TECHNISCHE UNIVERSITÄT
CHEMNITZ

Extraction of Key-Frames from an Unstable Video Feed

Master Thesis

Submitted in Fulfilment of the
Requirements for the Academic Degree
M.Sc.

Dept. of Computer Science
Chair of Computer Engineering

Submitted by: Nikhilesh Vempati
Student ID: 335043
Date: 01.03.2017

Supervising Tutors : Prof. Dr. W. Hardt
: Dr. Ariane Heller

Abstract

The APOLI project deals with Automated Power Line Inspection using Highly-automated Unmanned Aerial Systems. Beside the Real-time damage assessment by on-board high-resolution image data exploitation a post-processing of the video data is necessary.

This Master Thesis deals with the implementation of an Isolator Detector Framework and a Workflow in the Automotive Data and Time-triggered Framework(ADTF) that loads a video direct from a camera or from a storage and extracts the Key Frames which contain objects of interest. This is done by the implementation of an object detection system using C++ and the creation of ADTF Filters that perform the task of detection of the objects of interest and extract the Key Frames using a supervised learning platform.

The use case is the extraction of frames from video samples that contain Images of Isolators from Power Transmission Lines.

Keywords: Object Detection, Key-Frame Extraction, ADTF, SVM

Contents

List of Figures

List of Tables

List of Abbreviations

OOI	Object of Interest
UAV	Unmanned Aerial Vehicle
HD	High Definition
FPS	Frames per Second
SOC	System on Chip
ECR	Edge Change Ratio
SIFT	Scale Invariant Feature Transform
MIDI	Musical Instrument Digital Interface
SSD	Sum of Squared Differences
HOG	Histogram of Oriented Gradients
SUSAN	Smallest Univalue Segment Assimilating Nucleus
DOG	Difference of Gaussian
HOG	Histogram of Oriented Gradients
KKT	Karush Kuhn Tucker
VS	Microsoft Visual Studio
MLL	Machine Learning Library
DSP	Digital Signal Processor

1 Introduction

The military and defense sectors were until recently the sole users of UAVs. Over the years, drones have become lighter and more complex. The control electronics which were expensive are now being realised through less-expensive components and even indirectly with smartphones.

The spending on the research and manufacturing of drones is increasing at a rate of 15-20 % every year[11], the result of which is ever-improving hardware. Even though hardware for business drones is vital, but as in different sectors, the software package is what produces the distinction in several applications. As it becomes financially viable to customise commercial drones, the applications extend to a broad range of niche areas. The importance and application of drones are expected to rise enormously in the near future. Today drones have a significant presence in Agriculture, where they are replacing light manned aircraft for crop dusting. Drones are being used in the manufacturing sector for maintenance of equipment that are not easy for a person to manually perform. They are also used in construction for surveying the progress from different distances and angles.

Drones come in many sizes and with many variations, and the larger, more powerful drones will soon have the ability to carry larger payloads and will soon have a presence in logistics. When paired with high-resolution cameras drones have also found application in aerial surveillance, applied sciences and multimedia.[26]

With internet bringing people and businesses closer together, network infrastructure is growing in every country of the world. Drones will be able to connect to existing networks and can communicate with one-another and existing infrastructure, offering a path to an even greater range of applications.[8]

1.1 Motivation

Video based inspection techniques involve users reviewing videos for gaining vital information from the video. Today users are dealing with an ever-increasing number of videos. Manual video analysis, is a time-consuming process, even more so when the videos are relatively long. Additionally, the storage of videos over a duration of time becomes very expensive due to increasing storage hardware needs. An HD video of size 1920x1080 pixels requires 150MB of storage for every 1 minute of footage. A 10 minute video will therefore consume 1.5GB of storage. Under most circumstances, uncompressed media has prohibitively large storage and delivery requirements, but for some local playback scenarios, the quality level is important enough to not use compression.

Thus tools are required for efficient and quick video summarization. Video summarization is an important research topic which aims to create automatically a compact and representative summary of video content in terms of still images.[4]

One such technique for video summarization is Key-Frame Extraction. A Key-Frame can be generally defined as am image or frames of a video that can provide the essential information of a video. This process is usually performed by comparing the likeness of each consecutive video frame to consider whether there is a change in the scenary or not. Similar successive frames are termed as a Shot. The first of which is marked as the keyframe.

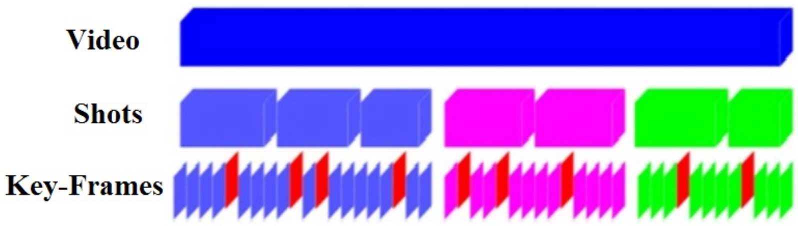


Figure 1.1: Video Summarization [17]

Cell phones and cameras have built-in software stabilization and in some more expensive capturing devices, hardware stabilization. However, stabilization is achieved by smoothening the motion between jittery frames. As, most digital cameras do not have a lot of processing power or memory, the result of such software smoothening is not always a stable video. Drone cameras face the same problem of instability as even the most experienced pilot is effected by changing wind directions, thereby almost always resulting in unstable video footage.[29]

The consequence of using such videos for inspection is that normal shot detection techniques would not be very useful, as the content of the video can change quickly between one frame to the next. Thus the need for a video summarization technique arises that is fast and in real-time is able to detect key-frames.

A Key-Frame can also be defined as the frames that contain our Object of Interest. So this task can be solved with feature/object detection techniques. Feature points are image intensities which are peculiar and robust in different views of the same object. These can be repeatedly detected in different images. Figure 1[17] represents how the storage requirements reduce significantly when using Key-frame extraction techniques.

1.2 Use Case

An Isolator has two main functions in transmission lines; it must hold the transmission cable at a fixed distance from the tower and the ground, and it has to provide insulation between the transmission cable and the ground. So a failure of an isolator can lead to electrical downtimes and in some cases severe damage to the lines around the isolator because of short circuiting.

Isolators in power transmission lines are generally more susceptible to damage than the tower or the cables. The towers are firmly planted in the ground, and the cables have slack to deal with changing wind conditions. Isolators though are suspended between two objects, so due to changing loads, changing weather and sometimes vandalism, isolators can fail.

An Isolator can fail in two possible ways; electrically or mechanically. A mechanical failure generally means that the entire electrical functionality of the isolator is lost. However, in case of electrical failure, the isolators may still provide the strength and rigidity needed for continued operation of the line.[2] Electrical failures are caused when a flash-over occurs between the two ends of the isolator. Flash overs are difficult to contain in high-voltage

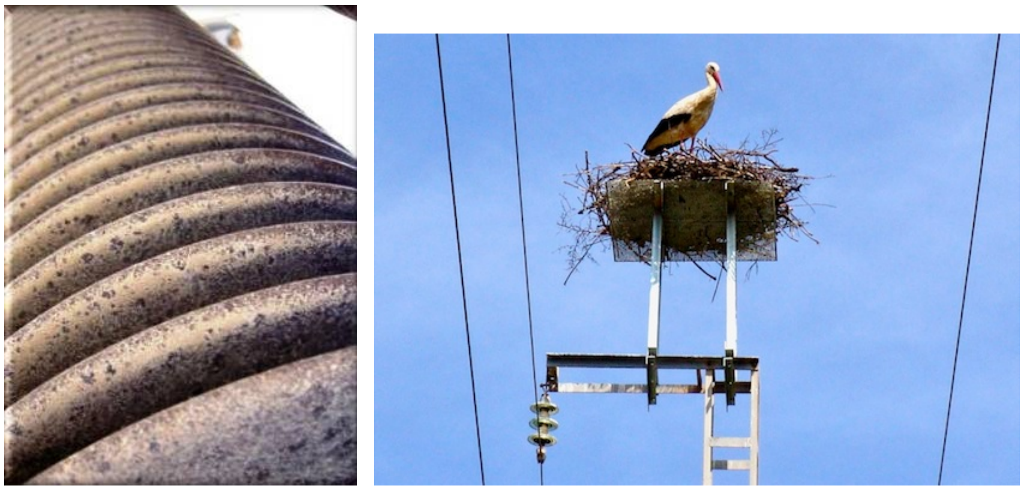

(a) Pre-deposit pollution (b) Instant pollution

Figure 1.2: Isolator pollution[23]

transmission lines because even the smallest pollution of the isolator could cause a flash-over path. Pollutant residue on the isolator together with atmospheric humidity can provide a conductive layer for leakage current.

