Gradient Based Power Line Insulator Detection

Submitted By

MD.Raihan
ID: 2012-2-60-040

S.M.Imran Hossain
ID: 2012-2-60-012

Kuntal Ghosh
ID: 2012-2-60-001

Supervised By

Dr. Taskeed Jabid
Assistant Professor
Department of Computer Science & Engineering,
East West University

A project submitted in partial fulfillment for the degree of Bachelor of Science in the Department of Computer Science and Engineering
East West University

August, 2017

Declarations

The project has been submitted to the Department of the Computer Science & Engineering, East West University in the partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science & Engineering. We hereby, declare that this project has not been submitted elsewhere for the requirement of any degree or diploma or any other purposes.

Signature of the students

…………………………..
(Md.Raihan)
(ID: 2012-2-60-040)

…………………………….
(S.M.Imran Hossain)
(ID: 2012-2-60-012)

…………………………….
(Kuntal Ghosh)
(ID: 2012-2-60-001)

Letter of Acceptance

This project is entitled “Gradient Based Power Line Insulator Detection” submitted by: Md.Raihan ID:2012-2-60-040, S.M.Imran Hossain ID:2012-2-60-012 and Kuntal Ghosh ID:2012-2-60-001 to the department of Computer Science & Engineering, East West University, Dhaka, Bangladesh is accepted as satisfactory for partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science & Engineering on August 2017.

Supervisor

………………………………………….
Dr. Taskeed Jabid
Assistant Professor
Department of Computer Science & Engineering
East West University, Dhaka, Bangladesh.

Chairperson

………………………………………….
Dr. Md. Mozammel Huq Azad Khan
Chairperson and Professor
Department of Computer Science & Engineering
East West University, Dhaka, Bangladesh.

Acknowledgement

First of all, we would like to thank almighty Allah for giving us strength, patience and knowledge to complete this project work.

Our most heartfelt gratitude goes to our beloved parents for their endless support, continuous inspiration, great contribution and perfect guidance from the beginning to end.

We would like to express our sincere gratitude to our supervisor Dr. Taskeed Jabid for the continuous support, for his patience, motivation, and immense knowledge. His guidance helped us in all the time of our work. We could not have imagined having a better supervisor and mentor for this project.

Abstract

Insulators are widely used in the power system to provide electrical insulation and mechanical support for high voltage transmission lines. Detecting and localizing the insulators automatically are very important to intelligent inspection, which are the prerequisites for fault diagnose. A method for insulator detection in the image of overhead transmission lines based on Histogram Oriented Gradient(HOG) is presented in this paper.

Dividing our work into two phases- electrical pole detection and power line insulator detection. In the first context, present a line based approach for automated detection of electrical poles. The detection is performed by first applying edge detection algorithm. Then, Applying Probabilistic Hough Transform to extract line information from the image. We Prepare data by considering only vertical lines and a variation in angle of ±5-degree as an empirical value. Group the pre-processed data and find the coverage area by applying a Heuristic function. Take the best three coverage group which consists of lines that create three individual lines. In those best three lines at least one line detects a pole which maximize the coverage area.

After detecting electrical pole, we go to the top of the pole and take a zoom in picture where we search for insulator. Before test the image we take different insulator and non-insulator images as our training Dataset. To train a classifier using support vector machine in its LIBSVM tools, extract HOG features of training Dataset. As classifier have been trained we apply Sliding-window object detection technique for identifying and localizing insulators in an image. The approach involves scanning the image with a fixed-size rectangular window. Extract HOG features of the sub-image defined by the window and apply classifier to check that the window bounds an insulator or not. The process is repeated on successively scaled copies of the image so that objects can be detected at any size. To detect angled insulator, rotate the image into 360-degree and apply sliding window technique with image pyramids to detect insulators at varying scales and locations in the image. Sliding window technique detects some non-

insulator images as an insulator called false positive. To remove false positive Hard Negative Mining is applied and re-train classifier using those false positive samples which improve the performance of the classifier from an initial run of the classifier. Usually non-maximal suppression is applied to the output to remove multiple detections of the same insulator.

Experimental results showed that the method could extract the insulator from the image precisely, and it was suitable for many practical applications such as insulator fault diagnosis, insulator contamination grade determination and so on. Compare to the other Features like -LBP, experimental results indicate that HOG based feature detect insulator more effectively and accurately and detect insulators in different angle under complex background.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Monitoring the status of Insulators on power lines is one of the most important jobs in power system operation. Insulators play an important role in power system safe operation. It is usually used to fix the transmission lines. Since it has to bear high mechanical tension and high voltage, it is easily to be damaged. Defects in ceramic insulators like broken, cracked and punctured discs give rise to the initiation of partial discharge (PD) activities within the samples which will threaten the safety of the electric power [1]. Hence Inspecting the status of the insulators is needed to identify such defective samples as early as possible so that appropriate replacement strategies can be devised.

To improve the efficiency of inspection, the traditional manual inspection technology is being replaced by new technology. Computer vision technology can quickly and efficiently detect defects of insulators, and greatly reduce the work load by automatically detecting and segmenting the power line insulators. The captured images often include various cluttered backgrounds such as mountains, rivers, grassland, and farmland thus, the processing of original images is complicated, which will easily lead to a wrong result [2].

In our project we, proposed a system that will detect the insulators and will help to detect faulty insulators more efficiently.

1.1 Motivation

The world economy is increasing day by day. With the economic growth, the demand for the power consumption has been increasing gradually. High voltage power lines and transmission systems become more and more important with the raising demand of energy. The distance over which electrical power is transmitted has been continuously increasing along with system voltage levels. This has led to a significant increase in the number of insulators in use on power lines.

Insulators are used to isolate the naked power lines of the power transmission lines and to support the lines mechanically. Since the service life of the individual insulator making up these strings is difficult to predict, they must be tested periodically to maintain adequate line reliability at all times. It’s a challenge to test insulators periodically by using man power [3].

Many paper has been published in recent years to reduce faultiness after detecting insulator using computer vision. They concentrate on insulator detection by using Deep Neural Network, Naïve Bayes classifier, support vector machine etc. Now, we want to use a new SVM tool called LIBSVM which will help in power line system by detecting insulator.

1.2 Objective

The objective of our work is to take computer vision technology one step forward in the sector of power lines by representing a computer vision based system to detect power line insulators. We are studying in power line insulator representation on histogram oriented gradient (HOG) to automatically detect insulator for finding its faultiness where a set of image's features are trained by LIBSVM tool.
We want to design a system that will help to detect the injury of insulator for monitoring the status as well as want to improve the diagnosis efficiency. we want to test the system using a set of positive and negative image to detect insulators more effectively and efficiently. We hope that our approach significantly outperforms in the power line system in term of both accuracy and efficiency.

.1.3 System Procedure

In our work, we proposed a computer vision based approach to detect power line insulator. We divided our work into two phases. In the first phase, we detected electrical pole and in the second phase, we detected insulators associated with electrical poles.

First, we took a close picture in a certain region where we would find a pole. To detect pole first, we extracted lines from the image. To do that we applied canny to find edge of the image and then applied Hough Transform to extract lines. As pole is a vertical object so we took lines only which were vertical. In real life, poles are not always vertical, sometimes it seems skewed so we took a variation in angle of ±5-degree for grouping lines. After that extracted lines are grouped in such way so that their end point stays close. After grouping, we calculated the coverage area of each group by using a Heuristic function. we selected the best three groups whose coverage area were maximum. As groups were consisting of lines so we found the best coverage lines that cover the most area in the image. In those best three lines we had our electrical pole.

In the second phase, we looked for the top of the pole as well as took a zoom picture and this is our test image. To classify insulator from test image we trained a classifier with a set of insulator image and non-insulator image using LIBSVM by extracting feature value based on Histogram Oriented Gradient (HOG).

Utilizing both a sliding window and an image pyramid we were able to detect insulators in images at various scales and locations. We applied Hard-negative mining to remove false positives and used non-maximal suppression method to the output for removing multiple detections of the same Insulator.

Pole Detection: Procedure

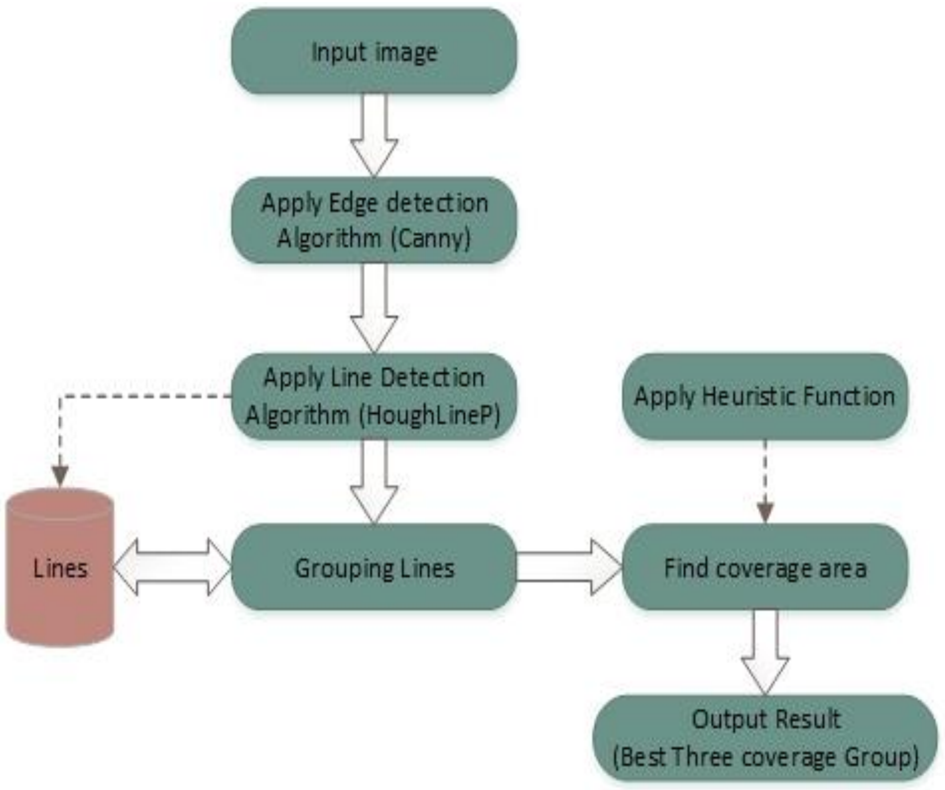

Input image
Apply Edge detection Algorithm (Canny)
Apply Line Detection Algorithm (HoughLineP)
Lines
Grouping Lines
Apply Heuristic Function
Find coverage area
Output Result (Best Three coverage Group)

Figure 1.1: Pole Detection Procedure

Insulator Detection: Procedure

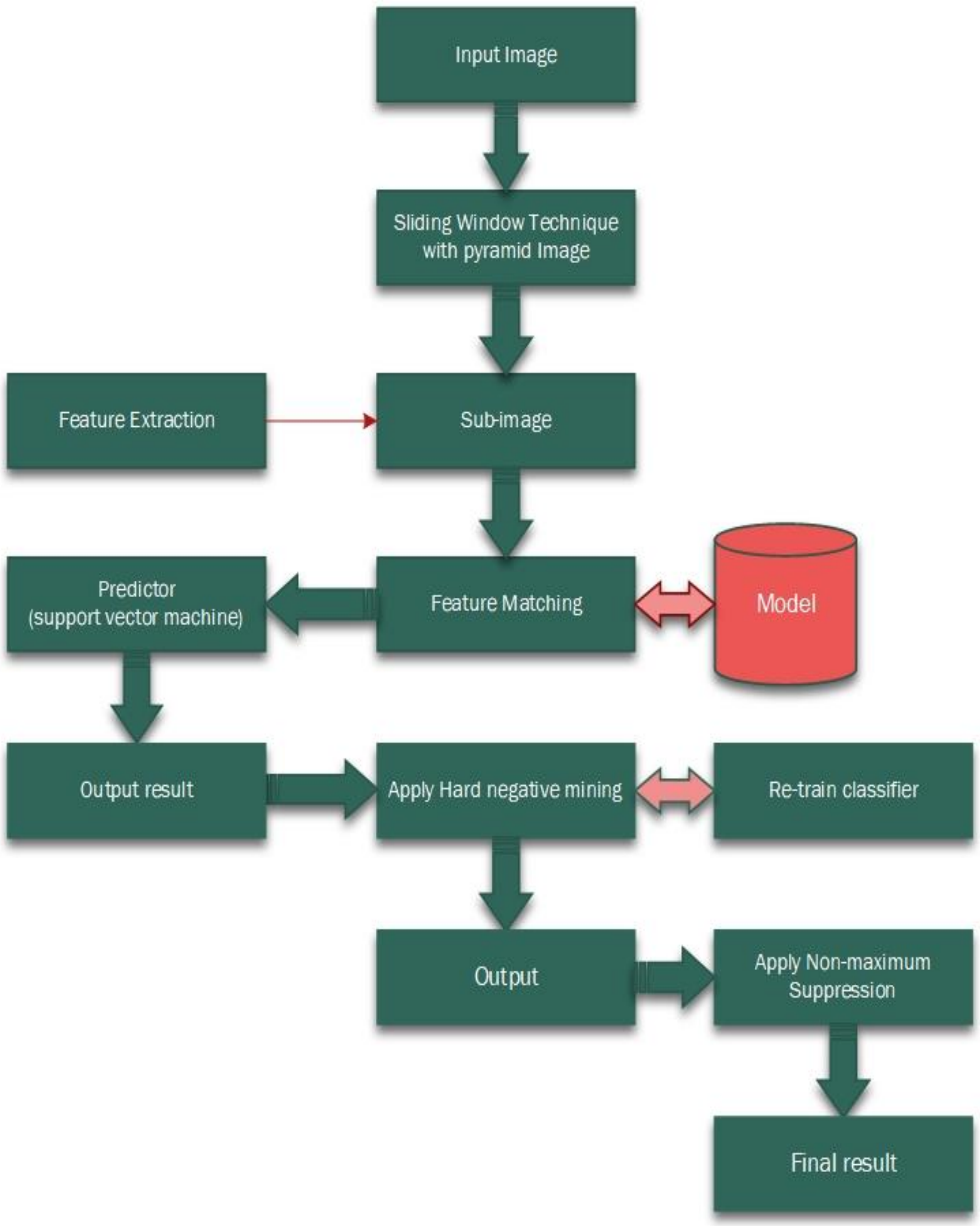

Input Image
Sliding Window Technique with pyramid Image
Feature Extraction
Sub-image
Predictor (support vector machine)
Feature Matching
Model
Output result
Apply Hard negative mining
Re-train classifier
Output
Apply Non-maximum Suppression
Final result

Figure 1.2: Insulator Detection Procedure

Chapter 2

Existing Work review

As a scientific discipline, computer vision is concerned with the theory and technology for building artificial systems that obtain information from images or multi-dimensional data. A significant part of artificial intelligence deals with planning or deliberation for system which can perform mechanical actions such as moving a robot through some environment. This type of processing typically needs input data provided by a computer vision system, acting as a vision sensor and providing high-level information about the environment and the machine [4].

In this chapter, we discuss about the pole detection and insulator detection early work. We discuss the pole related existing work in section 2.1 and insulator related work in section in 2.2. We discuss Rigid kernel based detection of viola-jones work in section 2.3. The classification methods and related processes which we used in our proposed system that describe in section 2.4.

2.1 Existing work On Pole

A electrical pole or utility is used to support overhead power lines and various other public utilities, such as electrical cable, fibre optic cable, and related equipment such as transformers, insulator and street lights. It can be referred to as a transmission pole or power pole depending on its application. Electrical wires and cables are routed overhead on utility poles as an inexpensive way to keep them insulated from the ground and out of the way of people and vehicles. Electrical poles can be made of wood, metal, concrete, or composites like fiberglass. They are used for two different types of power lines; sub-transmission lines which carry higher voltage power between substations, and distribution lines which distribute lower voltage power to customers [5].

There are many approaches to detect pole like Image Analysis-Based approach [6], Graph Cut approach [7], LIDAR Data based approach [8], urban point clouds approach [9], Analysis-Based approach focuses on the shape of the pole. Graph cut for image segmentation [10,11] is a newly developing graph based image segmentation technique**.** Other approaches also describe different method for detecting pole.

2.1.1 Image Analysis-Based Approach

From the remote surveillance video of any wide disaster area that is fairly long, it is important to extract key frames that contain specific component structures of the power grid. The key frames can then be analyzed for possible damage to the specific structure. In his context, they present an algorithm for automated detection of utility poles. Specifically, they show robust detection of poles in frames of videos available from various sources. The detection is performed by first extracting 2D shapes of poles as analytically defined geometric shape, quadrilateral, whose edges exhibit noise corruption. A pole is then detected as a shape-based template, where one long rectangular trapezium, is perpendicularly intersected by at least one trapezium representing a cross arm that suspends the conductors. Via testing and comparison, their algorithm is shown to be more robust as compared to other approaches, especially against highly variable background. They believe such detection, with limited false negatives, will form stepping stone towards future detection of damages in utility poles [6].

2.1.2 Graph Cut Approach

A precise detection is essential to inspect the defects of a power pole. In the paper, they propose a novel approach to detect the power pole object from images. Graph cut for image segmentation is a newly developing graph based image segmentation technique. It is effective but takes huge computation burden. The proposed approach combines prior knowledge with graph cut into detection. Firstly, it locates the rough region of the power pole to obtain two restricted regions based on prior rules. Then, a traditional graph cut framework is used in the restricted regions to improve the precision of segmentation. Experimental results verify its efficiency and accuracy [7].

2.1.3 LIDAR Databased approach

The paper presents a novel approach for detection of pole-like objects from LIDAR data. The designed method uses directional vector to detect pole-like structures in unordered point clouds. A new segmentation algorithm is presented as well. The segmentation is designed to overcome a common problem found in LIDAR data of urban environments, where a lot of poles are connected together with various types of wires. The method is tested on real world point clouds captured during mobile mapping process [8].

2.1.4 Urban point clouds approach

This approach focuses on detecting and classifying pole-like objects from point clouds obtained in urban areas. To achieve the goal, they propose a system consisting of three stages: localization, segmentation and classification. The localization algorithm based on slicing, clustering, pole seed generation and bucket augmentation takes advantage of the unique characteristics of pole-like objects and avoids heavy computation on the feature of every point in traditional methods. Then, the bucket-shaped neighbourhood of the segments is integrated and trimmed with region growing algorithms, reducing the noises within candidate's neighbourhood. Finally, we introduce a representation of six attributes based on the height and five-point classes closely related to the pole categories and apply SVM to classify the candidate objects into 4 categories, including 3 pole categories light, utility pole and sign, and the non-pole category. The performance of our method is demonstrated through comparison with previous works on a large-scale urban dataset [9].

