channel of the input or 4Kbits storage capacity per channel.

The coefficients of the 12 Gabor filters, two mean and variance filters with 11x11 window sizes are stored as 2D array in the SDRAM memory. Moreover, the scaling factors and shifting factors, the coefficients of six linear classifiers are also stored in the SDRAM. The instructions for the processes and all the coefficients are transferred from the SDRAM to the cache memory when the system powers on. These ensure high speed access of instructions and kernel coefficients by the main image processes and it helps to achieve low latency and high throughput on the overall system.

![Figure 5.1: Tree identification system structure](image-url)
At stage 1, for a window size of 11x11, the texture feature extraction have been done by the 12 Gabor filter banks with 30° orientations from 0° to 330°. Each Gabor filter bank is convolved with the gray scale of the test images. It is obviously known that the 2D convolution is computational intensive as it undergoes multiplication of each image’s pixel with the kernel window. The camera acquires video of the forest which is fragmented to series of frames with a size of 512x512x3 image. For each pixel per channel of a frame calculates 121 multiplications and accumulations to extract a single feature. This way the processor must quantify 3.2 x 10⁷ multiplications and accumulations to extract a single orientation feature of 512x512 size gray scale image. For the 16 features it will be unpredictable to achieve real time operation in serial processing. These large amounts of computations require a good choice of hardware processor that could use all the available resource to run at real time. All the latest FPGAs have dedicated DSP slices that handle large amount of multiplications with higher rates. From the Xilinx Virtex 6 FPGA family XC6VSX475T device has 2016 DSP slices and 1064 block RAMs of size 36Kb or it can be two independent 18Kb blocks [29]. It is crucial optimizing the available resources when designing the hardware architecture especially the 2D filters. Parallelization is a superior approach to increase the performance of the model. Nevertheless, the DSP slices available on this FPGA are still resource limited. Mathematical modification to the 2D convolution promotes the DSP slices to work in parallel.

### 5.1 Modified 2D Filters Convolution

Optimization can be done by taking advantage of the symmetrical property of the Gabor kernel elements. When the input image convolves with a symmetrical kernel, the pre adder introduced in the DSP slices adds two symmetrical elements of the kernel before the multiplication. In contrast, for anti-symmetric kernels the DSP slice pre subtracts anti-symmetric elements of the kernel before the multiplication. For a [K x K] kernel size Gabor filter bank the modified 2D convolution reduces the multiplication operations from $K^2$ to $\frac{K^2+1}{2}$ that saves 50% of the DSP slices. To reduce the bandwidth bottle neck on the accumulator cascading two multiplier outputs can be done which reduce the accumulators needed from $\frac{K^2+1}{2}$ to $\frac{K^2+1}{4}$. A rough estimation for the number of DSP
slices necessary for the 12 orientations of the modified 2D convolution are based on the following equation [26].

\[ N_{ds} = n_{\theta} \times \left( \left( \frac{k^2 - n_z}{2} + \frac{k^2 - n_z}{4} \right) - 1 \right) \]  

(5.1)

Where \( N_{ds} \) is the DSP slices needed, \( n_{\theta} \) is the number of orientations and \( n_z \) is the number of zero values in the kernel. Based on Eq.(5.1) for each pixel 1077 DSP slices are used for the 12 Gabor filters feature extraction.

The colour feature extraction on the Cb and Cr channels are done using the 2D mean and variance local filters. Taking advantage of the separable and symmetrical properties of these 2D filters as described in Figure 5.2. and 5.3 the vertical filter takes the line buffer data and use pre adder to add symmetrical values of the buffered line before multiplying with the 1D vertical filter. Then after multiplication accumulate them to form intermediate input to the 1D horizontal filter. Where, these accumulated results are filtered by the horizontal filters after number of register delays that ensure the relations across the filtering process [25].

For K x K 2D separable and symmetrical filter K+1 multiplication are required. Storing each kernel element requires 24 bits on fixed point representation basis. Then, for each kernel window of 11 x 11 size requires 3Kbits of RAM. In the proposed hardware the block RAM has a
size of 36Kbits. It can be used as two independent 18Kbits block RAMs. Table 5.1 depicts the DSP slices and block RAMs required to extract features of a pixel when applying all the filters in parallel at synchronized way.

<table>
<thead>
<tr>
<th>Type of 2D Filters</th>
<th>Number of filters</th>
<th>DSP slices</th>
<th>Block RAMs (3Kbits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gabor filters with 11 x 11 window size</td>
<td>12</td>
<td>1077</td>
<td>12</td>
</tr>
<tr>
<td>Mean filter with 11 x 11 window size</td>
<td>2</td>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>Variance filter with 11 x 11 window size</td>
<td>2</td>
<td>22</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 5.1: quantities of DSP slices for the 2D filters

At stage 2, the extracted feature vector [1 x 16] of each pixel is scaled by [1 x 16] scaling factors and shifted with shift factors of [1 x 16] vector. Each vector has different scaling and shifting factor. These scaling factors are temporarily stored in cache memory using an on chip block RAM of the FPGA. Since, scaling is the same as multiplying with some fixed values, The DSP slices used to quantify scaling of a pixel’s vector are 16 and for the shift operator it can be done on the logical operators.