Figure 1.2 illustrates the different types of isolator pollution; Pre-deposit pollution and Instant pollution. Pre-deposit pollution occurs over a period of time when oils, salts, dirt slowly deposit over the surface of the isolator. Instant pollution occurs due to bird excrement and flying debris.[23] So inspection and maintenance of isolators is absolutely essential for a reliable power transmission network.

Since the late nineteenth century, transmission lines and isolators have been inspected manually by workers travelling along the line, observing with binoculars or telescopes on the ground. Since electricity is not usually generated the same place that it is utilized, most high voltage transmission lines run for hundreds of kilometres, thereby not only is the process of inspection time consuming but the observer cannot get a observation in all desired angles from a point on the ground.[15] Getting a closer look makes the process more dangerous as it involves workers reaching the isolators with ladders and cables.

UAV-based automated power line inspection systems offer major advantages compared to conventional human expert-based examination. They realise automated and safe inspection of electric power distribution systems. They help greatly reduce costs due to downtimes and losses during power-down and power-up procedures. They also reduce the number of man-hours needed for the inspection process, as a drone camera is used to capture the required details, with lesser effort. In sparsely populated regions, the usage of drones will enable much faster inspection. Due to the advantages and the ease of use, several countries in the world are looking to implement drone based inspection techniques for their transmission grid.

The project Automated Power Line Inspection (APOLI) undertaken by the Professorship of Computer Engineering at Technische Universität Chemnitz aims to develop a vision-based inspection system for electric power distribution systems. The inspection shall be done with a UAV and shall focus on damage assessment of isolators, electric power poles, and transmission lines.[1]

The aerial inspection enables damage assessment of poles and lines with hard or no accessibility and from view-points different from ground. The application of high-resolution inspection sensors and the possibility of all-degree analysis provide a time- and cost-saving method which captures more damages and is more accurate/powerful in relation to ground-based visual assessment by expensive experts.

The inspection tasks are performed by a robot carrying detection instruments either along the transmission line or by recording videos of the transmission lines and reviewing them. The time to review such videos is time-consuming and brings forward a need to extract the key-frames which can reduce the human effort needed.

The system allows high-resolution/high-accuracy inspection of all parts of power distribution systems in operation. This reduces costs due to downtimes and losses during power-down and power-up procedure.

1.3 Outline

Chapter 2 describes the state of art technologies described in research publications and several internet scientific community websites. The various approaches were analysed and compared for their speed, accuracy and feasibility. **Chapter 3** is dedicated to the concept. Reasons are given, explaining the reasoning behind the choices made in this thesis. **Chapter 4** describes the Implementation process and also briefly describes the Elektronit *EB Assist Automotive Data and Time-Triggered Framework.* **Chapter 5** discusses the Isolator Detection Application and how it is to be used. It describes the dataset sizes and summarizes the training and detection procedures. **Chapter 6** discusses the data evaluation techniques and conditions and their results. **Chapter 7** finally describes the conclusions drawn during the course of this research. It also discusses the research that can be done to make progress in this field of research.

2 State of the Art

As discussed in the Motivation, key-frames help reduce the storage requirements when compared to videos. Key-frames help reduce human time and effort needed for the inspection process. This chapter discusses the state of art technologies used for video summarization, and the machine learning approaches used therein. A small comparison of these state of art technologies is discussed in the summary.

2.1 Video Summarization

Using videos for information gathering and storage has been on the rise rapidly over the last two decades. Advancements in video technology have made it possible to record bulk contents such as news, sports and documentaries by personal and professional consumers alike. Videos are being used by the scientific community today for surveillance, inspection, tracking and detection. The storage of these videos becomes a challenge when dealing with a large number of videos, which is inevitable because videos are recorded periodically over a period of time.

To solve this problem video summarization techniques play an important role as they help the user quickly navigate, retrieve and browse a long video sequence. Figure 2.1 represents a hierarchical classification of these techniques. [3] This section covers the most important of these summarization techniques shown in the hierarchy by their advantages, drawbacks and methodologies used therein. i.e.; Shot Selection, Cluster-Based Video Summarization and Feature Based Video Summarization.

2.1.1 Shot Selection

This approach is also called shot transition detection. A shot is defined as a series of frames that run for an uninterrupted period of time. A shot

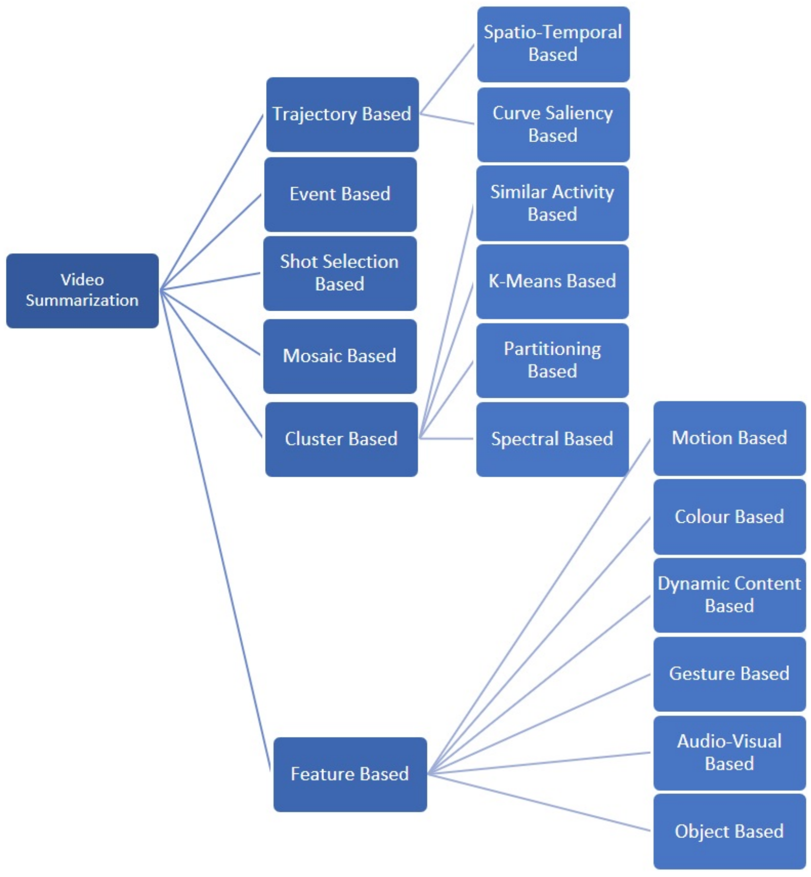


Figure 2.1: Hierarchy of Video Summarization

can be termed as either a Hard shot which are sudden and accompanied by extensive changes in the visual content, or a Soft shot which are distinguished by slow and gradual changes. The period of time is defined by the user to classify each Shot.[32] There exist several techniques to segment video into shots. The main approaches are described below-

Shot Boundary Detection Methods - These are the most primitive approaches to Shot Detection. Successive frames are taken as inputs and the intensity of pixels are calculated and compared. If the similarity is found to be more than a certain threshold, the frames are termed as a shot. The similarity between two frames can be computed by the comparison of