2.2 Existing Work On Insulator

Transmission and distribution system leak overhead line of the by chance current cannot flow, so that the line from the Earth, the Insulator is used for the line. Insulator plays an important role in system operation. Transmission line insulators are devices used to contain, separate or support electrical conductors on high voltage electricity supply networks. Transmission insulators come in various shapes and types, including individual or strings of disks, line posts or long rods. They are made of polymers, glass and porcelain--each with different densities, tensile strengths and performing properties in adverse conditions.

There are many approaches to detect power line insulator like- Local Features and Spatial Orders approach [12], local gradient-based descriptors approach [13], color image based approach [14], visual attention mechanism [15], Profile projection approach [16].

2.2.1 Local Features and Spatial Orders approach

The detection of targets with complex backgrounds in aerial images is a challenging task. In their work, they propose a robust insulator detection algorithm based on local features and spatial orders for aerial images. First, they detect local features and introduce a multiscale and multi featured descriptor to represent the local features. Then, they get several spatial orders features by training these local features, it improves the robustness of the algorithm. Finally, through a coarse-to-fine matching strategy, they eliminate background noise and determine the region of insulators. they test their method on a diverse aerial image set. The experimental results demonstrate the precision and robustness of our detection method, and indicate the possible use of their method in practical applications [12].

2.2.2 Local Gradient-Based Descriptors Approach

This approach is based on discriminative training of local gradient-based descriptors and a subsequent voting scheme for localization. Further, they introduce an automatic extraction of the individual insulator caps and check them for faults by using a descriptor with elliptical spatial support. They demonstrate their approach on an evaluation set of 400 real-world insulator images captured from a helicopter and evaluate our results with respect to a manually created ground-truth. The performance of their insulator detector is comparable to other state-of-the-art object detectors [13].

2.2.3 Color Image Based Approach

Helicopter patrol inspection system has been applied to diagnose the insulator injuries of overhead transmission lines. This method is proposed to improve the diagnosis efficiency. Firstly, the statistical information of blocked image, the form of connected domain and the characteristics of edge chain code are utilized to recognize the area where the defective glass insulator locates; then by use of sliding window histogram statistic and histogram matching judgment, the damaged region of glass insulator is recognized. The proposed method is suitable to real-time detection in field environment, and it can diagnose the injury of insulator under a certain light variation range and background complexity [14].

2.2.4 Visual Attention Mechanism

This detection method based on visual attention mechanism. Firstly, the local and global saliency for the aerial images were calculated. Secondly, the local and global saliency maps
were combined together to get the final saliency map. Thirdly, the saliency region was extracted from the saliency map and the non-insulator region was excluded by a pro-processing method. Lastly, the saliency map was converted into binary map and the insulator was separated from the background by adding the binary map to the original image. Experimental results showed that

the method could extract the insulator from the aerial image precisely, and it was suitable for many practical applications such as insulator fault diagnosis, insulator contamination grade determination and so on [15].

2.2.5 Profile Projection Approach

Unlike previous texture-based approach, in this approach they directly search insulators location in the images by using Profile projection. For overcoming the negative effect of image noise on object detection, they pre-process insulators image by thresholding method. To make insulators detection more effective and efficient, they design a tilt correction method based on principal component analysis. The correction enables their method to derive accurate feature extraction curve from insulators image, then they extract five features from the feature curve, which are related to the number of binary sequence and the normalized variance of the binary sequence length. After obtaining the insulators feature from an image, they apply SVM to identify insulators with the five features [16].

2.3 Rigid kernel based detection

In this section, we discuss detection algorithms that are based on learning a set of rigid kernels. In this class of algorithms, learning rigid templates and boosted cascades of classifiers are used. The Viola-Jones detection algorithm is the prime example, which also motivated many researchers in detection. Another line of research on rigid templates, which currently gains momentum, is based on Deep Convolutional Neural Networks (DCNNs).

2.3.1 The Viola-Jones Face Detector

The seminal work by Viola and Jones [17] had the most impact in the 2000s.The Viola-Jones detection method has three ideas that made it fast: the integral image, classifier learning with AdaBoost, and the attentional cascade. The first major development was achieved by the seminal work of Viola and Jones [17] on boosting based detection, which was the first algorithm that made face detection practically feasible in real-world applications. Today it is widely applied in photo organization software. Research in face detection has progressed significantly in the direction of providing algorithms that are able to detect faces as well as other objects like insulator. Modern detection algorithms have benefited from various methods, algorithms and features. structure.

2.3.1.1 The Integral Image

The integral image (summed area table) algorithm, computes quickly and efficiently the sum of values in a rectangle subset of a grid. It was introduced to the computer graphics field in [18].

2.3.1.2 AdaBoost Learning:

Boosting method finds a highly accurate hypothesis by combining many weak hypotheses, each with moderate accuracy. The AdaBoost (Adaptive Boosting) algorithm is generally considered as the first step towards more practical boosting algorithms [19,20].

2.3.1.3 The Cascade Structure:

The cascade is an important component of Viola-Jones detector. This boosted classifier can be built that reject most of the negative sub-windows while keeping almost all the positive examples. Majority of the sub-windows will be rejected in early stages of the detector, making the detection process extremely efficient. The process of classifying a sub-window forms a degenerate decision tree, which is called a cascade. It was presented in [17]. The cascade structure also has an impact on the training process. There are billions of negative examples needed to train a high-performance face detector. To handle the huge amount of negative training examples, the Viola-Jones face detector [17] used a bootstrap process. At each node, a threshold was manually chosen. A partial classifier was used to scan the negative example set to find more unelected negative examples for the training of the next node. Each node is trained independently, as if the previous nodes does not exist. One argument behind such a process is to force the addition of some nonlinearity in the training process, which could improve the overall performance. However, recent works showed that it is actually beneficial not to completely separate the training process [16] of different nodes, the cascade is constructed manually. That is, the number of weak classifiers and the decision threshold for early rejection at each node are both manually specified. This is a non-trivial task. If the decision thresholds were set too aggressively, the final detector will be very fast, but the overall detection rate may be affected. On the other hand, if the decision thresholds are set very conservatively, most sub-windows will need to pass through many nodes, making the detector very slow. Combined with the limited computational resources available in the early 2000s, it is no wonder that training a good face detector can take months of fine-tuning [21].

2.4 Insulator classification:

Classification of remotely sensed data is used to assign corresponding levels with respect to groups with homogeneous characteristics, with the aim of discriminating multiple objects from each other within the image. The level is called class. Classification will be executed on the base of spectral or spectrally defined features, such as density, texture etc. in the feature space. It can be said that classification divides the feature space into several classes based on a decision rule [22].

In many cases, classification will be undertaken using a computer, with the use of mathematical classification techniques. Various classification techniques will be compared with the training data, so that an appropriate decision rule is selected for subsequent classification. Depending up on the decision rule, all pixels are classified in a single class. There are two methods of pixel by pixel classification and per-field classification, with respect to segmented areas.

Popular techniques are as follows.

- Multi-level slice classifier
- Minimum distance classifier
- Maximum likelihood classifier
- Other classifiers such as fuzzy set theory and expert systems

There are several types of classification- Categorical (Nominal): Classification of entities into particular categories, Ordinal: Classification of entities in some kind of ordered relationship, Adjectival or Predicative: Classification based on some quality of an entity. And Cardinal: Classification based on a numerical value. [23]

Classifiers are different algorithms or data structure to perform classification. The most commonly use classifiers are support vector machine, Neural Network, Naïve Bayes classifier, Decision tree, K-nearest Neighbours etc. Now-a days, Deep learning is most popular for classification. In our work, we have used support vector machine to train and detect insulator.

2.4.1 SVM (Support Vector Machine)

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labelled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. [24]

Support Vector Machines were first introduced by Vapnik and Chervonenkis in [25]. The core idea is to find the optimal hyperplane to separate a dataset, while there are theoretically infinite hyperplanes to separate the dataset. A hyperplane is chosen, so that the distance to the nearest data point of both classes is maximized. The points spanning the hyperplane are the Support Vectors, hence the name Support Vector Machines [26,27].

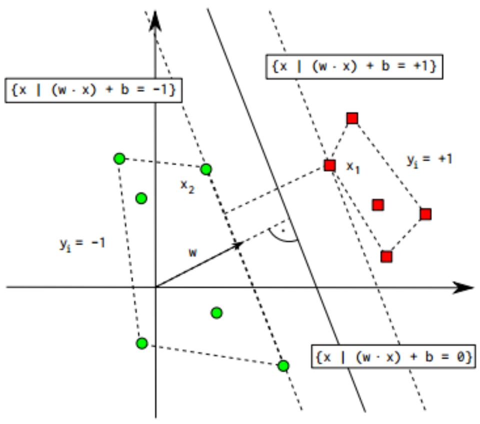


Figure 2.1: Maximum Margin Classifier

Due to their superior performance in general machine learning problems, they have also become a very popular approach for detection [28,29]. However, the detection speed of SVM based face detectors was generally slow. Thus, various schemes have been proposed to speed up the process. For instance, in [30] a method that computes a reduced set of vectors from the original support vectors was proposed. These reduced set vectors are then tested against the test example sequentially, making early rejections possible. In [31], detection speed was further improved by approximating the reduced set vectors with rectangle groups, gaining another 6-fold speedup. A hierarchy of SVM classifiers with different resolutions in order to speed up the overall system was applied. The early classifiers were at low resolution, say, 3×3 and 5×5 pixels, which can be computed very efficiently to prune negative examples. Multiview detection has also been explored with SVM based classifiers.

A cascade of SVMs were first trained through bootstrapping. The remaining positive and negative examples were then randomly partitioned to train a set of SVMs, whose outputs were then combined through majority voting. In [33] a single SVM for Multiview detection was used, and relied on the combination of local and global kernels for better performance. No experimental results were given in [32, 33] to compare the proposed methods with existing schemes on standard data sets, hence it is unclear whether these latest SVM based detectors can outperform those learned through boosting.

Support Vectors: Input vectors that just touch the boundary of the margin (street) – circled below, there are 3 of them (or, rather, the 'tips' of the vectors. Here, we are the actual support vectors, v1, v2, v3, instead of just the 3 circled points at the tail ends of the support vectors. D denotes 1/2 of the street 'width. [34]

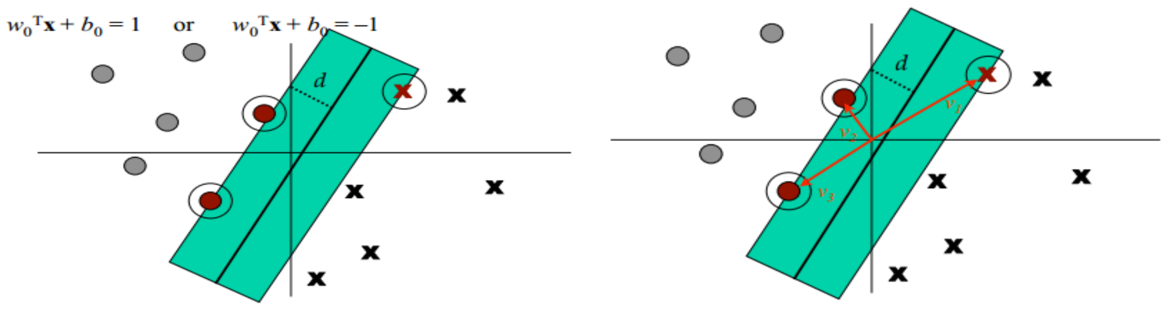


Figure 2.2: Support Vectors

2.4.1.1 Definition

Define a hyperplane H such that:

$W*x^i + b \geq +1$ when $y_i=+1$

$W*x^i + b \leq -1$ when $y_i=-1$

H_1 and H_2 are the planes:

H_1: $W*x^i + b \geq +1$ when $y_i=+1$

H2: $W*x^i + b \leq -1$ when $y_i= -1$

The points on the planes *H1* and *H2* are the tips of the Support Vectors [35].

The plane *H0* is the median in between, where *w*•*x*i+*b* =0

d+ = the shortest distance to the closest positive point

d- = the shortest distance to the closest negative point

The margin (gutter) of a separating hyperplane is d+ + d–

The optimization algorithm to generate the weights proceeds in such a way that only the support vectors determine the weights and thus the boundary [34],

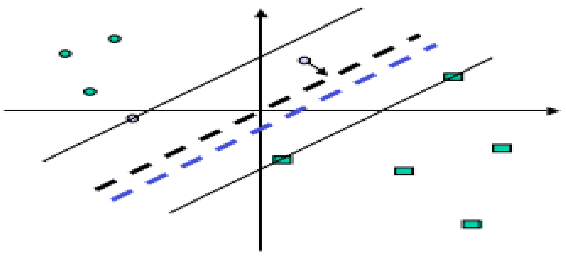


Figure 2.3: Support Vector Moves to The Decision Boundary

Form of equation defining the decision surface separating the classes is a hyperplane of the form: $w^T x + b = 0$ where

– w is a weight vector

– x is input vector

– b is bias

We can write

- $w^T x + b > 0$ for di = +1
- $w^T x + b < 0$ for di = –1

We want a classifier (linear separator) with as big a margin as possible. Recall the distance from a point (x0, y0) to a line: Ax+By+c = 0 is: |Ax0 +By0 +c|/sqrt (A2+B2), so, the distance between H 0 and H1 is then:

|w•x+b|/||w||=1/||w||, so the total distance between H1 and H2 is thus: 2/||w|| [34].

In order to maximize the margin, we thus need to minimize ||w||. With the condition that there are no data points between H1 and H2:

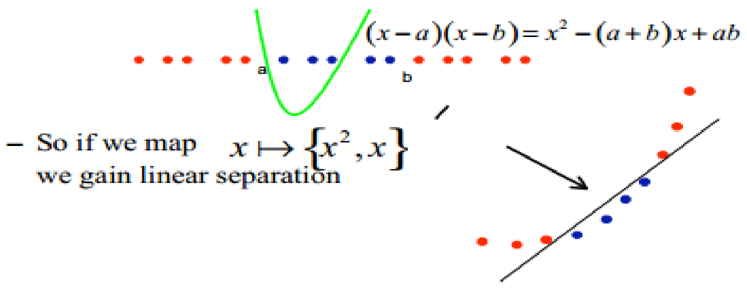
$\mathbf{x}_i \bullet \mathbf{w} + b \geq +1$ when $y_i = +1$…. (i)

$\mathbf{x}_i \bullet \mathbf{w} + b \geq -1$ when $y_i = -1$ …. (ii)

(i) & (i) Can be combined into: Vi(xi•w) ≥ 1 [34]

2.4.1.2 Non-linear SVM

The idea is to gain linearly separation by mapping the data to a higher dimensional space. The following set can't be separated by a linear function, but can be separated by a quadratic one.



$(x-a)(x-b) = x^2 - (a+b)x + ab$
a
b
– So if we map $x \mapsto \{x^2, x\}$
we gain linear separation

Figure 2.4: Non-linear Data separation

The kernel trick is used for classifying non-linear datasets. It works by transforming data points into a higher dimensional feature space with a kernel function, where the dataset can be separated again. The linear classifier relies on inner product between vectors $K(x_i,x_j) = x_i^T x_j$. If every data point is mapped into high dimensional space via some transformation Φ: $x \rightarrow \varphi(x)$, the inner product becomes: $K(xi,xj) = \varphi(xi)^T \varphi(xj)$. kernel function corresponds to an inner product into some feature space [35].

Some Commonly used kernel functions:

- Linear: $K(\mathbf{x_i},\mathbf{x_j}) = \mathbf{x_i}^T\mathbf{x_j}$
- Polynomial of power p: $K(\mathbf{x_i},\mathbf{x_j}) = (1 + \mathbf{x_i}^T\mathbf{x_j})^p$
- Gaussian (radial-basis function network): $K(\mathbf{x_i},\mathbf{x_j}) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$

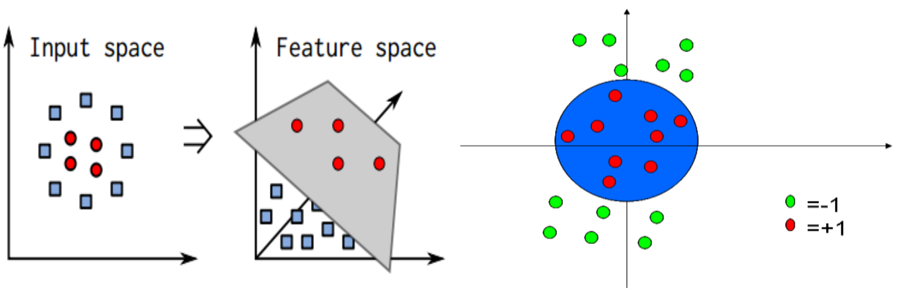


Figure 2.5: kernel trick

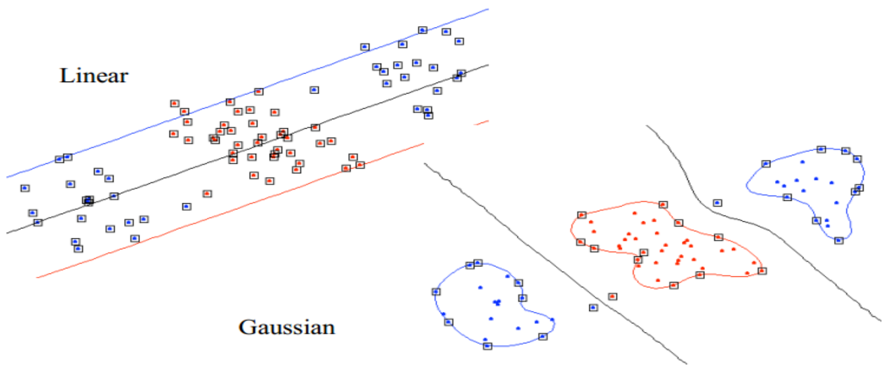


Figure 2.6: Non Linear SVMs 2 –Gaussian Kernel

2.4.1.3 Overfitting by SVM

A well-known problem with machine learning methods is overtraining. This means that we have learned the training data very well, but we cannot classify unseen examples correctly. Every point is a support vector too much freedom to bend to fit the training data – no generalization. In fact, SVMs have an 'automatic' way to avoid such issues. [Vapnik, 1995.] We add a penalty function for mistakes made after training by over-fitting: recall that if one over-fits, then one will tend to make errors on new data. This penalty function can be put into the quadratic programming problem directly [34].