As shown in Figure 5.1, at the third stage the coming pixel is classified using six different binary linear classifiers. The classifiers coefficients are copied to the cache memory from SDRAM parallel to the future extraction stage. Each linear classifier is expressed as a dot product of the weight vector [1 x 16] and the output vector [1 x 16] of scaling and shifting stage and finally adds a scalar bias value. On FPGA, the vector dot product are quantified by multiplication of each scalar values followed by accumulation. Therefore, each classifier will use 17 DSP slices and for the six classifiers it will be 102 DSP slices. As Figure 5.4 depicts, to reduce the bottle neck on the bandwidth of the accumulator the DSP slice’s adder is used to add two multiplication results.
The outputs of all six classifiers are compared using logic blocks and labelling is done on the classified pixel based on its class.

To remove the misclassified background a post-processing stage is applied. In Figure 5.1 shows connected component’s areas and bounding boxes are quantified in parallel. As described in the work of [29], connected components labelling are based on four neighbourhood connection of pixels. In a classified frame there could be many connected components and their labeling are unique among each other. In the figure 5.4(a) shown below the pixel P5 labeling inherited from its neighborhood pixels. As depicted on Figure 5.4 (b), a one line FIFO buffer and two registers are used as delay to hold recursive data dependency caused by the neighborhood of formerly labeled pixels [29]. Where, the size of the line buffer is equal to length of frame row $N^c - 2$. One block RAM is necessary for the buffer. Brief explanations on the labeling, area and bounding box calculation are found in [29].
There is a labeller which gives label codes for each pixel of the connected components in the frame. Label pairs are sent to equivalence table when neighbouring labels match and these labels are merged in order to map linked lists of equivalent labels onto linear memory array. Once the equivalence table is resolved the area and connected components calculation can be done. Since the data of labelled components are accumulated in parallel with the labelling process, a sequencer controls the merging and computation process. Continuous scans are applied by the sequencer on the equivalence table until all the connected components area and bounding box computations are finished [29]. As stated on section 4 of this paper, the area of connected component is the sum of the number of pixels found in a single connected component. To quantify the bounding box, the coordinates are stored, which is the smallest rectangle possible that inbound the object as shown in Figure 5.5. From these coordinates we can get the ratio of height over width.

![Figure 5.5: Bounding box](image)

Now we can remove the entire small objects that are lower than the threshold of area $T_{\text{area}} = 450$ and bounding box $T_{\text{bounding box}} = 3$.

The memory required to store the computed area of connected object dependent on the biggest connected object size. If the object size is $B \times C$ then the number of bits required is expressed by Eq. 5.2

$$\text{number of bits} = \log_2(B \times C) \tag{5.2}$$

On a 512 x 512 image, let’s say the maximum connected object is a tree of size 500 x 60 then to store the area of a single object we require 15 bits and to store a complex image’s connected components considering a depth of 1024 that will be 15Kbits. The same is true for the bounding box calculation except replacing the $(B \times C)$ multiplication with the maximum images’ column or row. For 512 x 512 image the maximum is 512 and 10 bits are used to store the maximum rows and a total of 10Kbits are used to store each corner parameter. This means 4 by 16Kbits block RAM for the four parameters of bounding box [29]. Since there are two
data tables kept, the required memory is doubled and 4 block RAMs of 16Kbits size are needed for the equivalence table.

The overall summary of the estimated hardware resources required for tree identification based on the significant resources of DSP slices and block RAMs is shown in Table 5.2. Where the block RAM size is 36Kbits and it can be divided into two 18Kbits and accessed through dual port.

<table>
<thead>
<tr>
<th>Processing stages</th>
<th>DSP slices</th>
<th>Block RAMs (36 Kbits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input one line buffer</td>
<td>--</td>
<td>3</td>
</tr>
<tr>
<td>Feature extraction</td>
<td>1121</td>
<td>14</td>
</tr>
<tr>
<td>Scale and Shift</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>Linear Multi-classifiers</td>
<td>102</td>
<td>6</td>
</tr>
<tr>
<td>Post-processing</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1240</strong></td>
<td><strong>34</strong></td>
</tr>
</tbody>
</table>

Table 5.2: Estimated hardware resource usage of tree identification model
6 Results

In order to analyze the influence of feature descriptors, colour spaces on the classification of the test images, Experiments are conducted on the feature descriptors separately and in combination. Where, the window sizes of the feature descriptors are 11x11 for the texture extraction and 11x11 for the colour extractions. The evaluation is done on two data sets where one data set is collected in autumn and the other data set is from the winter time. Each data set has separate training set (10 images) and test set (10 images). The experiments are made twice for each colour spaces (RGB, CbCr) on the test sets. The representation for the fusion of the feature descriptors are FD1, FD2, FD3, FD4, FD5 and FD6. In Table 6.1 it is specified clearly and only these representations are used throughout the result report.