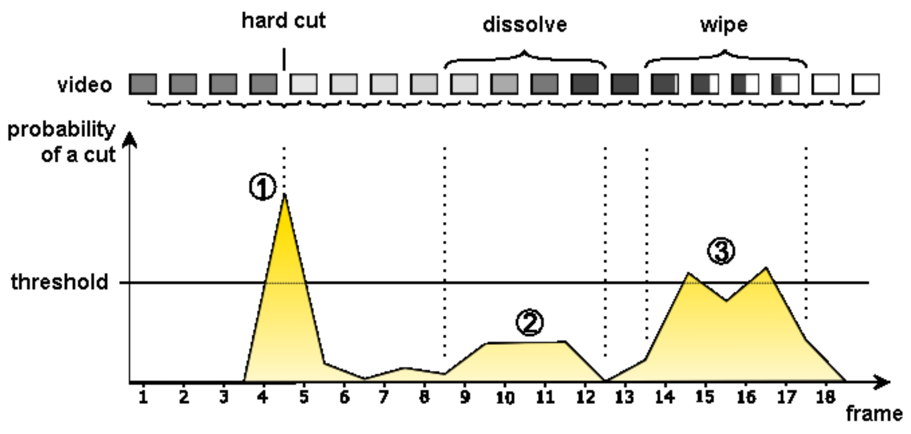


Figure 2.2: Shot Boundary Detection

the histograms of the color-space of the frames. The color histogram-based shot boundary detection algorithm is one of the most reliable methods for shot detection. This can be represented by the equation 2.1 where the intensity of each color is added over a pre-determined range. If the result of the equation is more than a threshold B, then the frames are termed as a shot. These methods are comparatively slow and setting manually threshold is not the best idea as seen in Figure 2.2.

$$\frac{1}{N}\sum_{r=0}\sum_{g=0}\sum_{b=0}(f_1(r,g,b)-f_2(r,g,b)) \tag{2.1}$$

Edge Change Ratio - The edge change ratio is calculated by the comparison of edge pixels between two successive frames.The edges are calculated by the Canny edge detector. A robust measurement is taken by disregarding edge pixels within a certain distance of an edge. (eg. within 6 pixels distance). Further robustness is added to the calculation by applying the Hausdorff distance motion compensation.[33] Once the ECR is calculated, Shots are recognised as isolated peaks as seen in Figure 2.3.

Shot Classification Algorithms - In shot classification algorithms, a linear classifier is trained using middle-level feature vectors calculated using different approaches. The size of the region from which individual features are extracted plays a great role in the performance of shot change detection. A small region tends to reduce detection in variance with re-

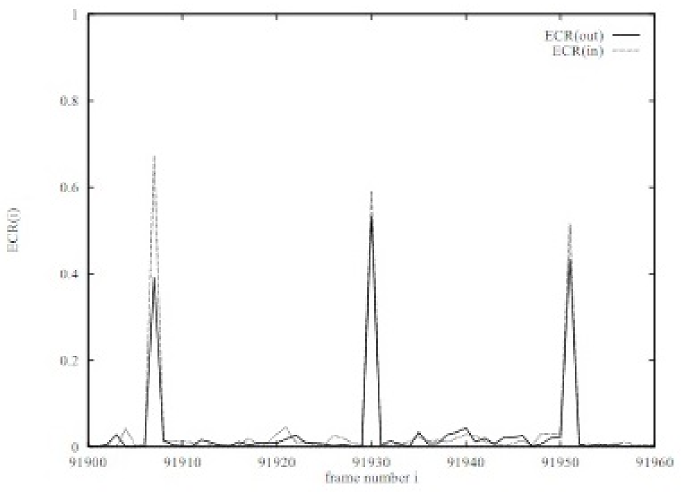


Figure 2.3: Isolated peaks representing shots[12]

spect to motion, while a large region tends to miss transitions between similar shots.[12] Feature vectors can be calculated using a single frame pixel, or from an arbitrarily shaped region which can exploit the most homogeneous regions of a frame. Due to advancements in Artificial Neural Networks over the last five years, the colour-histograms for the complete image can be calculated as a whole and fed as feature vectors for classification.

2.1.2 Video Summarization Based on Clustering

Clustering is among the most commonly used techniques for video summarization. The approach helps eliminate frames with irregular trends. Clustering also helps create a shorter summary of a video when compared to other approaches. The main methods for clustering are partitional, spectral, similar activities based and K-means clustering.

Partitional Clustering - Partitional clustering is also termed as a non-hierarchical clustering technique. It does not organise data into a nested sequence of groups, rather it works by generating a single partition of the data to represent the natural groups present in the data. This technique works by removing the redundant visual content that exists in video frames.

The whole video is grouped into clusters such that each cluster represents frames of similar video content. Starting with an initial partition, objects are moved from one cluster to the next iteratively till a local minima of the criteria function is reached. A validity analysis of these clusters is then performed in order to find the optimal number of clusters for a given video sequence. Each of these clusters contain frames of similar video content, and can be concatenated to produce a video summary.[16]

This approach is very effective for videos that contain slow moving shots, however when there is a rapid change from scene to scene, it fails to generate a good video summary.

K-Means Clustering - The K-means clustering algorithm works by placing K points in the space represented by the objects to be clustered. These points are termed as Centroids. Iteratively,the objects to be clustered as assigned to each of these centroids. Once all the objects have been assigned, the centroids are recalculated using the means of the objects for each of the k-clusters. This process is repeated till no change in mean is recorded. This process is visualised in Image 2.4 for k=2 clusters.

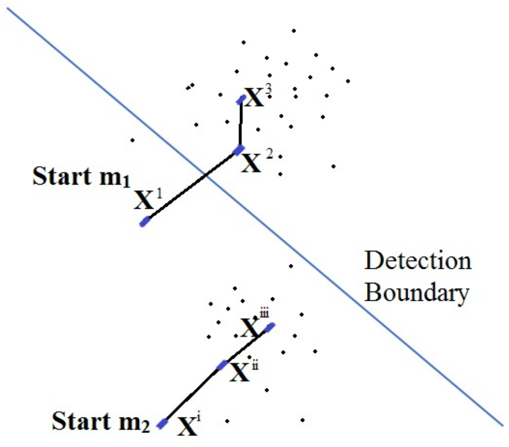


Figure 2.4: k-means Algorithm[16]

Using this algorithm, the input video is split into k clusters. The first frame of each cluster is taken as the representative for that cluster, and the algorithm is iterated till similar frames are clustered together. The comparison of frames can be done by using the histogram of colour or pixel intensity in the frame. Finally the required segments are selected and inserted into a list to generate a desired summary of the video.

Similar Activities Based Clustering - This approach was developed with the intention of packing similar activities together more efficiently for shorter video summaries. An activity is defined as a dynamic object that appears in multiple frames and defined by a sequence of object masks in those frames. These activities are further divided into Tubelets which are of a fixed length(eg. 50 frames) and can overlap with other tubelets. The creation of tubelets helps compare activities of similar length, eg; the action of kicking a ball into the goal in a football game. After the clustering of these tubelets, overlapping ones are merged together to form a longer activity.

The features used for clustering are extracted using the SIFT descriptor. For each object multiple SIFT features are calculated and compared for estimating the similarity of the objects. The motion of objects is represented by a sequence of frame by frame features with the X and Y axial displacement and the radius of the object. These features therefore represent the object as well as their motion.The similarity between various activities and the distance between objects are computed. Then a play-time is assigned to each object within each cluster, followed by a play-time to each cluster. Desired clusters are then selected for summarization. The spectral clustering approach is used for clustering of the features, therefore some of the disadvantages are inherited by this algorithm. Another drawback of using this approach is that a wrong perception about activity may lead to false summary.[25]

2.1.3 Feature Based Video Summarization

For the purpose of summarization the frames for a given video are selected by considering the information content of the frames in that shot. Every video contains features based on colour, motion, patterns or sounds. Feature based video summarization techniques are chosen over the other approaches when only a certain feature is of interest to the user. The main approaches used for feature based video summarization are motion based, colour based, gesture based or object based.

Motion Based Video Summarization - In situations where shot detection techniques fail because there are more than one key-frame making up the shot, motion based video summarization techniques are very useful. In many shots, the key frames are identified by stillness-either the camera stops on a new position or the characters hold gestures to emphasize their importance. Motion based summarization techniques use a hierarchical key-frame selection methodology in which first shots are detected, then classified into several categories. Then key-frame selection algorithms are applied to each category.