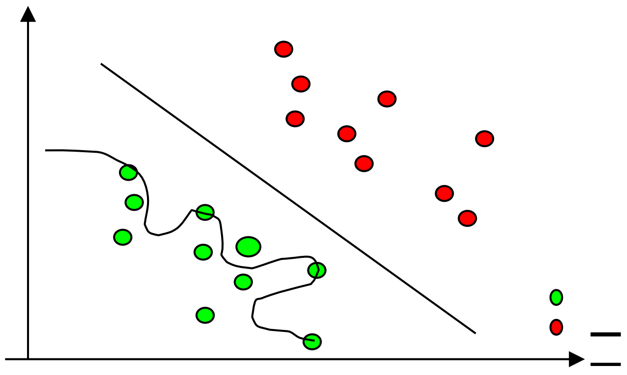


Figure 2.7: Overfitting Problem

2.4.2 LIBSVM:

LIBSVM is a simple, easy-to-use, and efficient software for SVM classification and regression [36]. It solves C-SVM classification, nu-SVM classification, one-class-SVM, epsilon-SVM regression, and nu-SVM regression. It also provides an automatic model selection tool for C-SVM classification. In LIBSVM training vectors **x**i are mapped into a higher (maybe infinite) dimensional space by the function φ. SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. C > 0 is the penalty parameter of the error term. Furthermore, K(**x**i; **x**j) ≡ φ(**x**i)Tφ(**x**j) is called the kernel function. Though new kernels are being proposed by researchers [36].

Following procedures are done by LIBSVM: Transform data to the format of an SVM package, conduct simple scaling on the data, Consider the RBF kernel, Use cross-validation to find the best parameter C and γ, Use the best parameter C and γ to train the whole training set5.LIBSVM includes the following methods [36]:
svm_type:

- C_SVC n-class classification ($n \geq 2$), allows imperfect separation of classes with penalty multiplier C for outliers.
- NU_SVC n-class classification with possible imperfect separation. Parameter nu (in the range 0 … 1, the larger the value, the smoother the decision boundary) is used instead of C.
- ONE_CLASS one-class SVM. All the training data are from the same class, SVM builds a boundary that separates the class from the rest of the feature space.
- EPS_SVR regression. The distance between feature vectors from the training set and the fitting hyper-plane must be less than p. For outliers, the penalty multiplier C is used.
- NU_SVR regression; nu is used instead of p.

LIBSVM uses a decomposition method for classification [37]. It used in a simple working set selected which lead to a faster convergence for different and difficult cases. Parameters in LIBSVM are

- svm_type: C_SVC, NU_SVC, ONE-CLASS, EPS_SVR, NU_SVR

- kernel_type: linear: d(x; y) = x · y == (x; y) ,polynomial: d(x; y) = (gamma ∗ (x · y) + coef0)^{degree} ,RBF: d(x; y) = exp(-gamma∗ |x - yj2),sigmoid: d(x; y) = tanh(gamma∗(x·y)+coef0),precomputed kernel
- C, nu, p Parameters in the generalized SVM optimization problem.
- lass_weights Optional weights, assigned to particular classes. They are multiplied by C and thus affect the misclassification penalty for different classes. The larger weight, the larger penalty on misclassification of data from the corresponding class.
- term_criteria Termination procedure for iterative SVM training procedure (which solves a partial case of constrained quadratic optimization problem)

    - type is either CV_TERMCRIT_ITER or CV_TERMCRIT_ITER
    - max_iter is the maximum number of iterations in training.
    - epsilon is the error to stop training.

2.4.2.1 Cross-validation and Grid-search

There are two parameters for an RBF kernel: C and γ. It is not known beforehand which C and γ are best for a given problem; consequently, some kind of model selection (parameter search) must be done. The goal is to identify good (C; γ) so that the classifier can accurately predict unknown data (i.e. testing data). Note that it may not be useful to achieve high training accuracy (i.e. a classifier which accurately predicts training data whose class labels are indeed known).

A common strategy is to separate the data set into two parts, of which one is considered unknown. The prediction accuracy obtained from the unknown" set more precisely reflects the performance on classifying an independent data set. An improved version of this procedure is known as cross-validation. In v-fold cross-validation, we first divide the training set into v subsets of equal size. Sequentially one subset is tested using the classifier trained on the remaining v - 1 subsets. Thus, each instance of the whole training set is predicted once so the cross-validation accuracy is the percentage of data which are correctly classified [36].

The cross-validation procedure can prevent the overfitting problem. We use a grid-search" on C and γ using cross-validation. Various pairs of (C; γ) values are tried and the one with the best cross-validation accuracy is picked. We found that trying exponentially growing sequences of C and γ is a practical method to identify good parameters (for example, $C = 2^{-5}; 2^{-3} \ldots 2^{15}$, $\gamma = 2^{-15}; 2^{-13} \ldots 2^{3}$). The grid-search is straightforward but seems naive. In fact, there are several advanced methods which can save computational cost by, for example, approximating the cross-validation rate. After "grid-search" the best (C; γ) is found, the whole training set is trained again to generate the final classifier [36].

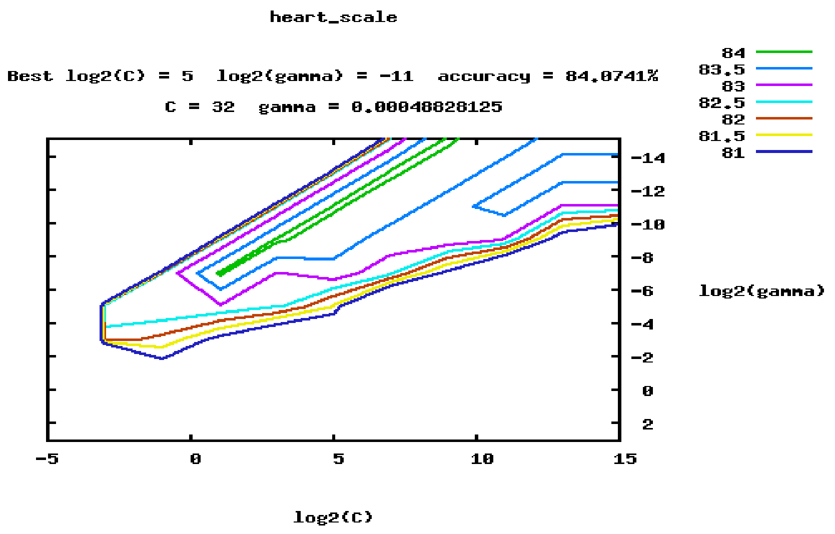


Figure 2.8: grid search in LIBSVM (image curtesy: Chih-Chung Chang and Chih-Jen Lin)

2.4.3 Sliding window

Sliding windows play an integral role in object classification, as they allow us to localize exactly "where" in an image an object resides. Utilizing both a sliding window and an image pyramid we are able to detect objects in images at various scales and locations [38]. The approach involves scanning the image with a fixed-size rectangular window and applying a classifier to the sub-image defined by the window. Successively scaled copies of the image we apply sliding window technique for identifying and localizing objects. A sliding window is rectangular region of fixed width and height that "slides" across an image.

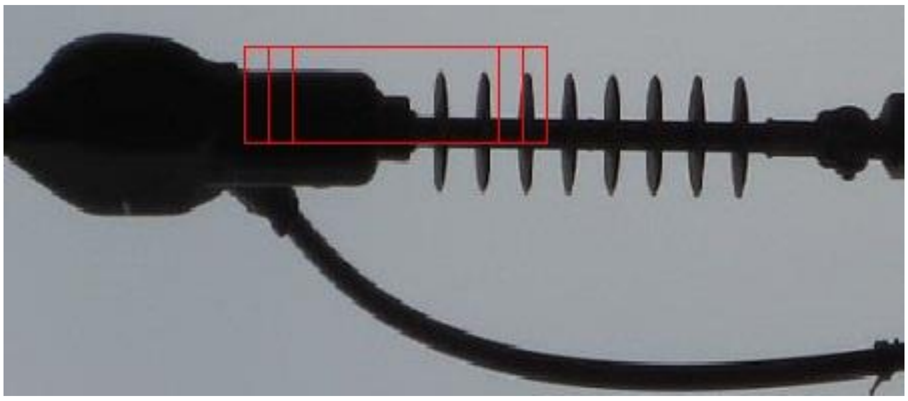


Figure 2.9: Example of the sliding window approach

For each of these windows, we normally take the window region and apply HOG to determine if the window has an object that interests us. Combined with image pyramids we can create image classifiers that can recognize objects at varying scales and locations in the image**.** These techniques, while simple, play an absolutely critical role in object detection and image classification [38].

Usually non-maximal neighbourhood suppression is applied to the output to remove multiple detections of the same object after apply sliding approach classification. We apply this technique not only on scaled based image but also rotation with pyramid image. We rotate image then we apply sliding technique with that fixed window for every scaled copy and continue rotating until we rotate the image in 360-degree. Hence, sliding window technique posses a great contribution for identifying and localizing objects in an image.

2.4.4 Pyramid Image

An image pyramid is a collection of images - all arising from a single original image - that are successively down sampled until some desired stopping point is reached. [39]. An “image pyramid” is a multi-scale representation of an image. Utilizing an image pyramid allows us to find objects in images at different **scales** of an image. And when combined with a sliding window we can find objects in images in various locations. At the bottom of the pyramid we have the original image at its original size (in terms of width and height). And at each subsequent layer, the image is resized (subsampled). [40]

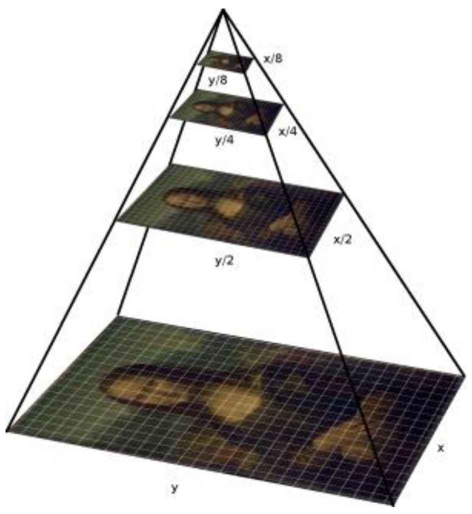


Figure 2.10: Example of Pyramid image. At each layer image is downsized

(image curtesy: Adrian Rosebrock page)

The image is progressively subsampled until some stopping criterion is met, which is normally minimum size has been reached and no further subsampling needs to take place.

There are two common kinds of image pyramids:

- **Gaussian pyramid:** Used to downsample images
- **Laplacian pyramid:** Used to reconstruct an upsampled image from an image lower in the pyramid (with less resolution) [39]

In our project to construct image pyramids we utilize C++ with OpenCV. As a convention, we took a zoom picture so **we** Perform downsampling. First, we reduce our image **by 25%** every time**.** Using a scale factor of 1.3333, 2, and 4, only 4 layers have been generated**.**

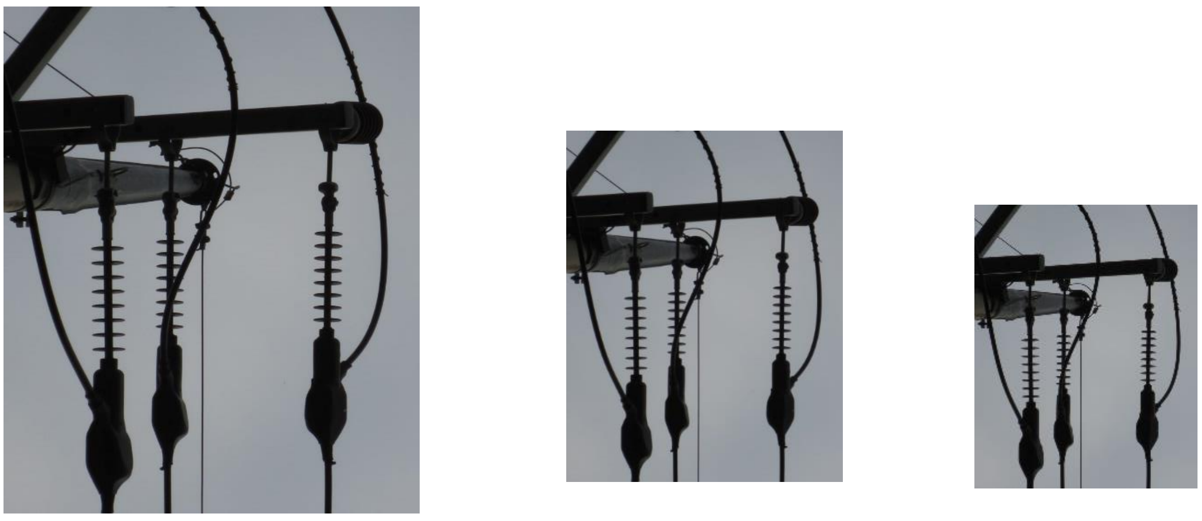


Figure 2.11: Multi Layer images

Hence, we are using the HOG descriptor for object classification so we do not use smoothing since smoothing tends to hurt classification performance.
In general, there is a trade-off between performance and the number of layers that we generate. The smaller the scale factor is, the more layers need to create and process — but this also gives an image classifier a better chance at localizing the object we want to detect in the image.
A larger scale factor will yield less layers, and perhaps might hurt our object classification performance; however, we have obtained much higher performance gains since we will have less layers to process.

Chapter 3

Proposed System Discussion

In this chapter we describe our proposed system to detect pole and insulator. First, we describe pole detection in section 3.1. In section 2 we describe insulator detection process.

3.1 Pole Detection

For carrying of overhead line, wooden poles**,** concrete poles**,** steel poles **and** rail poles are used. Which poles are to be used, depend on the importance of load, location and place, cost effect of such construction, including maintenance cost, and keeping its profit element in mind. In low voltage line for all phases, natural and earth single pole line is used. [41]

We have found insulator on the top of the pole so it was important for us to detect electrical pole first to find to insulator.

3.1.1 Methodologies

A electrical pole is a column or post used to support overhead power lines and various other public utilities, such as electrical cable, fibre optic cable, and related equipment such as transformers and insulator. Electrical wires, cables and insulators are routed overhead on utility poles as an inexpensive way to keep them insulated from the ground and out of the way of people and vehicles. [wiki]. As we found insulator on the top of the electrical pole so before detecting insulator first we had to detect pole. First we took a close picture in a certain area where an electrical pole can be founded. Then we try to extracted lines from the image. To do that we applied edge detection algorithm to find out the edge of the image. After that, we applied Hough line transform to extract lines. Next. we discard all the horizontal lines. As pole is a vertical object so we took lines only which angle, θ is 90^0 and -90^0. In real life, we see that pole is not always vertical, we see sometimes it skewed so took a variation in angle of $\pm5^0$ ($90^0 > \theta >= 85$ and $-85 => \theta > -90^0$) as an empirical value. After extracted the lines, we grouped the lines in a way so that their end point stays closer. After finished the grouping we calculated the coverage area for every group by using heuristic function. We took three best group which were consists of lines. In this three best lines we had our pole.

3.1.1 Edge detection

In an image, an edge is a curve that follows a path of rapid change in image intensity. Edges are often associated with the boundaries of objects in a scene. Edge detection is used to identify the edges in an image. To find edges, we can use the edge function. This function looks for places in the image where the intensity changes rapidly, using one of these two criteria: Places where the first derivative of the intensity is larger in magnitude than some threshold and Places where the second derivative of the intensity has a zero crossing [42].

We use canny in our project to find to the edges. Canny is the most powerful edge detection method. It is different from other edge detection methods that detects strong and weak edges.

The Canny edge detection algorithm can be broken down into 5 steps:

- Smooth the image using a Gaussian filter to remove high frequency noise.
- Compute the gradient intensity representations of the image
- Apply non-maximum suppression to remove "false" responses to edge detection
- Apply thresholding using a lower and upper boundary on the gradient values
- Track edges using hysteresis by suppressing weak edges that are not connected to strong edges. [43]

OpenCV puts all the above in single function and the function signature looks like this:

C++: Canny (InputArray image, OutputArray edges, double threshold1, double threshold2, int apertureSize=3, bool L2gradient=false)

C: void cvCanny(const CvArr* image, CvArr* edges, double threshold1, double threshold2, int aperture_size=3)

We use the second one to find out the edges. First argument is our input image. Second is the output edge map; it has the same size and type as input image. Third and fourth arguments are our minVal and maxVal respectively for the hysteresis procedure. Last, argument is aperture size. It is the size of Sobel kernel used for find image gradients. By default, it is 3 [44].

First, we convert the image into gray scale image and then we apply canny function.

Canny function in OpenCV we have used:

```
Canny (src, dst, 50, 200, 3);
```

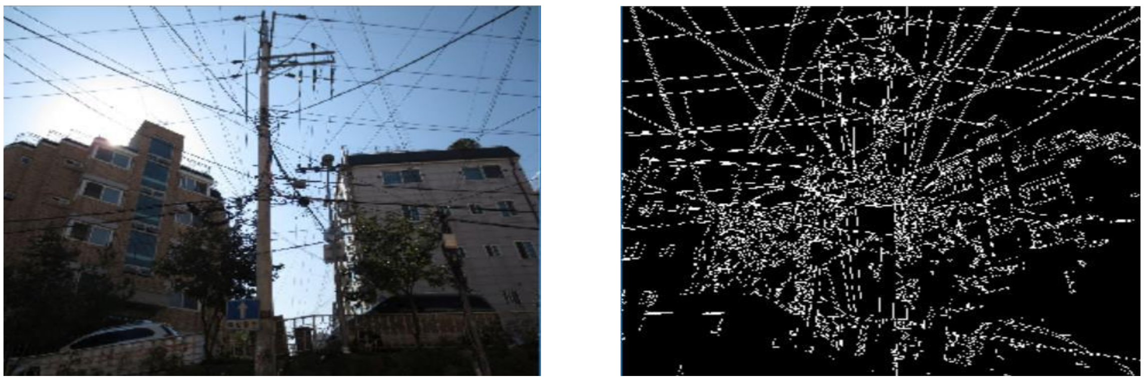

(a) (b)

Figure 3.1: (a) Original Image (b) Detected Edge after using canny

3.1.3 Line detection:

Hough lines transform The Hough transform is a feature extraction technique. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. It means that in general, a line can be detected by finding the number of intersections between curves [45].

We will see how Hough transform works for line detection using the HoughLine transform method. To apply the Transform, first an edge detection pre-processing is desirable.