<table>
<thead>
<tr>
<th>Colour Space</th>
<th>Representations</th>
<th>Feature descriptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB</td>
<td>FD1</td>
<td>Gabor filter</td>
</tr>
<tr>
<td></td>
<td>FD2</td>
<td>Mean and variance</td>
</tr>
<tr>
<td></td>
<td>FD3</td>
<td>Gabor filter and mean and variance</td>
</tr>
<tr>
<td>CbCr</td>
<td>FD4</td>
<td>Gabor filter</td>
</tr>
<tr>
<td></td>
<td>FD5</td>
<td>Mean and variance</td>
</tr>
<tr>
<td></td>
<td>FD6</td>
<td>Gabor filter and mean and variance</td>
</tr>
</tbody>
</table>

Table 6.1: Symbolical representation for the fusion of Feature Descriptors

The performance of classification is quantified by SAD between the classified image and the manually classified image. The SAD is then normalized with respect to the area of the whole image. These are done for the ten test images. Then, the performances of each classification were averaged according to Eq. 4.1 and the results of the performances are shown in Table 6.2.
Table 6.2: Classification performance for single feature descriptors and in fusion

The experimental analysis described in Table 6.2 shows that colour and texture descriptors alone does not perform well compared to their combinations both in the RGB and CbCr colour spaces. The fusion of mean and variance and Gabor filter in CbCr colour space performs better than the feature descriptors alone. The result in Table 6.2 also shows that colour has higher performance than texture when it comes to uniform illumination. But for images with non-homogeneous illumination, texture is robust than colour descriptors.

The presented result in Figure 6.1 and 6.2 also depicts that the CbCr colour space usually performs slightly higher than the RGB colour space for all test images; when the two feature descriptors are used in combination. Apart from the higher tolerance on non-homogeneous illumination by Gabor filters, the removal of luminance channel in the YCbCr colour space has also shown significant change in the performance of the classification.
Figure 6.1: Performance plot for the first 7 test images with 6000 training pixels

Figure 6.2: Performance plot for the first 7 test images for 3000 training pixels
Extensive analyses are done in sizing the training pixels for maintaining higher performance of classification. The average estimated training vector size must not be lower than 1000 pixels per class for RGB and 500 pixels for CbCr. However, further decrease in the size of the training pixels per class leads to a lower performance. The analysis is displayed on Figure 6.3 below. Where, the 6000 training pixels have relatively uniform performance throughout all the test images compared to the rest of the two training sizes.

![Figure 6.3: Performance comparisons among different training pixels size per class in CbCr colour space](image)

Table 6.4 summarizes the performance result on the highly performed fusion based classifiers FD6 with respect to the number of training pixels.

Table 6.4
Table 6.4: Summary of performance of FD6 based classifier with respect to the training size

As it can be seen in Table 6.4, the larger the training pixels provide the higher performance on classification. However, increasing the training pixels further does not show much improvement; in reverse it introduces misclassification of pixels. Therefore, a trade-off must be kept between the size of the training pixels/class and the performance of classification. The processing speed for classifying 512x512 image is 32.4 seconds. Where, the simulation was done using Matlab software that runs on ASUS Intel core i5 2.5GHz CPU laptop.

The following classified images in Figure 6.4-6.8(a, b, c) are some examples of the output of the tree identification system. All the classifications are done using the highly performed FD6 based multiclass SVMs. Where Figure 6.4 (a) shows the test images, Figure 6.4 (b) shows the trees classified from the backgrounds and Figure 6.4 (c) shows identification of tree among the different tree categories by its colour.
Figure 6.5 (a)                                Figure 6.5 (b)                        Figure 6.5 (c)

Figure 6.6 (a)                              Figure 6.6 (b)                            Figure 6.6 (c)

Figure 6.7 (a)
Figures:

- Figure 6.7 (b)
- Figure 6.7 (c)
- Figure 6.8 (a)
- Figure 6.8 (b)
- Figure 6.8 (c)
7 Discussion and Conclusions

7.1 Discussion

The designed tree identification system is able to classify trees from backgrounds in an image. Even though identifying trees in an image is difficult due to the complexity of backgrounds, non-homogenous illumination, shadow, the variations in the bark’s colour and texture even in a single tree and moreover the appearance of the same tree changes with a change in season. To tackle the problem arises due to the non-uniformity in appearance; the designed system incorporated a higher level of machine vision algorithms than the commonly used threshold and edge detection approaches. The developed pixel based multiclass SVMs can handle all the above stated problems. The developed system is able to find the exact boundaries of the tree trunks. The advantage of using pixel based classification lead to a better estimation of the boundary of the tree trunks and reduces the misclassification of pixels that arises from the block based classification approach by Wajid [1].