An approach described by Wolf in [31] involves an analysis of the optical flow to measure the motion within a shot and extraction of key-frames at the local minima of the motion. The research uses in particular, the Horn and Schunck's formula for calculating the optical flow. The sum of magnitudes of the components of optical flow for each pixel is calculated as a motion metric using the equation 2.2. Where X and Y represent the optical flow in the X and Y components of the optical flow at a pixel i,j in a frame t. An algorithm scans the M(t) vs t curve for every frame t within a video. Then two local maxima m1 and m2 are identified such that the value of m2 varies by a percentage N from the value of m1. The local minima of M(t) is then found in the curve to determine the key-frames.

$$M(t) = \sum_i \sum_j |X(i,j,t)| + |Y(i,j,t)| \tag{2.2}$$

Motion based approaches work well when there is medium-level motion in a video. However, these approaches fail when summarizing videos that contain huge or no motion at all.

Colour Based Key-frame Extraction - The colours within a video are often considered to be important aspects. Therefore colour based features are often used for video summarization. The simplest version of the colour based features approach, calculates the colour histogram for each frame and a probability model is used for identifying different shots within a video. The occurrence of each shot is found and integrated to give a meaningful summary for a video.

More complex algorithms involve colour histogram techniques combined with pattern repetitions to find the shots. The first frame of each shot is termed as a key-frame and it is compared to other frames by comparing their colour histograms to find the next key-frame in the shot. These algorithms are easy to implement and use, however the colour histograms used also account for noise and in videos with lots of motion, or with lots of noise, the result is not accurate.[35]

Audio Based Video Summarization - Audio based summarization techniques are very useful in today's world where online video platforms are becoming more and more popular. The consumers who want to buy a particular video or music album might want to watch the highlights first. The earliest audio based summarization techniques used only MIDI data. However, these methods cannot be applied to real-world videos where audio streams are of a higher bitrate and quality. Therefore audio based summarization techniques are combined with visual features. The first step is to achieve a synchronization between the audio and video, and based on the important audio features, the corresponding video segments are extracted to form shots. However this technique doesn't perform at all when the audio is missing from a video.[10]

The other way audio can be used is by using a speech recognition software to create a collection of all the words used in the video. Then the frequency of each word is found out to create a summary of a long video. The complete video is first distributed into similar segments and audio pause boundary detection is done through temporal analysis of the pauses between words. Now these segments are ranked based on the number of occurrences of the most dominant used words in the complete video. A summary is generated by maximizing the cumulative score of the video and audio segments. This technique is effective when there are gradual pauses in speech. However, it fails when the videos are noisy or videos where there is continuous speech.[7]

Object Based Video Summarization - Object based video summarization is most often used when the user is interested in objects within a video. The content change in a video is easy to do as they can be classified based on the appearance and disappearance of the objects of interest. These techniques are based on detecting of the objects of interest using feature detection algorithms and then applying a temporal technique to extract key-frames.

More advanced methods involve detecting an object of interest by first training an object detector by annotating the frames with the object of interest and eliminating frames with no detections. The machine can be made into a self-improving partially supervised machine learning algorithms where frames with high detection scores are pseudo-labelled as positive and the training is repeated a second time.[19] Object based approaches are commonly used in surveillance and inspection systems. The selection of the feature detection and classification algorithms must be chosen for accuracy as well as speed.

2.2 Feature Description

Feature description, detection and matching are fundamental segments of different computer vision applications, and they have undergone extensive research in recent decades. There exists no exhaustive or correct meaning of what constitutes a feature, and a correct definition regularly relies on upon the kind of utilization. Given that, a feature is characterized as an "interesting" portion of a picture, and they are utilized as a beginning stage for many computer vision based calculations. Since features are utilized as the beginning stage and fundamental primitives for resulting algorithms, the overall feature will frequently just be in the same class as its feature detector. Subsequently, the alluring property for a feature detector is repeatability: whether a similar feature will be recognized in at least two distinct pictures of a similar scene. Feature detection is a low-level picture preparing operation. It is typically executed as the primary operation on a picture, and inspects each pixel to check whether there is a feature present at that pixel. In the event that this is a piece of a larger algorithm, then the calculation will ordinarily just inspect the picture in the area of the features. As an implicit pre-requisite to feature detection, the information

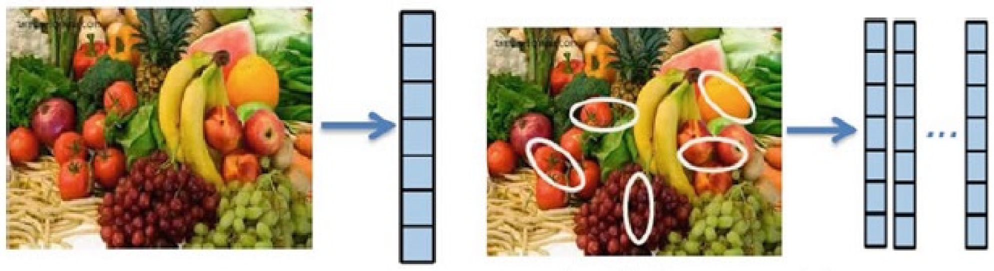

(a) Global Descriptors (b) Local Descriptors

Figure 2.5: Descriptors[14]

in an image is generally smoothed by a Gaussian Kernel in a scale-space representation and several feature images are computed, often expressed in terms of local image derivatives operations. Once in a while, when feature detection is computationally costly and there are time limitations, a higher level algorithm might be utilized to manage the feature detections, such that only certain parts of the picture are represented as features.

The fundamental thought is to first distinguish areas of interest (keypoints) that are covariant to a class of transformations. At that point, for each detected areas, an invariant component vector descriptor for picture information around the detected keypoint is constructed. Feature descriptors obtained from the picture can be based on second-order statistics, parametric models, coefficients gotten from an image transformation, or even a blend of these measures. Two sorts of feature descriptors can be extracted from an image representation, global features and local features. Global features such as colour and surface patterns, depict a picture as a whole and can be translated as a specific property of the picture including all pixels. While, local features plan to identify keypoints or regions of interest in a picture and depict them. For example, if a local feature calculation recognizes n keypoints in the picture, there are n vectors depicting each one's shape, pattern or orientation. A depiction of global and local descriptors can be seen in Figure 2.5. The utilization of global colour and texture features are demonstrated to be very effective for finding comparative pictures in a database, while the local structure oriented features are viewed as sufficient for object classification or finding similar objects in another image dataset.[14] The following subsections cover the feature descriptors most frequently used for computer vision applications today.

2.2.1 Harris Detector

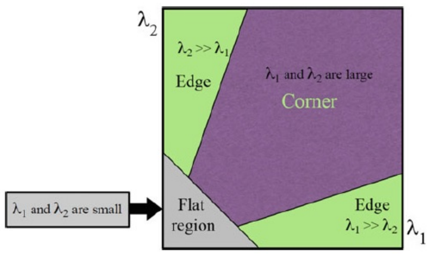


Figure 2.6: Harris Detector Classification[14]

Prior to the Harris Detector, corner detection algorithms defined a corner as a point with low self similarity. The Morevec algorithm[22] tested each pixel in an image by considering how similar a patch centred on a pixel was to nearby overlapping patches. The sum of squared differences between the pixels of each patch gave a score of inverse similarity. If the pixel was within a region of uniform intensity, then the nearby patches appeared similar. If the pixel was on an edge, then nearby patches which are at an perpendicular to the edge looked different. The corner was identified by the smallest SSD between the patch and its immediate horizontal, vertical and diagonal patches. However Morevec pointed out that, one of the main problems with this operator is that it is not isotropic: if an edge is present that is not in the direction of the neighbours (horizontal, vertical, or diagonal), then the smallest SSD will be large and the edge will be incorrectly chosen as an interest point.