Hough transform is based on the fact that a line in the x-y cartesian coordinate system can be mapped onto a point in the rho-theta space. [46]

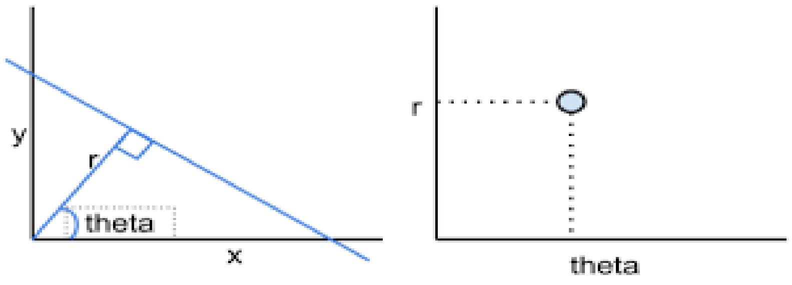


Figure 3.2: Polar system

Hough Transforms, we will express lines in the Polar system. Hence, a line equation can be written as:

$$y = -\frac{cos\theta}{sin\theta}x + \frac{r}{sin\theta}$$

Arranging the terms: r = x sin θ +y sin θ when we see a point on an image, and we are not sure whether or not it belongs to a line like structure in the image, we just go ahead and plot a point for all possible lines that can pass through that point. That would result in a sinusoidal curve in the rho-theta space. [46]

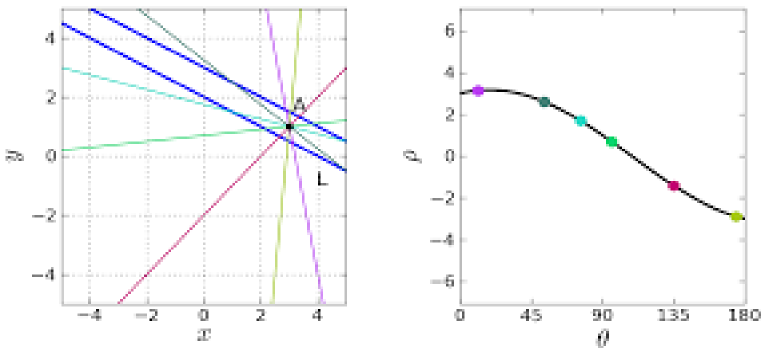


Figure 3.3: Sinusoidal curve in the rho-theta space

If the point actually does belong to a line, the actual rho-theta coordinate in the rho-theta plane will be reinforced by all points that belong to the line.

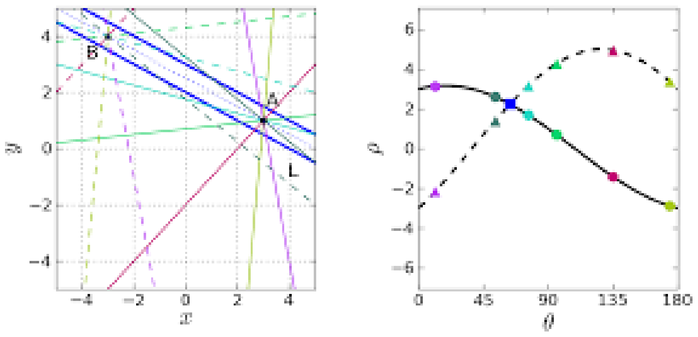


Figure 3.4: Intensity curve of the reinforcement strength

If we plot an intensity curve of the reinforcement strength (number of curves that cross a point in the rho-theta space), we can see peaks at the values that correspond to possible lines. These points can be isolated and picked up by applying a threshold value. [46]

OpenCV already has an implementation of the Hough transform:

a. **The Standard Hough Transform**
   - It consists in pretty much what we just explained in the previous section. It gives you as result a vector of couples (θ, r_θ)
   - In OpenCV it is implemented with the function HoughLines

b. **The Probabilistic Hough Line Transform**

- A more efficient implementation of the Hough Line Transform. It gives as output the extremes of the detected lines (x_0, y_0, x_1, y_1)
- In OpenCV it is implemented with the function HoughLinesP [47]

3.1.3.1 Probabilistic Hough Transform

In the Hough transform, you can see that even for a line with two arguments, it takes a lot of computation. Probabilistic Hough Transform is an optimization of Hough Transform we saw. It doesn't take all the points into consideration, instead take only a random subset of points and that is sufficient for line detection. Just we have to decrease the threshold. See below image which compare Hough Transform and Probabilistic Hough Transform in Hough space. [48]

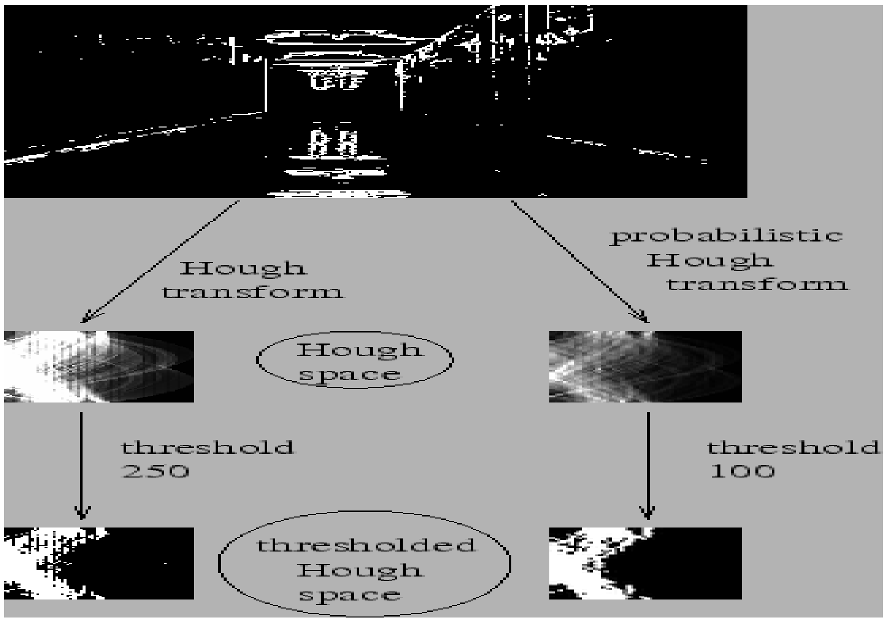


Figure 3.5: Compare Hough Transform and Probabilistic Hough Transform in Hough space (Image Courtesy: Franck Bettinger's home page)

OpenCV implementation is based on Robust Detection of Lines Using the Progressive Probabilistic Hough Transform by Matas, J. and Galambos, C. and Kittler, J.V. The function used is

C++: void HoughLinesP(InputArray **image**,OutputArray **lines**,double **rho**, double **theta**, int **threshold**, double **minLineLength**=0, double**maxLineGap**=0)

Parameters:

- **image** – 8-bit, single-channel binary source image. The image may be modified by the function
- lines – Output vector of lines. Each line is represented by a 4-element vector (x_1, y_1, x_2, y_2), where (x_1, y_1) and (x_2, y_2) are the ending points of each detected line segment.
- rho: The resolution of the parameter r in pixels. We use 1 pixel.
- theta: The resolution of the parameter θ in radians. We use 1 degree (CV_PI/180)
- threshold: The minimum number of intersections to "detect" a line
- minLinLength: The minimum number of points that can form a line. Lines with less than this number of points are disregarded.
- maxLineGap: The maximum gap between two points to be considered in the same line [49]

3.1.4 Grouping lines:

After applied probabilistic Hough transform we got a vector of $[x_0, y_0, x_1, y_1]$ with lots of line. But, all the lines were not needed for grouping. So, we discarded the horizontal lines first then we took the vertical lines. It was known that always pole was not found vertical sometimes it founded skewed. So, we granted angle 90^0 and -90^0 as well as we also allowed which lines angle is between $90^0 > \theta >= 85$ and $-85 => \theta > -90^0$. We allowed an empirical value of 5^0 in the angle. We use this formula to find out the angle:

Angle, $\theta = \tan^{-1}(\frac{\Delta y}{\Delta x}) = \tan^{-1}(m)$ Where, $m = \frac{\Delta y}{\Delta x}$ is the slope of the line.

For grouping lines, first we had to check if the lines were in correct order or not. If not, then we pre-processed the lines. First, we checked first end point every line's two and rearranged the values according every line's first end point.

Next. we arbitrarily choose a line from all the lines as a first line in one group. we did it for every time for grouping. For the first group we selected a line as group's first line. From the first line's first end point we calculated the distance of all line's first end point and store them as well as find the minimum distance. The minimum distance line in respect to the first line is our candidate for the next line. We took an empirical value of 5 pixels as a threshold1. We took the average of previously detected line's x-axis difference which is our threshold2. if our candidate line's and first line's x-axis difference is less or equal to threshold1 and threshold2 then went to the next step. Next step we checked that was our candidate line's first endpoint y-value is less than our first line's first endpoint. If all the requirement fulfilled, then we granted candidate line to enter into that group. And, candidate line was our nest first line whom we have found out the next line's in that particular group. In this way we continued the whole process until all the line's not grouped.

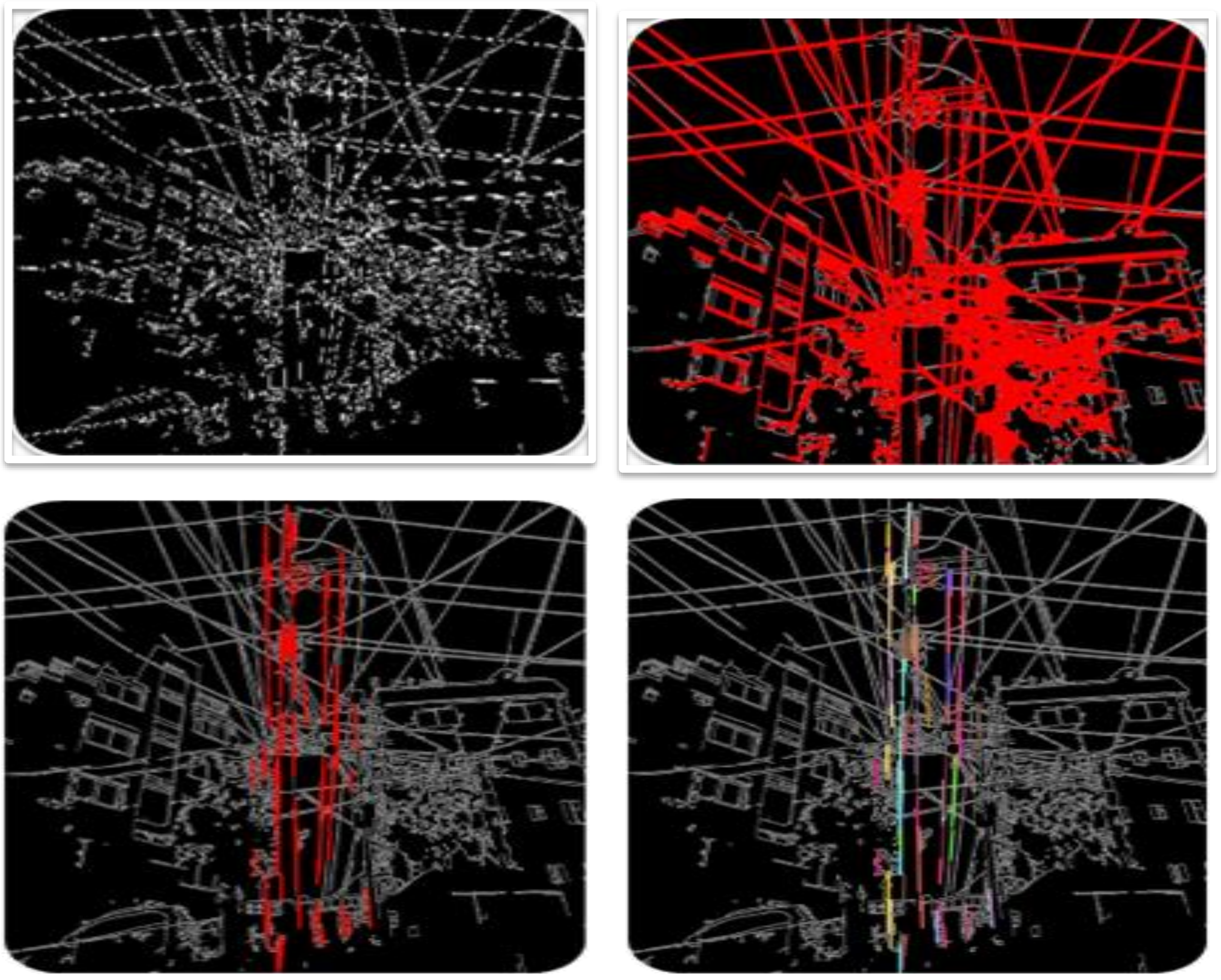


Figure 3.6: Left-Top: Canny edge detected image, Right-Top: Extracted lines, Left-Bottom: vertical lines with $\pm 5^{0,}$ Right-Bottom: Grouped lines (shown in different colour)

3.1.5 Coverage area:

After finished the grouping, we had the grouped data for finding out how much a line cover in the image. By finding out the coverage we could find out which group of lines covered most of the area in the image. So, to find out the coverage we applied a heuristic. First, we calculated the length of that group which was consist lines. After that we calculated every single line's distance. And, finally we multiplied the length with the total distance. This is our heuristic. first, we had find out the length of a group. We took the first and last line of the group. And, measured the difference between first line's first end point y value and last line's last endpoint y value.

Hence, length, l = $(y_0 - y_1)$ [y_0= first line first end point's y-axis, y_1=last line last end point's y-axis]

We did that for all the group.After we calculated the total distance. In a group, there were more than one lines. First, we calculated the last and first end point's difference of a line and took the absolute value of it. We did it for all the lines in a group. After that we added them up all which gave the total distance.

Hence, total distance, $d = |d_1| + |d_2| + |d_3| + \ldots + |d_n|$

Coverage area= length * total distance

$= l * d$

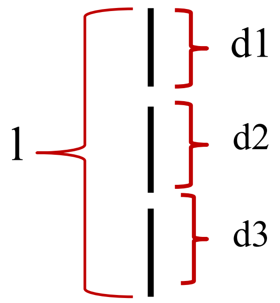


Figure 3.7: calculate length and distance

Now, we had the coverage area for all the group. We pick the best three coverage group. This three groups had the possibility to be a pole. The first one had a better possibility to be a pole if it misses then the next one had. If the first two missed, then we had to look for the third one was that a pole or not.

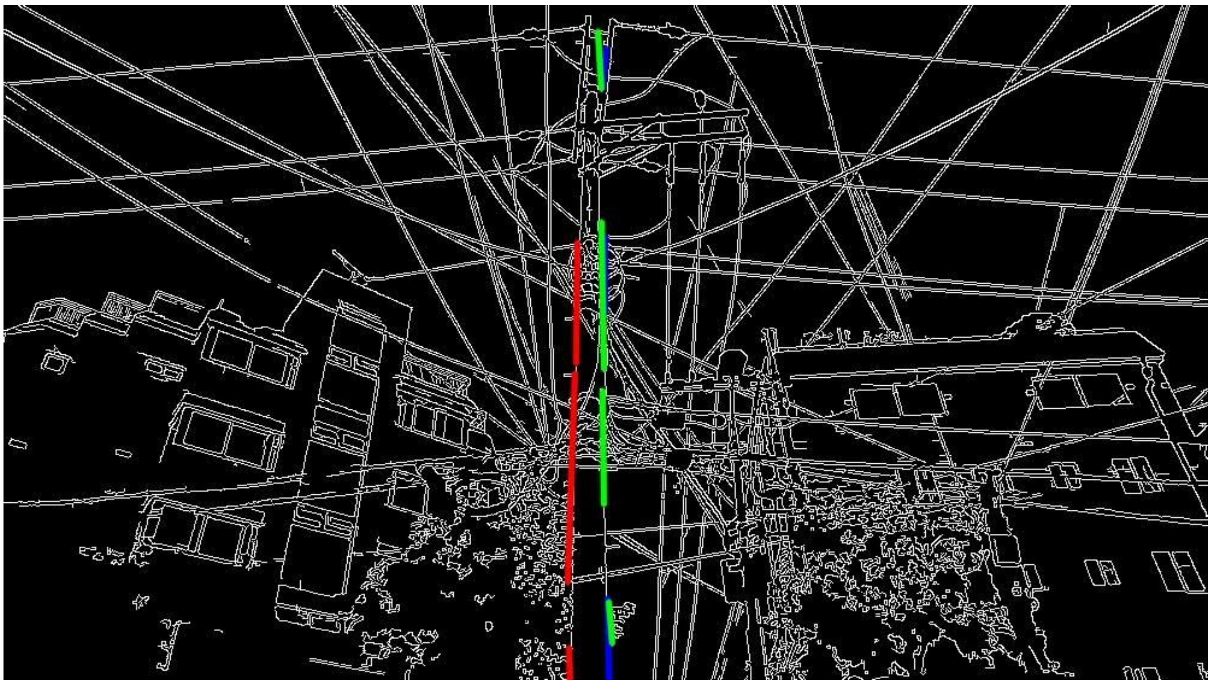


Figure 3.8: Final result (pole detect)

3.2 Insulator detection

Insulators are materials that have just the opposite effect on the flow of electrons. They do not let electrons flow very easily from one atom to another. Insulators are materials whose atoms have tightly bound electrons. These electrons are not free to roam around and be shared by neighboring atoms It is a device that used to contain, separate or support electrical conductors on high voltage electricity supply networks. It protects us from the dangerous effects of electricity flowing through conductors. [50] So, insulator detection is important to find out its faultiness and reduce time consumption and risk as well as increase diagnosis efficiency.

3.2.1 Methodologies

Insulator detection poses a vital role in inspection of faulty insulators which is the most common problems in power line transmission networks. It is a computer vision based method that will detect power line insulator which automatically analyses the images. To detect insulator perceiving insulator features are very important. We take lots of positive and negative image in a uniform manner to extract features from them. We used Microsoft visual studio using openCV to extract the features from those datasets. By using this feature, we trained the computer in LIBSVM tool to detect insulator. After training, we used Sliding-window object detection technique for identifying and localizing insulators in an image. It scans image with a fixed-size rectangular window. Then, extracting features of that window(sub-image) and applying SVM to classify that the sub-window bounds an insulator. The process is repeated on successively scaled copies of the image so that insulators can be detected at any size.We rotate the image with 10^0 increments from 0^0 to 360^0 as well as we apply sliding window technique and repeat the process on successively scaled copies to detect insulators at any size. By using negative mining, we reduce the detection of false positive rate. After that non-maximal neighborhood suppression is applied to the output to remove multiple detections of the same insulator.

3.2.2 Histogram oriented gradient

Detection in images is a challenging task owing to their variable appearance and the wide range of poses that they can adopt. The first need is a robust feature set that allows the insulator form to be discriminated cleanly, even in cluttered backgrounds under difficult illumination. The issue of feature sets for human detection, showing that locally normalized Histogram of Oriented Gradient (HOG) descriptors provide excellent performance relative to other existing feature sets [51].