The manual selection of training set could have problem on the performance of the classification results. Therefore, the training set must be prepared thoroughly. Then again, problem arises to distinguish trees reliably among each other when dense forests are behind the trees. These could be solved by changing the monocular camera vision system to a stereo vision system.

As described in the result section of this thesis report the experiment have made comparison of performances between two feature descriptors alone and in combination, the influence of using different colour spaces for feature extraction and the multi class SVM approach to classify trees from backgrounds. And the results specified shows that colour and texture features combination have higher accuracy of classifying each image pixels to its respective class that could be to the trees or to the backgrounds. The study shows that the Gabor filter and the mean and variance descriptors combination at CbCr colour space has a classification performance of 95.33% and the second best classifier is the combination of Gabor filter and mean and variance combination on RGB
colour space is 93.76%. The result from wajid[1] shows the combination of colour histogram and gabor filter on HSV colour space has performance of 86% and for the RGB colour space it is 84%. The performances are calculated using the normalized SAD. The significant robustness of Gabor filter to illumination changes and image noise has improved the performance of the fusion based trunk identification system.

Even though the forest environment is highly complex, combining the binary linear SVMs to classify trees from backgrounds have shown promising results that could be applied in real time system. The coefficients of these linear classifiers are quantified by training the classifiers offline on workstation PC using Matlab 2012a software. In real time application scaling and shifting must be applied on each feature vector of the test image based on the scaling factors and shifting factors of the training feature vectors. Classifying trees with linear multi class SVMs can have ease of implementing the model on the FPGA hardware, if the architecture is designed to work in parallel processing mechanism. The hardware processing speed, memory size and power consumption to implement the designed model on FPGA will be smaller since the classification will be dot product of two vectors and adding a scalar threshold value to categorize between two classes. The speed may also be optimized by developing an algorithm in FPGA. The algorithm processes the colour and texture feature extractions in parallel for the test images that are acquired at real time environment.

7.2 Conclusions

The tree trunk identification system proposed on this paper is based on linear multiclass SVM classification approach. Different attempt has been made to come up with the best methodology. The combinations of colour and texture features are more effective than any single kind of feature in classification of trees from complex backgrounds. The experimental results demonstrated that the combination of mean and variance and Gabor filter features achieve the best classification performance of 95.33% in CbCr colour space. Pixel wise classification of forest images attained better performance of classification. The empirical study has also shown promising result for the recognition of tree trunks in the forest using multiclass linear SVMs that could be applied in real time environment. However, further research has to be made on distin-
Automatic recognition of tree trunks in images – Robotics in forest industry
Tamrat Gebremedhin

7 Discussion and Conclusions

7.3 Future works

Problem arises to distinguish trees reliably among each other when dense forests are behind the trees. These could be solved by changing the monocular camera vision system to a stereo vision system. The distance between the tree and the forwarder machine can also be quantified by a triangulation method. Based on the estimated hardware cost real time trunk recognition system can be developed on FPGA using VHDL.
Discussion and Conclusions
References


References


[18] Simona E. Grigorescu, Nicolai Petkov, and Peter Kruizinga, “Comparison of Texture Features Based on Gabor Filters” IEEE transactions on image processing, vol. 11, no. 10, October 2002


[21] Pawan Lingras, Cory Butz, “Interval Set Representations of 1-vr Support Vector Machine Multi-classifiers”, Saint Mary’s University, Halifax, Nova Scotia, Canada, University of Regina, Regina, Saskatchewan, Canada


Appendix A: Classified images

Training set samples

Trunk 1

Trunk 2

Bushes
<table>
<thead>
<tr>
<th>Leaves</th>
<th>Snow</th>
<th>Trunk 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Leaves" /></td>
<td><img src="image2.png" alt="Snow" /></td>
<td><img src="image3.png" alt="Trunk" /></td>
</tr>
</tbody>
</table>

**Appendix A: Classified images**

**2014-01-23**
Test images and their classified output consecutively
Fel! Hittar inte referenskälla.
Fel! Hittar inte referenskälla.
Fel! Hittar inte referenskälla.

Appendix A: Classified images
2014-01-23
Fel! Hittar inte referenskälla.
Fel! Hittar inte referenskälla.
Fel! Hittar inte referenskälla.

Appendix A: Classified images

2014-01-23