Harris and Stephens researched a joined corner and edge detector to address the problems of the Morevec's descriptor. Acquiring the variation of the autocorrelation over every single possible orientation, results in a more rigid detector in terms of detection and repeatability. The 2x2 symmetric auto-relationship grid utilized for identifying features and depicting their nearby structures is represented by equation 2.3

$$M(x,y) = \sum_{u,v} w(u,v) * \begin{bmatrix} I_x^2(x,y) & I_x I_y(x,y) \\ I_x I_y(x,y) & I_y^2(x,y) \end{bmatrix} \tag{2.3}$$

where I_x and I_y are local image derivatives in the x and y directions respectively, and w(u, v) denotes a weighting window over the area (u, v). For finding interest points, the eigenvalues of the matrix M are computed for each pixel. If both eigenvalues are large, this indicates existence of the corner at that location. Figure 2.6 illustrates how the corners and edges are classified using the eigenvalues of the matrix M. [13]

2.2.2 SUSAN Detector

The SUSAN detector which is an abbreviation for Smallest Univalue Segment Assimilating Nucleus was developed by Smith and Brady which instead of using image derivatives to compute corners uses a circular mask of a fixed radius to every pixel in an image. The centre pixel which is termed as the nucleus is compared to all other pixels under the mask to check if they have similar intensity values. Pixels which have the same intensity are grouped together to form a Univalued Segment Assimilating Nucleus(USAN). The regions where the number of pixels in the USAN reaches a local minimum and below a specific threshold value T are categorized as corners.[28] The comparison function to find the similarities in the pixels is represented by equation 2.4.

$$C(r, r_0) = \begin{cases} 1 & \text{if } |I(r) - I(r_0)| \leq T \\ 0 & \text{if } |I(r) - I(r_0)| > T \end{cases} \tag{2.4}$$

The size of the USAN region is found by equation 2.5.

$$n(r_0) = \sum_{c(r(0))} c(r, r_0) \tag{2.5}$$

where r_0 is the nucleus's coordinates and r is the coordinates of any other points within the mask area. The performance of SUSAN corner detector mainly depends on the similar comparison function C(r, r0), and not the value of the threshold set by the user. The algorithm produces a constant response to edge detection unlike other algorithms such as Canny which produce multiple responses at almost all the edges. Since no derivative filters are used, the computations are fast. It is invariant to translation and rotational changes, however it fails for changing scales.

2.2.3 FAST Detector

The Features from Accelerated Segment Test or FAST Algorithm was developed by Rosten and Drummond to enable real-time detection by combining an edge and corner detection methods. The algorithm begins by detecting candidate points by applying a segment test to every image pixel by considering a circle of 16 pixels around the corner candidate pixel as a base of computation. If a set of n-contiguous pixels in an area termed, the

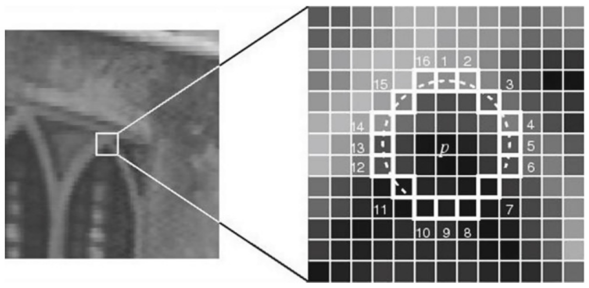


Figure 2.7: Harris Detector Classification[14]

Bresenham circle with a radius of r are all brighter than the intensity of candidate pixel denoted by I_p, added to the threshold value t, $I_p + t$, or all darker than the intensity of candidate pixel minus the threshold value I_p - t, then p is classified as a corner. A quick test can be used to exclude a very large number of non-corner points, the test examines only four particular pixels, such as 1, 5, 9, 13. A corner is found to exist if three of these test pixels are brighter than I_p + t or darker than I_p - t and the rest of the pixels are then examined for a final conclusion. Figure 2.7 describes the process where the highlighted squares are the pixels used in the corner detection process. The pixel at p is the centre of the candidate corner. The arc is indicated by the dashed line passing through 12 contiguous pixels which are brighter than p by a threshold. The best results were achieved at r=3 and n=9.[14]

For fast matching of features it is important to try and reduce the costs associated to processing referred to as $(O(N^2))$, and F_I represents the feature point that must be calculated by minimizing the sum of square differences between the two of the energy into the first few elements. This process allows for a faster rejection of potential feature matches and allowing for a lower $(O(N^2))$.[27]

The FAST corner detector is very suitable for real-time video processing applications because of its high-speed performance. However, it is not invariant to scale changes and not robust to noise as it depends on a threshold, that needs to be set non-trivially.

2.2.4 Scale Invariant Feature Transform

The Scale Invariant Feature Transform or SIFT introduced by Lowe[20] detects a number of interest points using a difference of Gaussian operator.(DOG) The local extremes of the DOG function are selected as points at which the feature vectors are extracted. Over a number of scales and over the region surrounding the point of interest, the local orientation of the image is estimated using the local image properties to provide invariance against rotation.

For each detected point, a descriptor is computed based on local image information at the characteristic scale. The SIFT descriptor builds a histogram of gradient orientations of sample points in a region around the keypoint, finds the highest orientation value and any other values that are within 80 percent of the highest, and uses the orientations as the dominant orientation of the keypoint. A region of 16x16 pixels around each keypoint is first selected, and then the image gradients and magnitudes are sampled to select the level of Gaussian blur for the image. Then, a set of orientation histograms which contain the samples from a 4x4 sub-region of the original neighbourhood region with eight orientation bins are created. A Gaussian

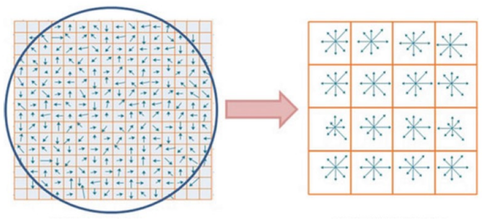


Figure 2.8: Calculation of SIFT Features[14]

weighting function with spread of blobs σ, equal to half the region size is used to assign weight to the magnitude of each sample point and gives higher weights to gradients closer to the center of the region, which are less affected by positional changes. A vector is formed containing the values of all the orientation histograms. Since there are 4x4 histograms with 8 bins, the feature vector has 128 elements for each keypoint. This process of computing orientation, binning and then weighting by a Gaussian function is illustrated in Figure 2.8. To compensate for changes in illumination the feature vector is set to a threshold of 0.2 and then again normalized.

The standard SIFT descriptor is designed carefully to avoid problems due to boundary effects, orientation and scale. It is compact, expressing the patch of pixels using a 128 element vector, and is invariant to slight deformations such as those caused by change in perspective. The SIFT vector is however complicated and of high dimensionality, thereby resulting in larger computation times.

2.2.5 Speeded Up Robust Features

The Speeded-Up Robust Features (SURF) detector-descriptor algorithm developed by Bay et al[5] was designed as an efficient alternative to SIFT features. The process does away with the Gaussian derivatives and instead

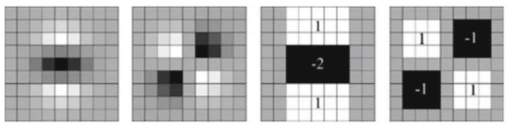


Figure 2.9: Gaussian second order derivatives in the y and xy directions and their approximations in the same directions[14]

uses 2D filters in which a scale invariant blob detector based on the determinant of a Hessian matrix for both scale selection and locations. Its basic idea is to approximate second order Gaussian derivatives in an efficient way with the help of integral images using a set of box filters.

In Figure 2.9 the 9x9 box filters shown are approximations of a Gaussian with σ=1.2 and represent the lowest scale for the calculation of the blob response maps. The approximations are denoted by D_{xx}, D_{yy} and D_{zz}. The approximated determinant of Hessian can be expressed as shown in equation 2.6.

$$det(H_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2 \tag{2.6}$$

where w is a relative weight for the filter response and it balances the expression for the Hessians determinant. The blob response in the image is represented by the approximated determinant of the Hessian. These responses are stored in a blob response map, and local maxima are detected and refined using quadratic interpolation. A non-maximum suppression in a 333 neighbourhood is performed to obtain stable interest points and the scale of values.