3.2.2.1 Feature descriptor

A feature descriptor is a representation of an image or an image patch that simplifies the image by extracting useful information and throwing away extraneous information. Typically, a feature descriptor converts an image of size width x height x 3 (channels) to a feature vector / array of length n. In the case of the HOG feature descriptor, the input image is of size 64 x 128 x 3 and the output feature vector is of length 3780 [51]. Hence, HOG descriptor can be calculated for other sizes. the feature vector is not useful for the purpose of viewing the image. But, it is very useful for tasks like image recognition and object detection. The feature vector produced by these algorithms when fed into an image classification algorithm like Support Vector Machine (SVM) produce good results. Good features extracted from an image should be able to tell the difference between one object and other objects. In the HOG feature descriptor, the distribution (histograms) of directions of gradients (oriented gradients) are used as features. Gradients (x and y derivatives) of an image are useful because the magnitude of gradients is large around edges and corners (regions of abrupt intensity changes) and we know that edges and corners pack in a lot more information about object shape than flat regions.

Compute a Histogram of Oriented Gradients (HOG) by [52]

- (optional) global image normalisation
- computing the gradient image in x and y
- computing gradient histograms
- normalising across blocks
- flattening into a feature vector

The first stage applies an optional global image normalisation equalisation that is designed to reduce the influence of illumination effects. In practice, we use gamma (power law) compression, either computing the square root or the log of each colour channel. Image texture strength is typically proportional to the local surface illumination so this compression helps to reduce the effects of local shadowing and illumination variations.

The second stage computes first order image gradients. These capture contour, silhouette and some texture information, while providing further resistance to illumination variations. The locally dominant colour channel is used, which provides colour invariance to a large extent. Variant methods may also include second order image derivatives, which act as primitive bar detectors - a useful feature for capturing, e.g. bar like structures in bicycles and limbs in humans.

This gradient can easily have achieved by filtering the image with the following kernels.

-1	0	1

-1
0
1

Figure 3.9: Horizontal and Vertical gradients filtering kernel

We can also achieve the same results, by using Sobel operator in OpenCV with kernel size 1.

Next, we can find the magnitude and direction of gradient using the following formula

$$g=\sqrt{gx2\ +\ gy2}$$

$$\theta = \tan^{-1}(\frac{gy}{gx})$$

By using OpenCV, the calculation can be done using the function **cartToPolar.**

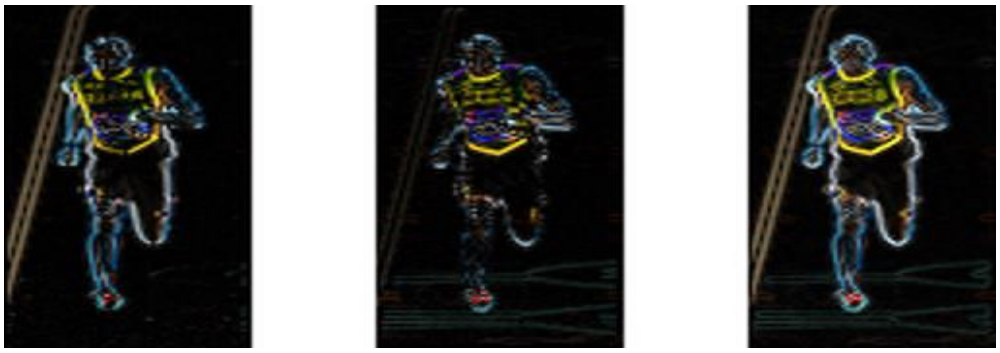


Figure 3.10. Left: Absolute value of x-gradient. Centre: Absolute value of y-gradient. Right: Magnitude of gradient (image curtesy: Satya Mallick Blog)

The luminance gradient is calculated at each pixel. The luminance gradient is a vector with magnitude m and orientation θ represented by the change in the luminance.

The equation is:

$$m\ (x,\ y) =\sqrt{}((L(x+1,y)-L(x-1,y))\ ^2+((L(x,y+1)-L(x,y-1))^2$$

$$\theta(x,y)=\tan^{-1}\left(\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right)$$

Where, L is the luminance of pixel.

The third stage aims to produce an encoding that is sensitive to local image content while remaining resistant to small changes in pose or appearance. The adopted method pools gradient orientation information locally in the same way as the SIFT [53] feature. The image window is divided into small spatial regions, called "cells". For each cell, we accumulate a local 1-D histogram of gradient or edge orientations over all the pixels in the cell. This combined cell-level 1-D histogram forms the basic "orientation histogram" representation. Each orientation histogram divides the gradient angle range into a fixed number of predetermined bins. The gradient magnitudes of the pixels in the cell are used to vote into the orientation histogram.
8×8 cells 9-bin histogram looks like this [54]:

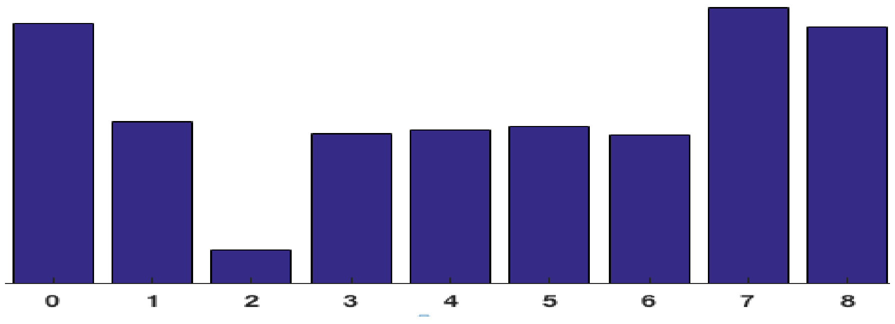


Figure 3.11: 9-bin histogram

The fourth stage computes normalisation, which takes local groups of cells and contrast normalises their overall responses before passing to next stage. Normalisation introduces better invariance to illumination, shadowing, and edge contrast. It is performed by accumulating a measure of local histogram "energy" over local groups of cells that we call "blocks". The result is used to normalise each cell in the block. Typically, each individual cell is shared between several blocks, but its normalisations are block dependent and thus different. The cell thus appears several times in the final output vector with different normalisations. This may seem redundant but it improves the performance. We refer to the normalised block descriptors as Histogram of Oriented Gradient (HOG) descriptors.

The final step collects the HOG descriptors from all blocks of a dense overlapping grid of blocks covering the detection window into a combined feature vector for use in the window classifier.

3.2.3 Feature extraction method:

In order to extract feature (values) from an image first, we convert the original image into a gray scale image. Before gray scaling we cropped the image from the selected area then resized the image into a fixed aspect ratio of 8:3. The image patch is 128×48 which ratio is 8:3 and ready for calculating the HOG descriptor.

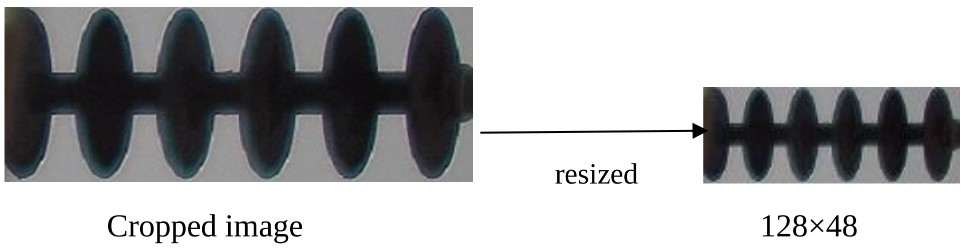


Figure 3.12: Prepare image to calculate the HOG descriptor

To calculate a HOG descriptor, we need to first calculate the horizontal and vertical gradients; after all, we want to calculate the histogram of gradients. We can achieve the results, by using **Sobel** operator in OpenCV with kernel size 1.

```
gx = cv2.Sobel(img, cv2.CV_32F, 1, 0, ksize=1)
gy = cv2.Sobel(img, cv2.CV_32F, 0, 1, ksize=1)
```

we can find the magnitude and direction of gradient using the following formula:

$$g = \sqrt{gx2\ +\ gy2}$$

$$\theta = \tan^{-1}(\frac{gy}{gx})$$

At every pixel, the gradient has a magnitude and a direction. The magnitude of gradient at a pixel is the maximum of the magnitude of gradients of the three channels, and the angle is the angle corresponding to the maximum gradient.

In next step, the image is divided into 8×8 cells and a histogram of gradients is calculated for each 8×8 cells. The histogram is essentially a vector (or an array) of 9 bins (numbers) corresponding to angles 0, 20, 40, 60 … 160. the angles are between 0 and 180 degrees instead of 0 to 360 degrees. These are called **“unsigned” gradients** because a gradient and it’s negative are represented by the same numbers [54]. Next step is to create a histogram of gradients in these 8×8 cells. The histogram contains 9 bins corresponding to angles 0, 20, 40 … 160.

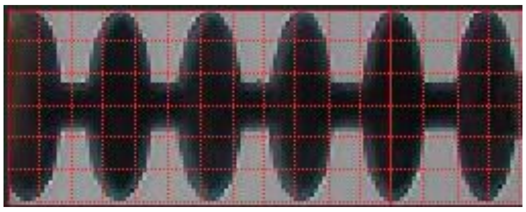
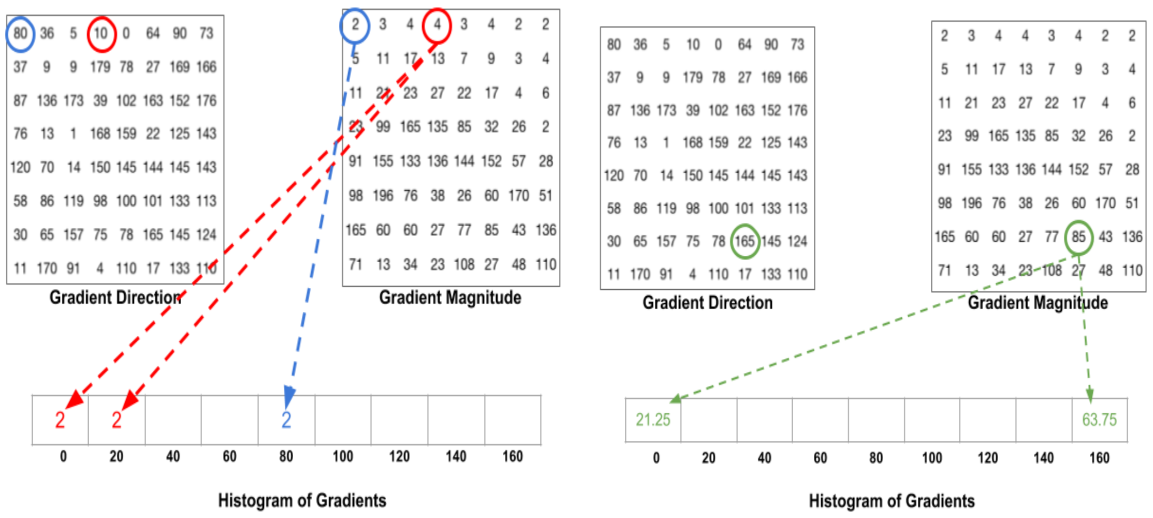


Figure 3.13: 8×8 cells of HOG

We are looking at magnitude and direction of the gradient of the 8×8 patch. A bin is selected based on the direction, and the vote (the value that goes into the bin) is selected based on the magnitude. Let’s first focus on the pixel encircled in blue. It has an angle (direction) of 80 degrees and magnitude of 2. So, it adds 2 to the 5th bin. The gradient at the pixel encircled using red has an angle of 10 degrees and magnitude of 4. Since 10 degrees is half way between 0 and 20, the vote by the pixel splits evenly into the two bins.



Gradient Direction

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Magnitude

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Histogram of Gradients

0	20	40	60	80	100	120	140	160
2	2			2				

Gradient Direction

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Magnitude

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Histogram of Gradients

0	20	40	60	80	100	120	140	160
21.25								63.75

Figure 3.14: Magnitude and Direction look up table (image curtesy: Satya Mallick Blog)

If the angle is greater than 160 degrees, it is between 160 and 180, and we know the angle wraps around making 0 and 180 equivalents [54]. So, in the example below, the pixel with angle 165 degrees contributes proportionally to the 0-degree bin and the 160-degree bin.

we created a histogram based on the gradient of the image. Let’s say we have an RGB color vector [128, 64, 32]. The length of this vector is $\sqrt{128^2 + 64^2 + 32^2} = 146.64$. This is also called the L2 norm of the vector. Dividing each element of this vector by 146.64 gives us a normalized vector [0.87, 0.43, 0.22] [54] .

No, we simply normalize the 9×1 histogram the same way we normalized the 3×1 vector. A

16×16 block has 4 histograms which can be concatenated to form a 36 x 1 element vector and it can be normalized just the way a 3×1 vector is normalized. The window is then moved by 8 pixels and a normalized 36×1 vector is calculated over this window and the process is repeated.
To calculate the final feature vector for the entire image patch, the 36×1 vectors are concatenated into one giant vector.
There are 15 horizontals and 5 verticals positions making a total of 15 x 5 = 75 positions.
Each 16×16 block is represented by a 36×1 vector. So, when we concatenate them all into one giant vector we obtain a 36×75 = 2700-dimensional vector.

We use GPU based HOGDescriptor in openCV for extracting the feature from the image. We have a compute function in OpenCV to extract Descriptors values.
HOGDescriptor function in OpenCV is
Gpu: HOGDescriptor (Size **win_size**=Size(64, 128), Size **block_size**=Size(16, 16),
Size **block_stride**=Size(8, 8), Size**cell_size**=Size(8, 8), int **nbins**=9,
double **win_sigma**=DEFAULT_WIN_SIGMA, double **threshold_L2hys**=0.2,
bool **gamma_correction**=true, int**nlevels**=DEFAULT_NLEVELS)
where

- win_size – Detection window size. Align to block size and block stride.
- block_size – Block size in pixels. Align to cell size. Only (16,16) is supported for now.
- block_stride – Block stride. It must be a multiple of cell size.
- cell_size – Cell size. Only (8, 8) is supported for now.
- nbins – Number of bins. Only 9 bins per cell are supported for now.
- win_sigma – Gaussian smoothing window parameter.
- threshold_L2hys – L2-Hys normalization method shrinkage.
- gamma_correction – Flag to specify whether the gamma correction preprocessing is required or not.
- nlevels – Maximum number of detection window increases.

We set the HOGDescriptor function parameter according to our program. We set window size (128×48) , block size (16 × 16). Block stride or overlapping (8×8) , cell size (8×8) and number of bins 9.
This function computes the hog features
Compute (image, descriptors, winStride, padding, locations)
Which give us the descriptors value. We set padding and winStride to (0,0).
By applying this we get 2700 descriptors value. And, we use this function for every image to extract feature. After find, the value we save them into a text file to train the system. We have

libsvm training procedure so we have to format the file according to lib-svm. The format looks like

+1 1:0.117419 2:0.0915074 3:0.0575953 4:0.0183482 5:0.0274423 6:0.0192094 7:0.177189 8:0.17299 9:0.143298 10:0.129288 11:0.0972997 12:0.0313414 13:0.00684788 …………...

+1: represents the positive image.

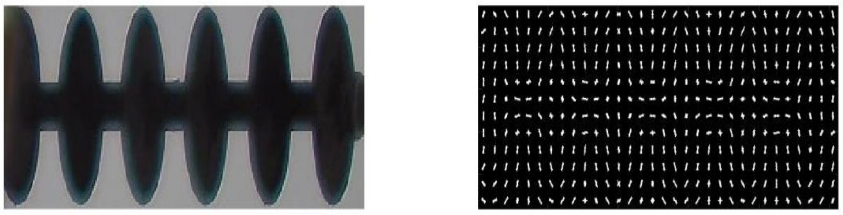


Figure 3.15: Left: original positive image, Right: Extracted HOG feature visualization

3.2.4 Alternative feature extraction method

Rather than HOG there are other methods too which extract feature from the image one of this is Local Binary Pattern (LBP).

We use OpenCV in C++ for extracting the feature from the image. We implement the local Binary Pattern (LBP) method by a code on every image. The code is implemented in C++ language. Now, we explain the LBP method which applying in code. We are applying the LBP count from row 2 and column 2. It assigns 3×3 neighbourhood of each pixel with the center pixel value and the result is considering as a binary number which is calculated by clockwise way. This way it is applicable for the image for every pixel. An example is given below for LBP calculation [55]

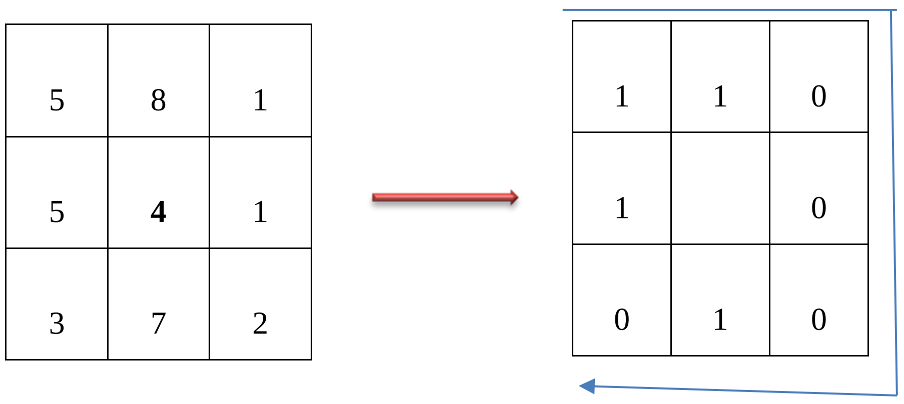


Figure 3.16: The first step in constructing a LBP is to take the 8-pixel neighborhood surrounding a center pixel and threshold it to construct a set of 8 binary digits

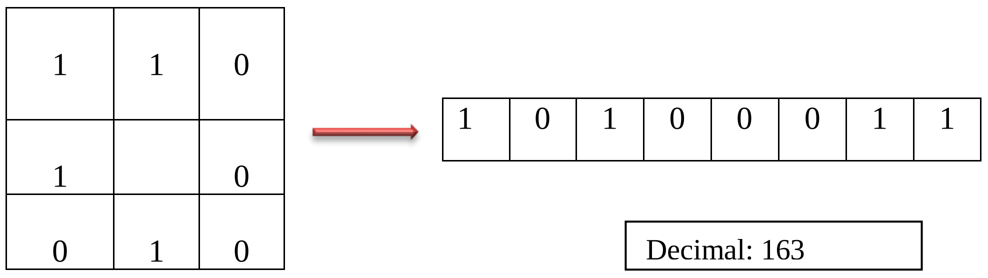


Figure 3.17: 8-bit binary Decimal value

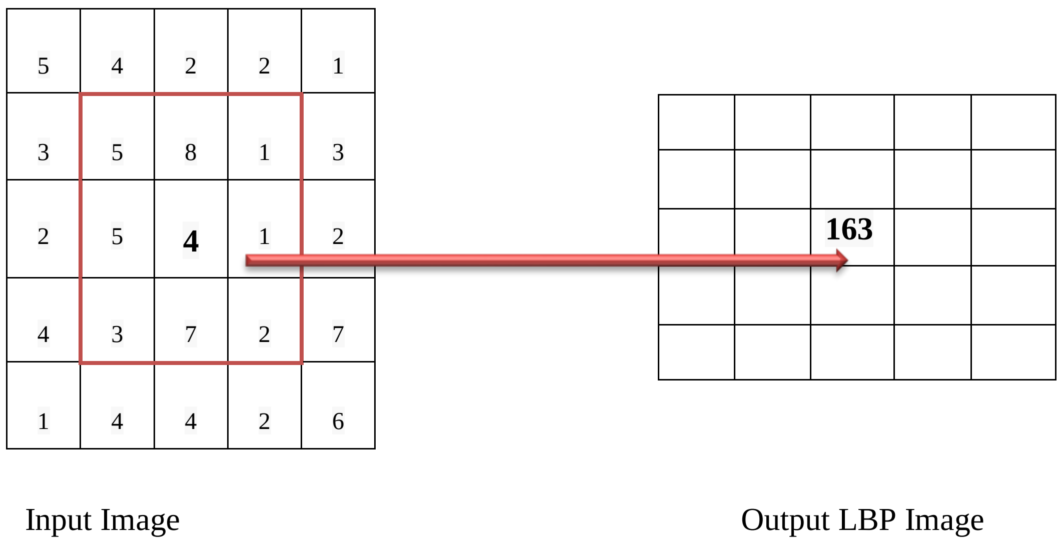


Figure 3.18: Calculated LBP value stored into output array

We compared each pixel with its 8 neighbors where which pixel is greater than the center pixel then it encoded with 1 otherwise with 0. Then we converted those binaries encoded to decimal number in clock wise order. After calculating all values in this way then we count the pixel values range are within 0 to 255. For other variation in encoded values, we follow same procedure, but those values saved another file.