The SURF descriptor begins with a square region centred around the detected interest point and oriented along its main orientation. The size of this window is 20s, where s is the scale at which the interest point is detected. The interest region is divided into even smaller 4x4 subregions and for each subregion the Haar wavelet responses in the x and y directions are computed with 55 sampled points. The responses are then weighted with a Gaussian window centred at the interest point to increase the robustness against geometric deformities and localization errors. A feature vector v with wavelet responses $\mathrm{d_x}$ and $\mathrm{d_y}$ are summed up each sub-region, as represented in equation 2.7.

$$v = (\sum d_x, \sum |d_x|, \sum d_y, \sum |d_y|) \tag{2.7}$$

A feature vector of length 64 is calculated when equation 2.7 is calculated for all the 4x4 subregions. Normalization is performed to reduce the effects of illumination changes. The SURF descriptor is much faster than the SIFT descriptor because it uses a smaller feature vector (64 vs 128). The problem with SURF is with rotational invariance for 2D, 3D objects as it does not work when the object is rotated a lot.

2.2.6 Local Binary Patterns

Local Binary Patterns feature the spatial structure of a texture and it is invariant to monotonic transformations of features detection, description and matching the grey-levels. An ordering relationship is encoded by comparing neighbour pixels with the central pixel. It generates an ordered feature for each pixel by comparing the intensities of a set of two. The feature responses are marked 1 when the intensity of the neighbour exceeds the intensity of the central pixel or the feature is marked 0 in the other case. The co-occurrence of the comparison are saved as a binary string. The weights from a geometric sequence with a common ratio of 2 are assigned to the bits according to their indices in the strings. The binary string with its weighted bits is then transformed by into a decimal valued index. The descriptor describes the result over the neighbourhood as a binary pattern.

$$S(g_p - g_c) = \begin{cases} 1 & \text{if } g_p \geq g_c \\ 0 & \text{if } g_p < g_c \end{cases} \tag{2.8}$$

For a pixel c, with g(c) as its intensity is labelled as equation 2.8 where pixels p belong to a 3x3 neighbourhood with grey levels $\mathrm{g_p}$(p = 0 1 2 ..

7) are summed to corresponding threshold values S(g_p - g_c) weighted to a binomial factor of 2^k. This is termed as the local binary pattern and is shown in equation 2.9.

$$LBP = \sum_{k=0}^{7} S(g_p - g_c).2^k \tag{2.9}$$

The label is computed for each pixel and the labels are sorted into a 256bin Histogram which gives us a feature detector for the texture. In general usage 2N distinct values for the binary pattern by interpreting the differences in a neighbourhood as an N-bit binary number. The binary patterns are classified into uniform patterns as they contain at most two bitwise transitions and non-uniform patterns are the ones that contain more than two bitwise transitions.

Local binary patterns as a concept proved a success in texture description while being robust to illumination changes and being computationally efficient. However as LBP is a series of patterns, rather than a numerical feature therefore preventing the combination of LBP features with other discriminative ones.[24]

2.2.7 Gradient Location-Orientation Histogram

The GLOH feature descriptor was developed by Mikolajczyk and Schmid as an extension to the SIFT descriptor. A log-polar locator grid is used with a principle component analysis to reduce the size of the descriptor. GLOH uses location bins in its features and these bins are calculated by

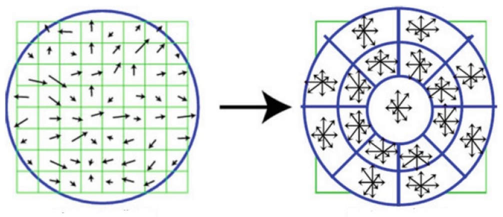


Figure 2.10: log-polar bins in a feature[14]

finding the number of radial bins and the number of angular bins.[21] So if the number of bins in the radial direction are 3 and number of bins in the angular direction are 8 will have a total of 17 bins as illustrated in Figure 2.10. The gradient orientations are quantized in 16 bins , therefore giving us a 272bin histogram. The size of this descriptor is further reduced by performing a PCA with a covariance matrix estimated on a target images. The highest 128 eigenvalues are used for description. The GLOH descriptor performs significantly better with more accuracy than SIFT however it has a computational overhead compared to SIFT descriptors.

2.3 Machine Learning

Features that are described from a sample image using a description method are compared to one another to give the prediction and therefore machine learning is used as a method to devise complex models and algorithms that can make predictions with lower computation costs and the ability to classify data not previously sampled. The main approaches in machine learning are Supervised learning where the machine is fed with input and output data and the goal is to learn a general rule that maps inputs to outputs; Unsupervised Learning where no labels are given to the training data and the machine forms its own pattern through the input data; Reinforced Learning where the computer makes decisions based on reaching a certain goal while receiving feedback on the accuracy of its predictions.

2.3.1 Linear Classification

Linear classifiers are used to classify an object's features and then identify whether an object belongs to a certain class or not. An object's characteristics are loaded as features to the machine. For a two class classification problem the inputs are mapped to a higher dimensional feature plane and then learning a feature plane to successfully separate the dataset. Points on one side of the hyperplane are termed 0 and points on the other side are termed 1. This can be seen in Figure 2.11 in which the red line and the blue line are possible hyperplanes. The green line represents a bad classifier as it fails to classify the problem.

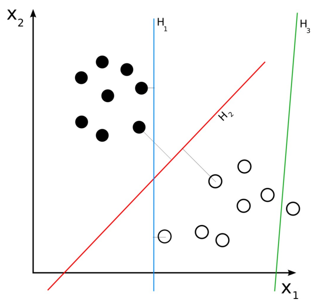


Figure 2.11: Binary Classification

Neural Networks - The training of a neural networks signifies the selecting a model from a set of allowed models that minimize the cost criteria. Multi Layer Perceptrons are feed forward neural networks which are typically composed of several layers of nodes with unidirectional connections and is trained using a back propagation with the gradient descent method. The features from the input data are featured onto the perceptrons and the machine generates an output which is a function of a weight vector and the inputs.

The weight vector is modified iteratively by minimising the error function shown in equation 2.10

$$E(w) = \frac{1}{2}\sum_{i=1}^{N} ||y(x_i, w) - d_i^2||^2 \tag{2.10}$$

where error is defined on the learning set (x_i, d_i) and N is the size of the input vector.

$$w(k+1) = w(k) + \gamma\rho(k) \tag{2.11}$$

The weight function is represented by the equation 2.11 where w is the weight, k is the step number and $\gamma$is the learning rate. ρ(k) represents the direction of minimization for a step. The problem with neural networks is that a large training data is required for convergence when compared to other classifiers.[34]

Linear Discriminant Analysis - The LDA or Fisher's linear discriminant works by finding a linear combination of features that can be used to separate two different classes of data. It is also a way to reduce dimensionality while preserving the class discrimination. The process works by projecting the input data to a straight line. This method is often used to reduce the dimensional requirements during mapping.[30]

Support Vector Machines - In machine learning classifying data is common task. Support Vector Machines (SVM) are supervised learning models associate with learning algorithms that analyze data used for classification and regression analysis in machine learning. A support Vector Machine models the situation by creating a feature space, which is a finite dimensional vector space. Each dimension represents a feature of a object. The aim of SVM is to train a model that assign a new unseen example in to a category. This can be achieved by creating a linear partition of the feature space in to two categories.

The general support vector machine framework includes components such as regularized linear learning models, convex duality and the associated dual-kernel representation, theoretical bounds and sparseness of the dual-kernel representation. The benefit of SVM comes from the fact that they are not restricted to being linear classifiers. Non-linear decision boundaries can be achieved utilizing the kernel trick. Formally, in mathematical language, SVMs construct linear separating hyperplanes in high-dimensional vector spaces.

3 Concept

The problem of Isolator detection from a drone video footage faces the problem of the video footage being unstable, moreover the background in each frame of the video can change based on the relative position of the drone to the horizon. The object itself in the video faces translation, rotation and variable scales based on the dataset procured from drone footage. Another big problems facing this task is that the isolators that need to be detected can be of different lengths or radii based on the purpose and insulation ratings. Here we describe the histogram of oriented gradients, its extension the deformable parts model and the usage of SVM max-margin object detection.

3.1 Histogram of Oriented Gradients

The HOG feature descriptor introduced by Dalal and Triggs has features which are based on a combination of SIFT and GLOH. The basic idea behind this approach is that objects characteristics can be represented by the distribution of local intensity gradients or edge directions.

The goal is to detect objects under varying illumination and varying scales so the approach describes how this goal can be met. HOG descriptors are multi-scale sliding window detectors. The process of extraction of these HOG features is described below.

3.1.1 Sliding Window Detection

The process begins by dividing the image window into several smaller regions of equal size called cells. The cells are shaped as rectangles, however even radial shaped cells are also possible.

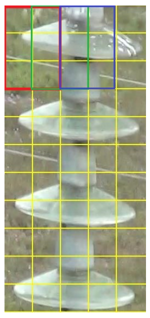


Figure 3.1: Sliding Window on an Isolator

A group of cells are collectively called a block and they can be chosen to fit the size of the training window evenly. A matrix of blocks usually constitutes the detection window. This detection window is run along the actual image and the features are extracted. Figure 3.1 shows how the detection window moves across the image. (1.red, 2.green, 3.blue).

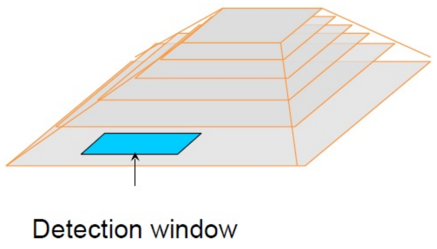


Figure 3.2: Scale-space Pyramid[6]

To make the algorithm invariant to illumination changes the image is normalized for gamma and colour. To make the algorithm invariant to scale the image is scaled down to form a scale-space pyramid as shown in Figure 3.2. To each scaled down image the the sliding window is once again applied therefore a dense cluster of features are extracted at varying scales of the same object.

3.1.2 Gradient Computation

- The most common method to compute gradients is by applying a 1-dimensional centred, point discrete derivative mask in the x and y directions. The filter kernels are represented in equation 3.1.

$$M = [-1, 0, 1] and [-1, 0, 1]^T \tag{3.1}$$

Different kernel filters can be used such as the Sobel filter however were proven to not have a considerable positive effect for the added computation required. Upon applying the filter a list of orientations is obtains for each cell which are cast on an orientation based histogram.

3.1.3 Orientation Binning

The pixels within each cell cast a weighted vote for a histogram after the gradient computation. The histogram channels are spread evenly over 0 to 180° for signed gradients or over 0 to 360° for unsigned orientations. After attempts at various binning sizes, the number 9 performed better than any other bin size.

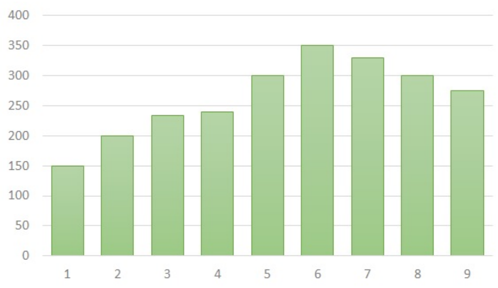


Figure 3.3: Orientation Bins

So the channel is split further into 9 bins where the bin size is 20^o for signed gradients. The vote weight cast by the cell is determined by the gradient magnitude or a square of the gradient magnitude or by other normalization methods. The histogram of 9 bins is shown for illustration purposes in Figure 3.3.

3.1.4 Contrast Normalization

The image made invariant to contrast and robust to illumination changes by tiling the detection window by a dense grid of HOG descriptors. The

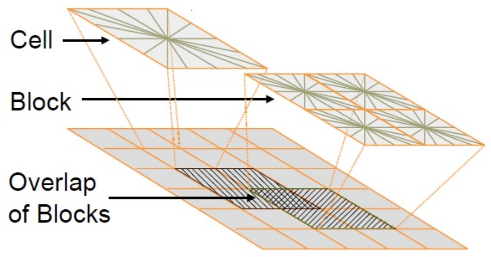


Figure 3.4: Detection Window Overlapping[6]

window is moved accross the image such that the blocks are selected by the training window with a 50 percent overlap for each measurement. This way each cell can contribute twice to the overall descriptor. This process is illustrated in Figure 3.4. Therefore for each image the window scans, it scans also the five downscaled images of the pyramid with the contrast normalization technique.

The orientations for each cell within a block are voted into a histogram of orientation gradients and therefore a denser feature scan is performed. At this step the Histogram of Oriented gradients is very similar to the SIFT descriptors, however SIFT descriptors are usually computed at sparse, scale-invariant key image points and are rotated to align orientation.

3.1.5 Block Normalization

Block normalization must be performed so that the range of operation is known. The speed of calculation can therefore be much faster by knowing the range and scale of calculations. The Normalization equations shown in 3.2(L2-norm), 3.3(L1-sqrt) can be used with near similar performance where v_k is the normalized feature vector at k=1,2.

$$L2 - norm : f = \frac{v}{\sqrt{||v||_2^2 + e^2}} \tag{3.2}$$

$$L1 - sqrt : f = \sqrt{\frac{v}{||v||_1 + e}} \tag{3.3}$$

3.2 Deformable Part Based Model

The deformable part based model introduced by Felzenszwalb et al.[9] is seen as an extension to the HOG Features by using a Principle Component Analysis(PCA) applied extracts of the HOG features. The resultant features of 13-dimensions performed nearly on-par with the original 36-dimensional feature vector. This dimensional reduction leads to more than 60 percent savings in the original computing costs.

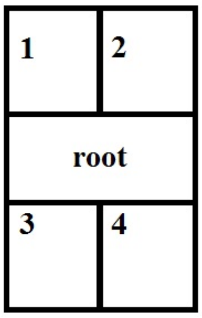


Figure 3.5: Star-structured Parts[9]

The DPM starts by defining a set of star-structured parts which consists of a root filter that approximately covers an entire object and and n part higher resolution filters of higher resolution that cover smaller parts of the object as illustrated in Figure 3.5 with 4 parts and a root. If a pyramid of scaled space of levels λ= 5 is considered then the root filter is placed at a location that matches a detection window whose pixels contribute to the part of the feature map covered by the filter. The part filters are placed

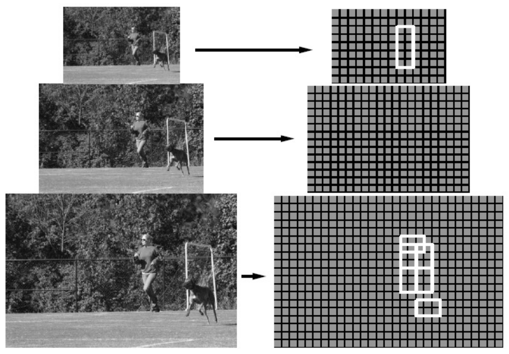


Figure 3.6: Deformable Part Model[9]

at λlevels from the root filter, thereby scanning the feature pyramid twice at twice the resolution of the features in the root filter. The placement of the part based model in a space-scale pyramid is illustrated in Figure 3.6.

The part filters are high resolution features because it is essential for obtaining high recognition performance. The part filters therefore are used to capture the finer resolution features that are localized features that are more accurate than the ones in the root filter. Therefore in the application of an isolator detector, the root filter detects the outline of the isolator and the deformable parts check for the repeated ridge based patterns accross the isolator at varying scales. Figure 3.7 shows how the parts work for two scales of the same object.

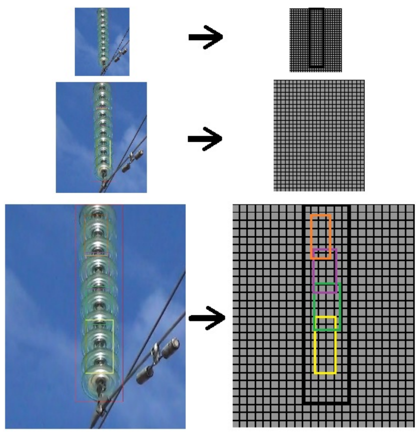


Figure 3.7: DPM over Isolator Features[9]

A model of an object with n parts is represented in equation 3.4 by an

$$(n+2) - tuple = (F_0, P_1..P_n, b) \tag{3.4}$$

Where F_0 is the root filter and P_i refers to the part filter number. The term b is a real valued bias term. Each part model is therefore represented as shown in equation 3.5

$$(F_i, v_i, d_i) \tag{3.5}$$

where F_i is a filter for the i^{th} part, v_i is a two-dimensional vector which specifies an anchor position for the part i relative to the root position, and d_i is a four dimensional vector specifying the coefficients of a quadratic function that defines the deformation cost for each possible placement of a part relative to the anchor position.