We divide the whole image first in such way (here 2×2) that full image divided equally. We count the pixel values of every divided part (which in number 4) ranged to 0 to 255. This process will be followed for every image. We also divide the image into 2×4 where image not equally divided. And, we also have to count the pixel value of all 8 part in range of 0 to 255.

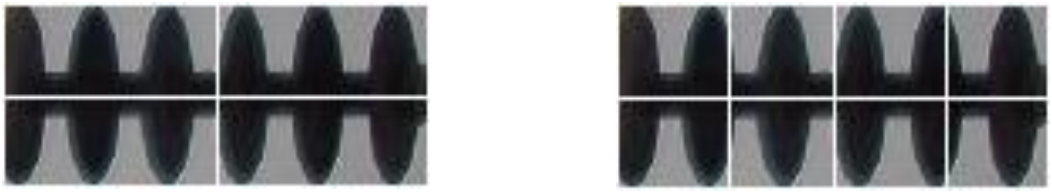


Figure 3.19: Left: An insulator image divided into 2×2 sub-regions, Right: An insulator image divided into 2×4 sub-regions

For an image, we count the pixel value from 0 to 255 that save count 1 to 256 positions to a text file in libsvm tool format. This is for non-divided image. For dividing image, we count pixel value for every partition from 0 to 255 and save count 1 to 256 positions. For the second portion the count value increase which starts from 257 and represents how many zeros count in that portion. The other portions counting value of same image will be increase same way. For the 2×2 divided image we found 1024 values and 2×4 divided image we found 2048 values. For 2×2 divided image we have four portion in that for every portion we get 1364 value from 0 to 255. We count them as how many zeros or ones in them like this. And, we store them into a file in libsvm format start with 1. For first portion we have 256 values and we store them in 1 to 256 as well as for the other portion we have also 256 values and we start store them from 257 which represent the count value of zeros.

Text file format is given bellow:

256:971 257:2 258:0 259:0 260:0 261:0 262:0 263:1 264:2 265:2 266:0 267:0 268:0 269:1

1013:0 1014:0 1015:0 1016:1 1017:9 1018:1 1019:0 1020:6 1021:1 1022:3 1023:5 1024:1173

Here, the value we have found after counted is our feature value as like here in 256 we have the value of 971 which is the value of zeros. And, 1024 is the value of 255 as we counted 1173 in the fourth portion.

We run this process for all the image in our dataset and save the counted value in the text file.

3.2.4 Training For Insulator Detection

3.2.4.1 Image Data-set:

For training, we used cropped version of the insulator images. First, we cropped down insulator from different images. As all the insulator are not horizontal some are rotated so we fixed them. And, make sure that all the insulator in the same uniform manner. We not only take the insulator but also take insulator with some part wire or other parts of the power line that cover insulator. There are almost 215 insulators which is counted as our positive dataset. All the positive images have a fixed aspect ratio of 8:3. So, when we train our data we resized images into (128×48) so that they maintain the ratio. We also used 385 non-insulator images in training to define the non-insulator class. This non-insulator images are our negative sample. We also resized our negative samples into (128×48) so that they maintain aspect ratio. Totally, we have 600 positive and negative images in our dataset [21].

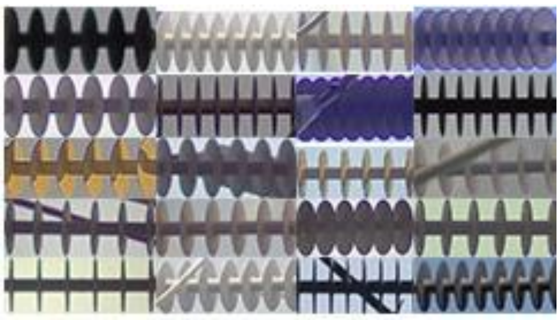


Figure 3.20: positive image dataset

3.2.4.2 Training Using LIBSVM

After taking all the images into GRAY scale we apply HOG for each insulator and non-insulator image we obtain feature values to construct a feature-vector. Each image is of size 128 × 48. As our window is 128 × 48, block size is 16×16, block overlapping 8×8, cell size 8×8 and bins are 9. 16×16 block means it will take 4 cells which has 9 bins individually. So, from our calculation the descriptors value should be 4×9×15×5= 2700. So, after applying HOG on our image we also got 2700 descriptors value. We apply HOG on every image and get training feature vector x_i. These feature vectors were saved in a text file with a label as a training file. For an insulator-image, the label is +1 and for non-insulator image, label is -1. These values with label are saved in certain format so that LIBSVM [12] can read, analyse and produce a classifier. We completed the scaling of each attribute to the range between 0 and 1 before training procedure. Training vectors are mapped into a higher dimensional space. LIBSVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space [21].

In the training, as kernel, we used radial basis function (RBF):

$K (Fi; Fj) = \exp (-\gamma \|Fi – Fj\|2)$

$\gamma > 0$: $C > 0$ is the penalty parameter of the error term. We set $\gamma = .01$; $C = 8$ for training. Since the number of insulator images is not larger than the number of non-images, so we do not provide any larger penalty for non-insulator image class than insulator image class using -w in LIBSVM. Support vector machine performs an implicit mapping of data from lower to higher dimension feature space. We use C-SVC type SVM.

For train data set, we need to use command mode. By extracting feature value and write it into text file we directly use it into command mode but in our program, we use system call to use it. For using command line at first, we keep the LIBSVM in our c drive or any other drive. In LIBSVM there is a folder called window. Here, many exe files we see like svm-train.exe, svm-predict.exe etc. We keep the text file into this folder which contains the training dataset.

First, we go to the command mode window then follow some procedure: -

- C:\Users\user> cd path
- C:\Users\user\libsvm\window> svm-train [options] training_set_file [model_file]

LIBSVM training command shown below:

- svm-train.exe -s 0 -c 8 -t 2 -g .01 trainfile.txt

Or

- svm-train.exe -s 0 -c 8 -t 2 -g .01 trainFile.txt. scale [if we take our data into scale]

In Microsoft visual studio:

- system ("svm-train.exe -s 0 -c 8 -t 2 -g .01 trainFile")

This command gives us a model which We will use later, for classification.

- svm-train.exe -s 0 -c 8 -t 2 -g .01 -v 5 trainfile.txt

It will give a cross-validation result of 5-fold which means it divides training data set into 5 parts where 4 parts used for train dataset and one part is used for test dataset.

parameters:

-s: C-SVC (if s=0, then the SVM type is C-SVC) only.

-c: set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)

-t: radial basis function: exp(-gamma*|u-v|^2)

-g: set gamma in kernel function (default 1/num_features)

-v: Cross Validation Accuracy

3.2.4.3 Hard-negative mining

For each image and each possible scale and rotation in our training set, we apply the sliding window technique and slide our window across the image. At each window, we compute our HOG descriptors and apply our classifier. If your classifier (incorrectly) classifies a given window as an insulator (and it will, there will absolutely be false-positives), we record false positive image. This approach is called hard-negative mining [55].

Take the false-positive samples found during the hard-negative mining stage, put them into our negative samples folder as a negative image and re-train our classifier using those hard-negative samples. Often performance can be improved by re-training the object detector with a set of negative examples that has been augmented with false-positives from an initial run of the object detector. Intuitively, this makes a much better negative training set than the random patches chosen initially. The gains in accuracy on subsequent runs of hard-negative mining tend to be minimal.

3.2.5. Classification (Insulator Detection)

After completion of training successfully, we apply sliding window based technique to identify and localize insulators in an image from our test image set. In this approach, we slide a rectangular window with fixed width and height across an image from left-to-right and top-to-bottom. First, we apply this technique in our original image then repeated on successively scaled copies of the image. For each detected window(sub-image), we normally take that sub-image and convert it into GRAY scale image as well as apply HOG to extract descriptors value and store them into text file to classify. This process is repeated on full image as well as successively scaled copies of the image so that objects can be detected at any size.

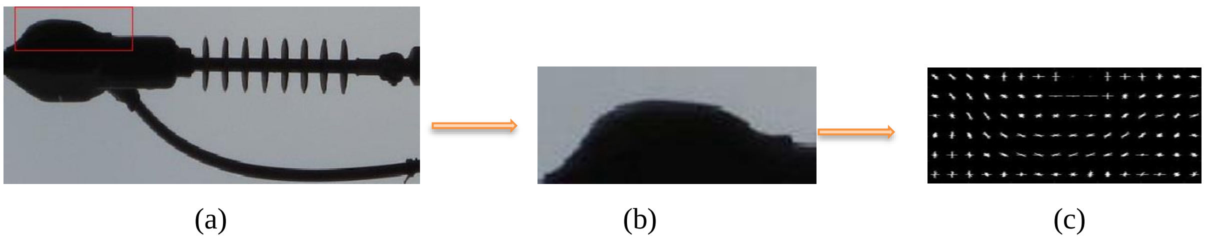
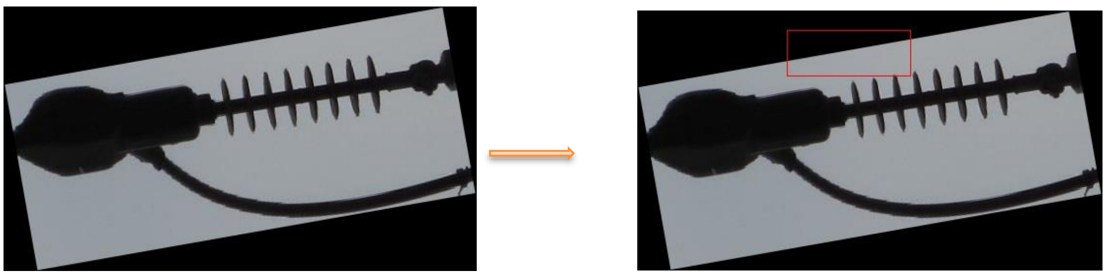


Figure 3.21: (a) Scanning image with a fixed-size rectangular window (b) Sub-image defined by the window (c) Apply HOG on sub-image

We apply this process in image with zero-degree rotation. After that we rotate the image and apply all the techniques we used before. We increment the angle of rotation by 10-degree until it reaches 360-degree. In rotated image, we apply sliding technique combined with image pyramid. With fixed size rectangular window, we scan rotated image as well as for every sub-window we calculate descriptors value and applying a classifier to detect that particular bounds is our insulator or not.



(a) 10-degree rotation in 100% size (b) sliding window in rotational image

Figure 3.22: Sliding window technique on rotated image with fixed detection window

After successfully applying sliding window we bound a window and extract HOG descriptors value. And, save them into a single test instance text file. We use this text featured file to classify insulator and non- insulator using svm-predict function of LIBSVM.
We write below code

- svm-predict.exe test_data test.model result.txt

parameters:

test_data: HOG features value of a sub-image
test.model: classifier
result.txt: predicted label

Model has all necessary parameters for svmpredict.
If the variable classifier is greater than 0, we consider the instance as insulator.
Else, classifier < 0 and the instance is a non-insulator.

At each window, we compute HOG descriptors and apply our classifier. our classifier incorrectly classifies a given window as an insulator which is absolutely be a false positive image. When we complete our scanning, we get lots of false positive image so we apply hard-negative mining approach. And, retrain our classifier with incorrectly classified image with negative samples. Performance improved by re-training classifier with a set of negative examples that has been augmented with false-positives from an initial run of the classifier.

Now, for each scale and rotation we apply the sliding window technique. At each window extract HOG descriptors and apply our classifier. After classify our insulator, we record the bounding box of the window. Those windows bounded our desire object.

3.6 Non- maximum suppression (NMS)

The output had multiple detections of the same object. So, we applied non-maximum suppression to remove redundant and overlapping bounding boxes.

Non-maximum suppression is used as an intermediate step in many computer vision algorithms. Non-maximum suppression is often used along with edge detection algorithms. The image is scanned along the image gradient direction, and if pixels are not part of the local maxima they are set to zero. This has the effect of supressing all image information that is not part of local maxima [56].

Approaches based on sliding windows [43-45] typically produce multiple windows with high scores close to the correct location of objects. This is a consequence of the generalization ability of object detectors, the smoothness of the response function and visual correlation of close-by windows. This relatively dense output is generally not satisfying for understanding the content of an image. As a matter of fact, the number of window hypotheses at this step is simply uncorrelated with the real number of objects in the image. The goal of NMS is therefore to retain only one window per group, corresponding to the precise local maximum of the response function, ideally obtaining only one detection per object. Consequently, NMS also has a large positive impaction performance measures that penalize double detections. [57]

The most common approach for NMS consists of a greedy iterative procedure [43, 44], which we refer to as Greedy NMS. The procedure starts by selecting the best scoring window and assuming that it indeed covers an object. Then, the windows that are too close to the selected window are suppressed. Out of the remaining windows, the next top-scoring one is selected, and the procedure is repeated until no more windows remain. This procedure involves defining a measure of similarity between windows and setting a threshold for suppression. These definitions vary substantially from one work to another, but typically they are manually designed. Greedy NMS, although relatively fast, has a number of downsides. First, by suppressing everything within the neighbourhood with a lower confidence, if two or more objects are close to each other, all but one of them will be suppressed. Second, Greedy NMS always keeps the detection with the highest confidence even though in some cases another detection in the surrounding might provide a better fit for the true object. Third, it returns all the bounding-boxes which are not suppressed, even though many could be ignored due to a relatively low confidence or the fact that they are sparse in a sub-region within the image. [57]

Rather than greedy NMS method, there are multiple ways to remedy this problem. Triggs et al. suggests to use the Mean-Shift algorithm to detect multiple modes in the bounding box space by utilizing the (x, y) coordinates of the bounding box as well as the logarithm of the current scale of the image. [55]

However, no matter what method of object detection we use, we will likely end up with multiple bounding boxes surrounding the object we want to detect. In order to remove these redundant boxes, we apply Non-Maximum Suppression.

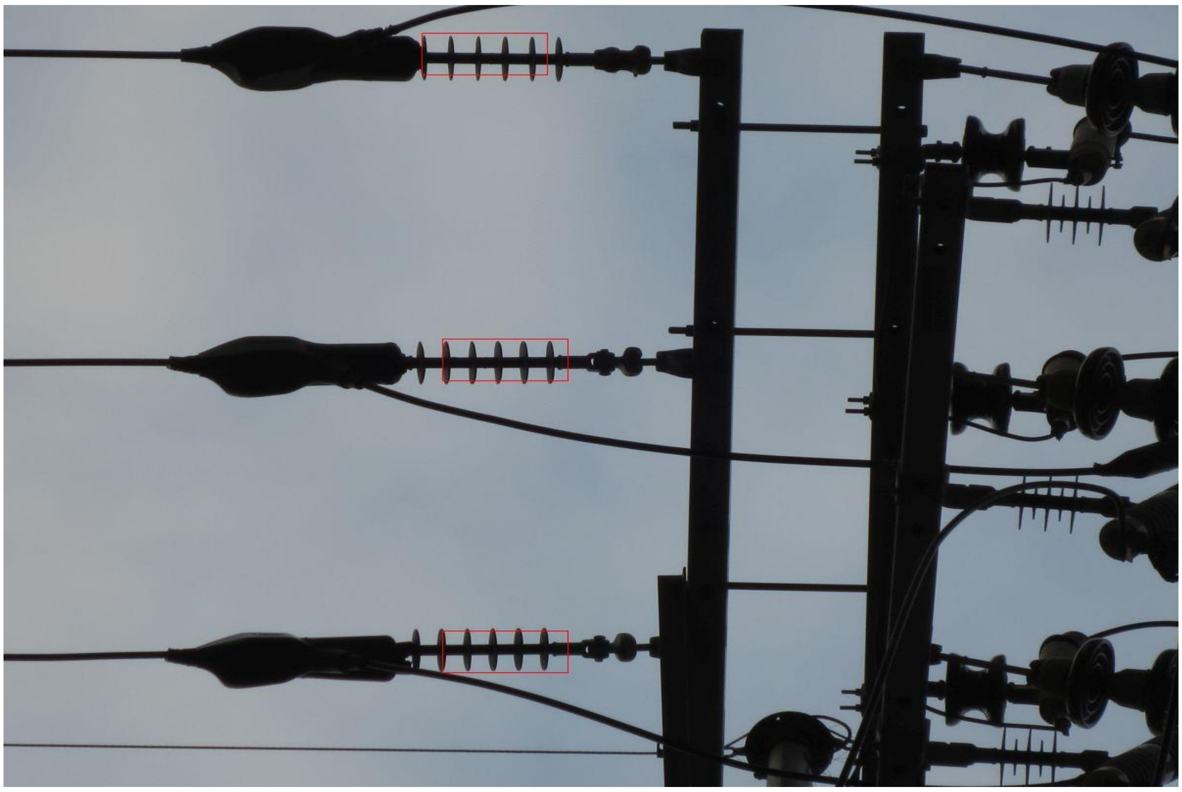


Figure 3.23: Applying non-maximum suppression, correctly able to reduce the number of bounding boxes

Chapter 4

Experimental Result and Performance Analysis

In chapter 3, We discussed about the test of insulator detection based of Histogram Oriented Gradient (HOG). Here, we perform a comprehensive experiment. The experimental result and discusses focuses on the performance of the result analysis of the system which is divided into several categories.

4.1 Experimental result

In this chapter we discuss about our experimental result on HOG and other feature extraction method. The result and discussion focusses on the performance of our proposed method. In section 4.1.1 and4.1.2 we discuss about the result of pole and insulator detection. In section 4.1.3 we discuss about the result of other feature extraction method.

4.1.1 Pole Detection result

We tested about 26 images to find out pole with x-axis threshold 5 pixels and variation in theta $\pm 5^0$ as an empirical value. In 26 images, we found pole in our best three lines in 25 images. We draw the them into red, green and blue. We found pole in our first best line in 20 images, second best in 4 images and last one in on image. We found no result in one image.

In total we got, 96.154% accuracy to find a pole in 26 images with three best coverage lines.

Table 4.1: Accuracy of pole detection with empirical value 5 pixels and theta $\pm 5^0$

Best Coverage Line	Total Image	Detected Pole (26 image)	Accuracy (First Line)
Line one (Red)	26	20	80%
Line two (Green)	26	4	16%
Line three (Blue)	26	1	4.167%

Here, our first best line detects the pole so we don't need to look for the pole in the other two lines which is shown in Red color.

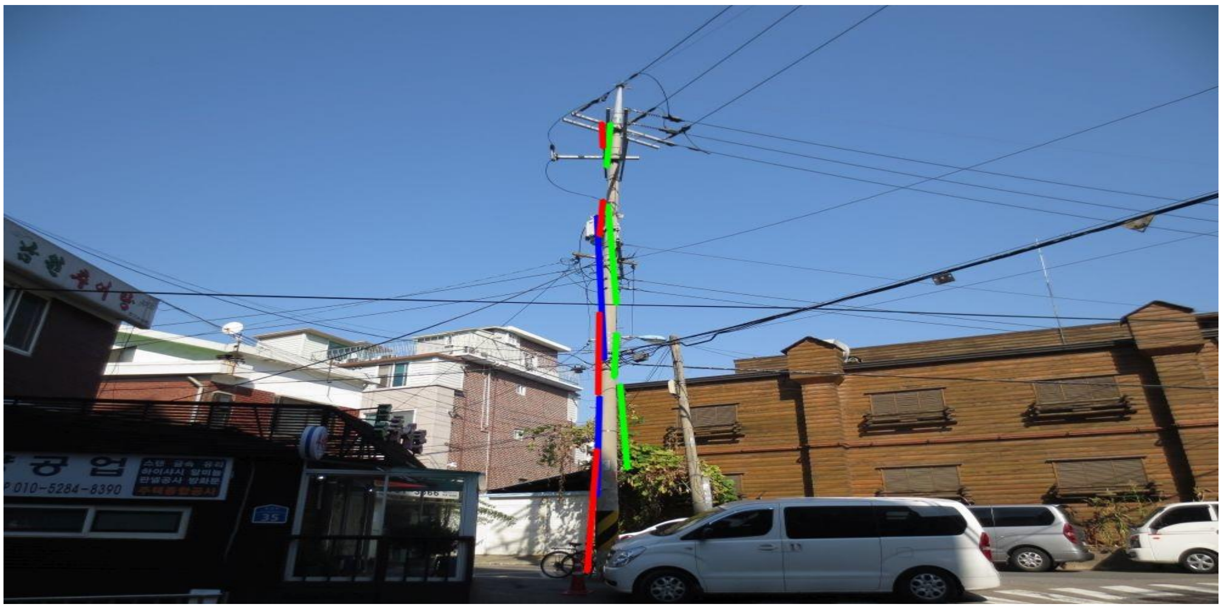


Figure 4.1: Detected pole shown in Red, Green and Blue (Three best lines)

We got no result in one image from 26 images. This happened because in that image all the building's corner counted as a line so when we calculate coverage area those lines were counted. So, we need more pre-processed data from overcome this type of situation.

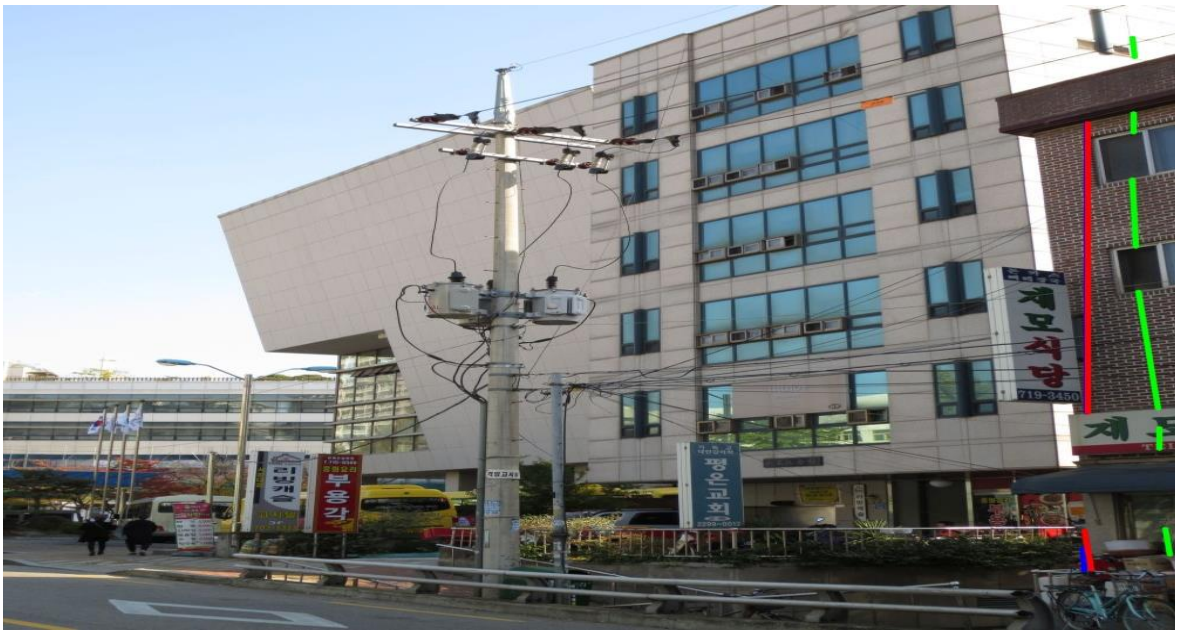


Figure 4.2: Non Pole Detection Image

4.1.2 Insulator result

To evaluate the performance of features, we ran various cross-validation test in **LIBSVM**. The number of test classes are 2. In the first class, we have 215 images of insulator. In the second class, there are 385 images of non-insulator. Features are taken from both classes. These features are used for classification by the standard classifier of **LIBSVM**. So, the performance of features is directly co-related to the performance of classifier.

Table 4.2: **Accuracy of classifier for Radial Basis Function under different parameters**

No.	Sample Size	Cost	Gamma	Cross Fold Validation	Accuracy
1	600	512	.0001	5	99.5%
2	600	256	.0001	7	98.8333%
3	600	512	.0005	9	99.3333%
4	600	128	.005	20	99.1667%
5	600	64	.001	50	99.3333%

From Table 4.2 we see that for different gamma, cost and fold we got different results. For gamma= .0001, cost= 512 and fold No.=5 we got the highest accuracy for classifier. And, other results have variation in accuracy but we got better result in fold No. 5 as well as There is a little difference between different folds' results.

For better accuracy of classifier, we searched best value of γ; C through grid.py and easy.py (Python program).

This programs do grid-search for γ; C in an interval, the instruction command in Windows. Command Prompt is

- python grid.py Feature Value
- python easy.py Feature Value

Both command we can use for finding the best gamma and cost. Grid.py is only search for best gamma and cost which maximize our classifier. And, easy.py does everything automatic from data scaling to parameter selection. Both use Gnuplot for visualizing contour plot.

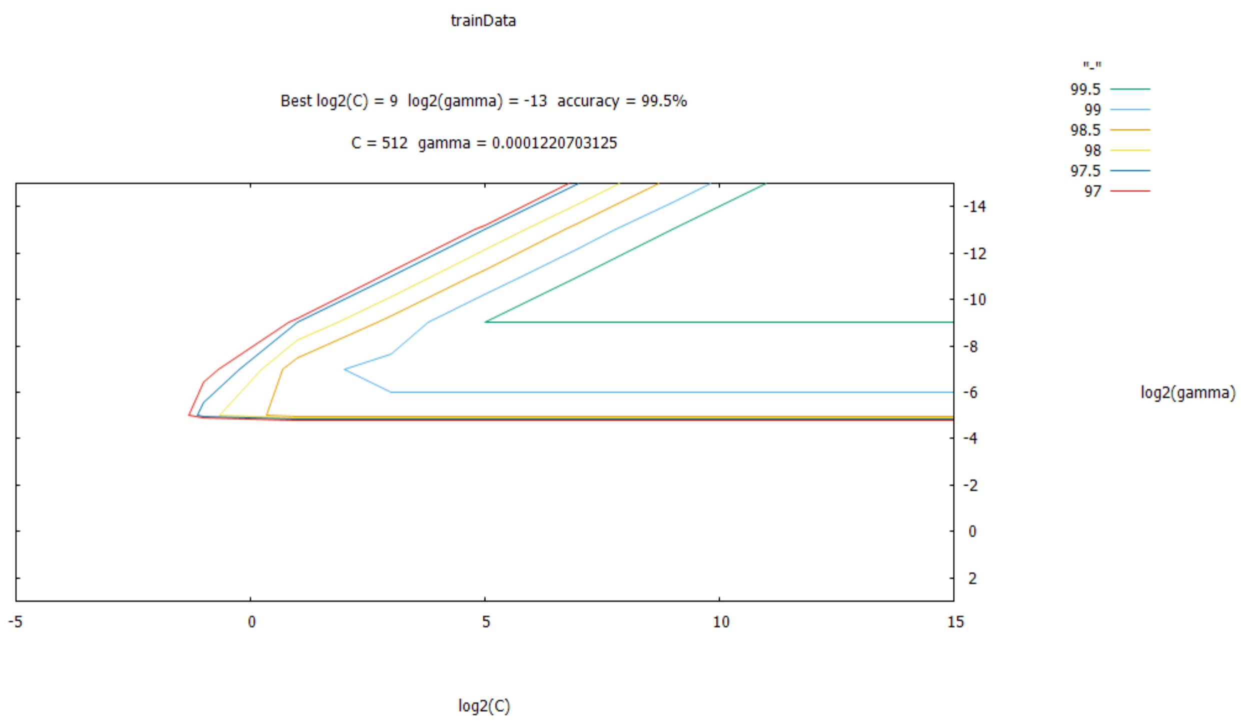


Figure 4.3: Contour plot of our data running in LIBSVM to find best gamma and cost

ROC - Receiver Operating Characteristics graphs are a useful technique for organizing classifiers and visualizing their performance. [58]

The ROC curve of our classifier is

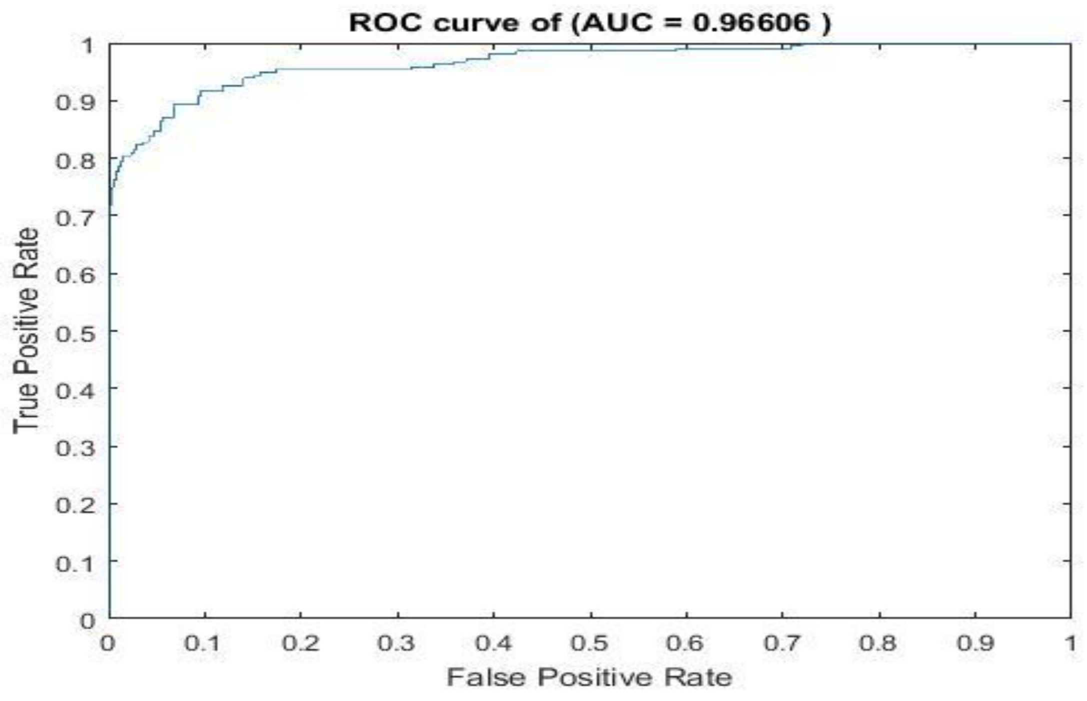


Figure 4.4: ROC curve of classifier

4.1.2.1 Visualize Experimental Result

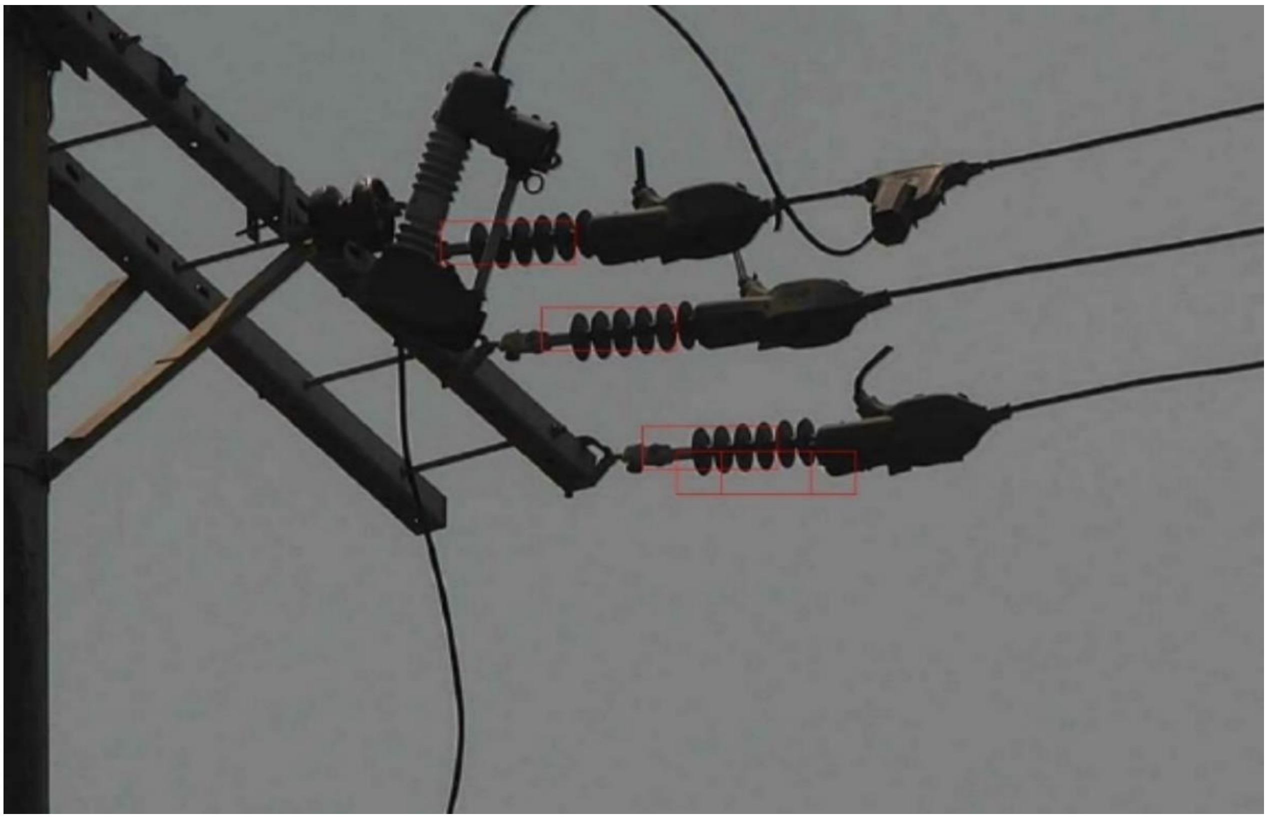


Figure 4.5: Horizontal insulator detection

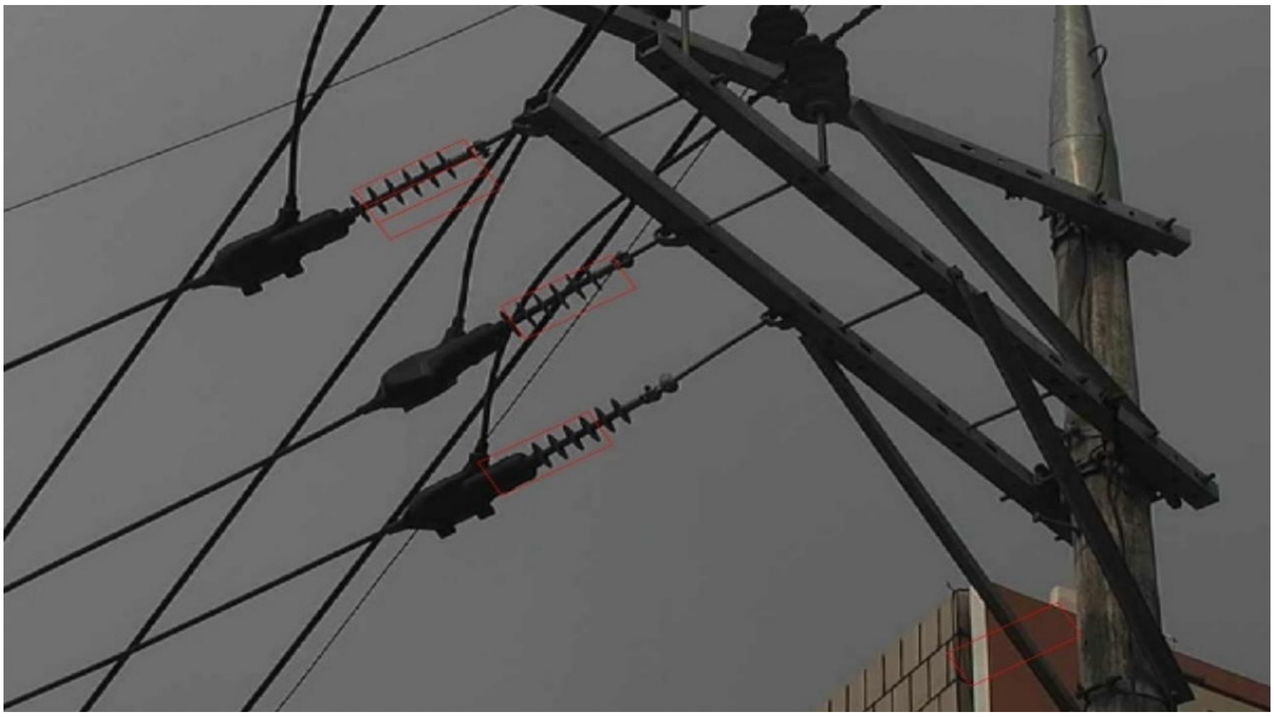


Figure 4.6: Angled insulator detection

4.2 Performance Analysis

In pole detection, with our x- axis threshold value 5 pixels and theta difference ±5^0 we got 96.154% accuracy to detect a pole in an image and our first best line detect 80% pole in 26 images. We try different threshold value and theta for our experiment. Below we shown the comparison between different empirical values:

Table 4.3: Comparison between different empirical value

No.	Different empirical value	Total Image	Detected pole (26 image)	Total Accuracy
1	Difference=5 pixels Angle= 5^0	26	25	96.154%
2	Difference=3 pixels Angle= 3^0	26	24	92.308%
3	Difference=5 pixels Angle= 3^0	26	24	92.308%
4	Difference=5 pixels Angle= 7^0	26	24	92.308%

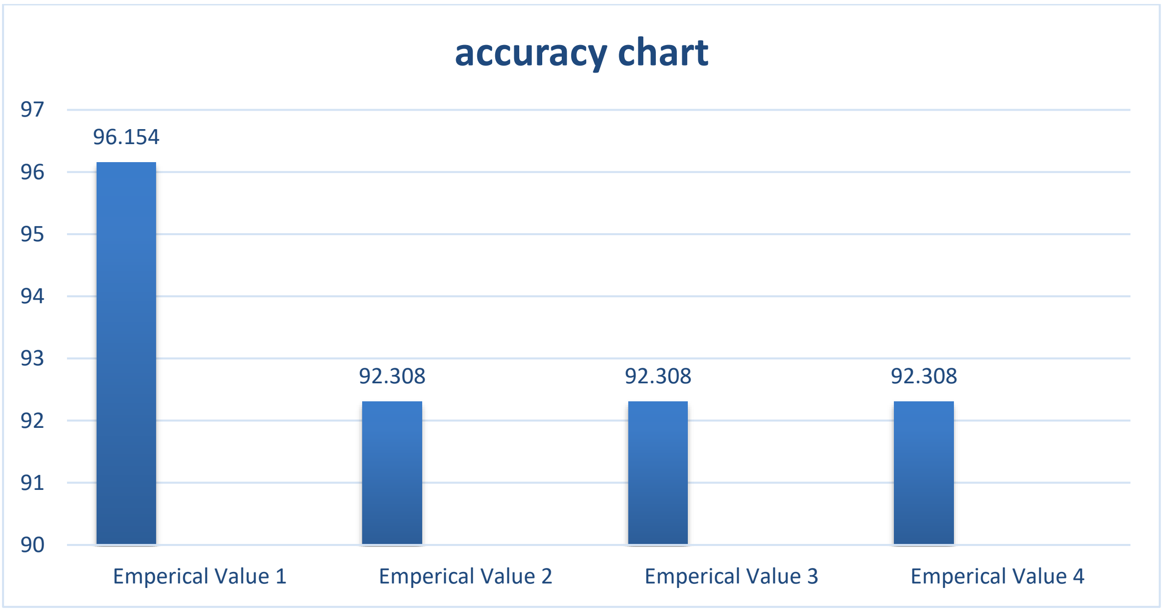


Figure 4.7: Comparison of different empirical value's accuracy

The performance analysis module tests the proposed system with different feature value to detect insulator using support vector machine. We used Local Binary Pattern(LBP) and LBP with HOG to saw how they perform in insulator detection. The Results of LBP and LBP-HOG are shown below:

Table 4.4: Accuracy of classifier for Radial Basis Function using LBP (images divided into 2×2) in SVM (LIBSVM) for different γ, C and V

No.	Sample Size	Cost	Gamma	Cross Fold Validation	Accuracy
1	600	2	.00001	5	98.5%
2	600	8	.00002	7	97.5%
3	600	16	.00003	9	96.3333%
4	600	64	.00001	20	98.3333%
5	600	32	.00004	50	93.8333%

Table 4.5: Accuracy of classifier for Radial Basis Function using LBP (images divided into 2×4) in SVM (LIBSVM) for different γ, C and V.

No.	Sample Size	Cost	Gamma	Cross Fold Validation	Accuracy
1	600	8	.00001	5	97.8333%
2	600	8	.00003	7	97.1667%
3	600	128	.000001	9	98.5%
4	600	64	.000009	20	97.3333%
5	600	32	.000003	50	98.1667%

Table 4.6: Accuracy of classifier for Radial Basis Function using LBP-HOG in SVM (LIBSVM) for different γ, C and V

No.	Sample Size	Cost	Gamma	Cross Fold Validation	Accuracy
1	600	8	.005	5	96.3333%
2	600	8	.007	7	95.5%
3	600	32	.005	9	95.3333%
4	600	64	.009	20	95.8333%
5	600	8	.003	50	96%

Table 4.7: Accuracy of classifier for Radial Basis Function using LBP-HOG (images divided in 2×2) in SVM (LIBSVM) for different γ, C and V

No.	Sample Size	Cost	Gamma	Cross Fold Validation	Accuracy
1	600	8	.008	5	96.5%
2	600	16	.01	7	96.3333%
3	600	8	.01	9	96.8333%
4	600	32	.009	20	96.3333%
5	600	8	.003	50	95.1667%

From our experiment we got different cross validation result for different feature. By dividing images into (2×2) and images divided into 2×4 (LBP) we got accuracy of our classifier 98.5% and 97.8333% in 5-fold cross validation. We used LBP and HOG feature together and got accuracy 96.3333%. By dividing images into 2×2 using LBP-HOG feature together we got 96.5% result in 5-fold cross validation. Compare to LBP and LBP-HOG LBP gives little better result than LBP-HOG.

Table 4.8: Overall performance of different features accuracy

Feature	Cross Fold Validation	Accuracy
HOG	5	99.5%
LBP (2×2)	5	98.5%
LBP (2×4)	5	97.8333%
LBP-HOG	5	96.3333%
LBP-HOG (2×2)	9	96.8333%

Chapter 5

Conclusion

In this chapter we summarize the project work. Contribution of our proposed system presents in section 5.1. The limitation of our system discuss in section 5.2. We make final concluding remarks with few directions for future works and improvements in section 5.3.

5.1 Contribution of proposed system

In last, few years many works have been done in power line insulator detection and fix the faultiness of insulator in the field of computer vision. Therefore, insulator detection is challenging task in power line system.
we discuss, a computer vision based technique to detect power line insulator using support vector machine using LIBSVM tool. As Electrical pole poses a great role to detect insulator so we detect different types of electrical pole by using line detection approach.
From extracted data for insulator LIBSVM used to train the machine. These features are extracted by using Histogram Oriented Gradient method that saved in a text file. The system trained by SVM (LIBSVM tool) and tested by using sliding window technique with pyramid image. We also apply rotational- invariant method to detect insulator at varying angle with the help of sliding window technique. The result of our experiment is given overall satisfactory.

5.2 Limitation of the System

A major problem in detection of electrical pole is that if we have not found any pole in our best three lines then we have to take another angle picture or considering other logic to detect. The time complexity of our pole detection is $O(n^2)$. When we have lots of line then it took extra time for grouping lines which is the drawback of our pole detection system. If there are lamp post or other vertical object, then our system cannot compete with them as well as if there is high rise building, our system detects extensive amount lines than system failed to detect pole.
Sliding window technique fail to take into account contextual cues as well as the window has a

fixed aspect ratio making it difficult for articulated insulators or insulators with large intra-class variation. It is time consuming because it slides the image at a fixed amount. If sliding amount is greater it cannot detect insulator. It searches at every possible scale and rotation so it took more time to terminate. It detects lots of false positive which is time consuming for re-train the classifier.

5.3 Future Work and Improvement

Rather than problem in pole detection improvement can be made by reducing false detection. We take count on horizontal lines and try to make a cross which we see in the pole or may be two horizontal lines along with vertical lines. We try to confide on the texture of the pole and other objects but it does work properly. As well as we can take count on the width of the object. High rise building line detection problem can be solved easily by this. Our next plan how could we more accurately detect pole at low cost and reduce the time consumption.

When we took insulator image in our dataset we have not applied any pre-processing technique to normalize the effect of illumination which is a major issue. Due to the different amount of light, the intensity value of the image tends to vary a lot, thus it makes hard to produce consistent result. When we re-sized images lots of information have been lost. We will apply faster sliding window technique to boost slide onto the image fast. We also try to classify insulator without using LIBSVM. We will try different classifier like-KNN, ANN Etc. to classify insulator. We also hope that, HOG based feature will perform better in sophisticated machine learning algorithms like Adaptive Boosting, Convolutional Neural Network [21].

In future, we plan to detect fault of the insulator and make our approach more robust in detect insulator. In real life insulator faultiness or injury finding is more important. As we detected insulator our next plan is to find out the faultiness of insulator which is a challenging task for every researcher.

In spite of some problems in our proposed system, the performance of the technique can play a significant role in terms of detection and memory.

References

[1] "Detection and classification of defects in ceramic insulators using RF antenna – IEEE Xplore Document",Ieeexplore.ieee.org.

Available: http://ieeexplore.ieee.org/document/7873474/.

[2] http://or.nsfc.gov.cn/bitstream/00001903/151571/1/1000013766367.pdf.

[3] R. Gaur, "A Novel Analytical Approach For The Protection Of The Wireless Network", Rroij.com.

Available:https://www.rroij.com/open-access/a-novel-analytical-approach-for-the-protection-of-the-wireless-network-22-25.php?aid=37756.

[4] "Computer vision", ScienceDaily,

Available: https://www.sciencedaily.com/terms/computer_vision.htm.

[5] "Utility pole", En.wikipedia.org,

Available: https://en.wikipedia.org/wiki/Utility_pole.

[6] "Image Analysis-Based Automatic Utility Pole Detection for Remote Surveillance - IEEE Xplore Document", Ieeexplore.ieee.org,

Available: http://ieeexplore.ieee.org/document/7371267/.

[7] "Power Pole Detection Based on Graph Cut - IEEE Xplore Document", Ieeexplore.ieee.org,

Available: http://ieeexplore.ieee.org/abstract/document/4566577/.

[8] "Detection of Pole-like Objects from LIDAR Data - ScienceDirect", Sciencedirect.com,

Available: http://www.sciencedirect.com/science/article/pii/S187704281630595X.

[9] "Pole-like object detection and classification from urban point clouds - IEEE Xplore Document",Ieeexplore.ieee.org,

Available: http://ieeexplore.ieee.org/document/7139615/. [Accessed: 15- Aug- 2017].

[10] "An optimal graph theoretic approach to data clustering: theory and its application to image segmentation - IEEE Xplore Document", Ieeexplore.ieee.org,

Available: http://ieeexplore.ieee.org/document/244673

[11] "A new graph-theoretic approach to clustering and segmentation - IEEE Xplore Document",Ieeexplore.ieee.org,

Available: http://ieeexplore.ieee.org/document/1211348/.

[12] "A Robust Insulator Detection Algorithm Based on Local Features and Spatial Orders for Aerial Images - IEEE Xplore Document", Ieeexplore.ieee.org,

Available: http://ieeexplore.ieee.org/abstract/document/6979220/.

[13] http://cmp.felk.cvut.cz/cvww2014/papers/19/19.pdf.

[14] "Defects Detection of Glass Insulator Based on Color Image--《Power System Technology》2011年01期", En.cnki.com.cn,

Available: http://en.cnki.com.cn/Article_en/CJFDTotal-DWJS201101023.htm.

[15] L. Junfeng, L. Min and W. Qinruo, "A Novel Insulator Detection Method for Aerial Images".

[16] "A Method of Insulator Detection from Video Sequence - IEEE Xplore Document",

Ieeexplore.ieee.org, Available: http://ieeexplore.ieee.org/document/6495370/.

[17] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Proc. of CVPR, 2001

[18] F. Crow, Summed-area tables for texture mapping, in: Proc. of SIGGRAPH, Vol. 18, 1984, pp. 207212

[19] Y. Freund, R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, in: European Conf. on Computational Learning Theory, 1994.

[20] Y. Freund, R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55 (1) (1997) 119139

[21] Fahad Bin Mortuza, Kernel-Coefficient Based Feature For Face-Detection. 2017.

[22] "11.1 Classification Techniques", Wtlab.iis.u-tokyo.ac.jp,

Available: http://wtlab.iis.u-tokyo.ac.jp/~wataru/lecture/rsgis/rsnote/cp11/cp11-1.htm.

[23] "An Introduction to Classification: Feature Selection", Improvedoutcomes.com,

Available:http://www.improvedoutcomes.com/docs/WebSiteDocs/Classification_and_Prediction/SLAM/An_Introduction_to_Classification.htm.

[24] "Introduction to Support Vector Machines — OpenCV 2.4.13.3 documentation", Docs.opencv.org,

Available:http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html.

[25] Vapnik, V. N., and Chervonenkis, A. Y. Theory of Pattern Recognition [in Russian].

Nauka, USSR, 1974.

[26] Cortes, C., and Vapnik, V. Support-Vector Networks. In Machine Learning (1995), vol.20, pp. 273{297.

[27] http://www.bytefish.de/pdf/machinelearning.pdf.

[28] E. Osuna, R. Freund, F. Girosi, Training support vector machines: An application to face detection, in: Proc. of CVPR, 1997

[29] B. Heisele, T. Poggio, M. Pontil, Face detection in still gray images, Tech. rep., Center for Biological and Computational Learning, MIT, A.I. Memo 1687 (2000)

[30] S. Romdhani, P. Torr, B. Scholkopf, A. Blake, Computationally efficient face detection, in: Proc. of ICCV, 2001

[31] M. Ratsch, S. Romdhani, T. Vetter, Efficient face detection by a cascaded support vector machine using haar-like features, in: Pattern Recognition Symposium, 2004

[32] P. Wang, Q. Ji, Multi-view face detection under complex scene based on combined svms, in: Proc. of ICPR, 2004. rep., Microsoft Research, MSRTR-2001-09 (2001)

[33] K. Hotta, View independent face detection based on combination of local and global kernels, in: International Conference on Computer Vision Systems, 2007

[34] http://web.mit.edu/6.034/wwwbob/svm.pdf.

[35]http://www.cogsys.wiai.uniamberg.de/teaching/ss06/hs_svm/slides/SVM_Seminarbericht_Hofmann.pdf.

[36] "LIBSVM -- A Library for Support Vector Machines", Csie.ntu.edu.tw,

Available: https://www.csie.ntu.edu.tw/~cjlin/libsvm

[37] https://www.cs.cornell.edu/people/tj/publications/joachims_06a.pdf.

[38] A. Rosebrock, "Sliding Windows for Object Detection with Python and OpenCV PyImageSearch",PyImageSearch,

Available: http://www.pyimagesearch.com/2015/03/23/sliding-windows-for-object-detection-with-python-and-opencv/.

[39] " Image Pyramids — OpenCV 2.4.13.3 documentation", Docs.opencv.org,

Available: http://docs.opencv.org/2.4/doc/tutorials/imgproc/pyramids/pyramids.html.

[40A.Rosebrock,"Image Pyramids with Python and OpenCV PyImageSearch", PyImageSearch, Available:http://www.pyimagesearch.com/2015/03/16/image-pyramids-with-python-and-opencv/.

[41] "Electric Poles | Electrical4u", Electrical4u.com,

Available: https://www.electrical4u.com/electric-poles/.

[42] "Edge Detection - MATLAB & Simulink - MathWorks United Kingdom", Mathworks.com,

Available: https://www.mathworks.com/help/images/edge-detection.html.

[43] A. Rosebrock, "Zero-parameter, automatic Canny edge detection with Python and OpenCV - PyImageSearch", PyImageSearch

Available: http://www.pyimagesearch.com/2015/04/06/zero-parameter-automatic-canny-edge-detection-with-python-and-opencv/.

[44] C. Detection, "Canny Edge Detection", Opencvexamples.blogspot.com,

Available:http://opencvexamples.blogspot.com/2013/10/void-canny-inputarray-image-outputarray.html.

[45] "eyantrainternship/eYSIP_2015_Marker_based_Robot_Localisation", GitHub,

Available:https://github.com/eyantrainternship/eYSIP_2015_Marker_based_Robot_Localisation/wiki/Line-Detection.

[46] "Implementing Hough Transform (Line Detection)", Sidekick.windforwings.com, Available: http://sidekick.windforwings.com/2012/12/implementing-houghtransform.html.

[47] "Hough Line Transform — OpenCV 2.4.13.3 documentation", Docs.opencv.org,

Available:http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html.

[48] "Hough Line Transform — OpenCV-Python Tutorials 1 documentation", Opencv-python-tutroals.readthedocs.io,

Available:http://opencvpythontutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html.

[49] "Hough Line Transform — OpenCV 2.4.13.3 documentation", Docs.opencv.org,

Available:http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html.

[50]https://www.ndeed.org/EducationResources/HighSchool/Electricity/conductorsinsulators.htm

[51] "Histograms of oriented gradients for human detection - IEEE Xplore Document", Ieeexplore.ieee.org,

Available: http://ieeexplore.ieee.org/document/1467360/.

[52] "Histogram of Oriented Gradients — skimage v0.14dev docs", Scikit-image.org, 2017. Available:http://scikitimage.org/docs/dev/auto_examples/features_detection/plot_hog.html .

[53] David G. Lowe, “Distinctive image features from scale-invariant keypoints,” International Journal of Computer Vision, 60, 2 (2004), pp. 91-110

[54] S. Mallick, "Histogram of Oriented Gradients | Learn OpenCV", Learnopencv.com, 2017.

Available: http://www.learnopencv.com/histogram-of-oriented-gradients/.

[55] http://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/

[56] http://users.ecs.soton.ac.uk/msn/book/new_demo/nonmax/

[57] https://www.vision.ee.ethz.ch/publications/papers/proceedings/eth_biwi_01126.pdf

[58] https://www.mathworks.com/matlabcentral/fileexchange/19950-rocout=roc-varargin-