We describe the locations of the filters in a feature pyramid Z to specify a hypothesis for an object class. The parts are represented by (p_0, p_1, p_2,. p_n). Part p_i is represented as in equation 3.6 defines the location of each filter in the model in a feature pyramid Z.[9]

$$p_i = (x_i, y_i, l_i) \tag{3.6}$$

The level of each part is set as in equation 3.7, l is selected such that the feature map at that level is computed at twice the resolution of the root level.

$$l_i = l_0 - \lambda \text{ for } i > 0 \tag{3.7}$$

The score of a hypothesis is given by the scores of each filter at their respective locations (the data term) minus the deformation cost as shown in equation 3.8.

$$score(p_0, ..., p_n) = \sum_{i=0}^{n} F_i'.\phi(H, p_i) - \sum_{i=1}^{n} d_i.\phi_d(dx_i, dy_i) + b \tag{3.8}$$

where the displacement of the i$^{\text{th}}$ part relative to its anchor position is shown in equation 3.9

$$(dx_i, dy_i) = (x_i, y_i) - (2(x_0, y_0) + v_i) \tag{3.9}$$

and the deformation features are represented as in equation 3.10.

$$\phi_d(d_x, d_y) = (d_x, d_y, dx^2, dy^2) \tag{3.10}$$

3.2.1 Detection

The HOG detector computes an image pyramid with HOG features, score of each window, and a learned linear classifer. The trained classifer is run across all possible positions in the image pyramid. This sometimes results in multiple detections and therefore the others are suppressed by a greedy algorithm such as non-maxima suppression. The matching is done by a learning algorithm such as SVM.[6]

For the deformable part model the detection phase is similar till the pyramid of hog features is obtained. According to the best possible scores obtained by placing the parts p_0 to p_n. High scoring root locations define detections and the location of the parts which score highly form the full hypothesis.

To find the best location of the parts as a function of the root location dynamic programming and generalized distance transforms(min-convulations) were used. The resulting method is very efficient, taking O(nk) time once filter responses are computed, where n is the number of parts in the model and k is the total number of locations in the feature pyramid.

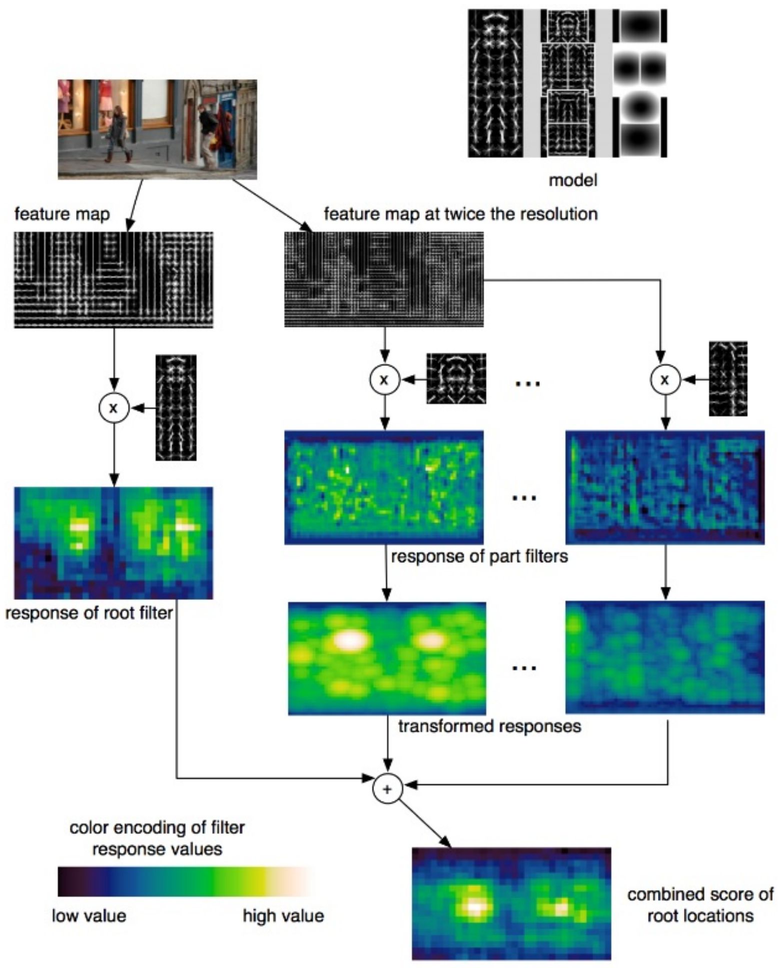


Figure 3.8: Detection Process[9]

Let $R_{i,l}$ equation 3.11 represent an array storing the response for the i-th model filter in the l-th level in the feature pyramid.

$$R_{i,l}(x,y) = F_i'.\phi(H,(x,y,l)) \tag{3.11}$$

$R_{i,l}$ is a cross relation between the feature F_i and level l of the feature pyramid. After computing the response we transform the responses of the part filters to allow for spatial uncertainty.

$$D_{i,l}(x,y) = max_{dx,dy}(R_{i,l}(x+dx,y+dy) - d_i.\phi_d(dx,dy)) \tag{3.12}$$

This transformation spreads high filer scores to nearby location taking into account the deformation costs. The value $\mathrm{D}_{\mathrm{i,l}}$ shown in equation 3.12 is the maximum contribution of the i-th part makes to the score of the root location that places the anchor of this part in level l. The transformed array $\mathrm{D}_{\mathrm{i,l}}$ can be computed in linear time from the response array $\mathrm{R}_{\mathrm{i,l}}$ in linear time using the generalized distance transform algorithms. The overall root scores at each level can be expressed by the sum of the root filter response at that level, plus shifted versions of transformed and subsampled part responses. This process is illustrated Figure 3.8. After the root location (x_0, y_0, y_0) we can find the corresponding part locations by selecting the optimal displacement in $\mathrm{P}_{\mathrm{i,l}_0}$ equation 3.13

$$P_{i,l_0-\lambda}(2(x_0,y_0)+v_i) \tag{3.13}$$

3.3 Max-margin Object Detection

Object detection and positioning in an image is an important aspect of computer vision. The basic training procedure has remained almost the same. Normally a set of positive and negative image windows comprise a training set. Then a binary classifier is trained on these images. The classifier was fed more data through iterations for reaching the optimal detection. In the following training methods, the trained classifier is tested on images containing no targets of interest and false alarm windows are extracted and fed as hard negatives into the training set. These approaches are not efficient as training is performed only on a subset of image windows. Additionally windows overlapping partially an object cause frequent false alarms.[18]

If we consider a rectangle area r, we can use it to denote a rectangular area in the image. R is the total number of rectangular areas scanned by the window in an image. We must therefore find the set of sliding window positions which have the largest scores but simultaneously do not overlap. Normally this is accomplished with a greedy peak sorting algorithm. An ideal learning algorithm would find the window scoring function which jointly minimizes the number of false alarms in the greedy method in Algorithm 1.

Algorithm 1 Object Detection[18]

```
1: procedure IMAGE X(window scoring function f())
2:     $D{:=} \leftarrow$ all rectangles r$\epsilon R$, such that $f(x,r) > 0$
3:     Sort $D$ such that $D_1 \geq D_2 \geq D_3 \geq ...$
4:     $y^* := \{\}$
5:     for i=1 to D do
6:         if $D_i$ does not overlap any rectangle in $y^*$ then
7:         $y^* := y^* \cup \{D_i\}$
8:         end if
9:     Return: $y^*$, The detected object positions
```

For a set of images and associated labels the machine is trained till the machine is able to predict 100 percent of the training labels. So we have a convex optimization problem as shown in equation 3.14

$$\min_{w} \frac{1}{2}||w||^2 \ s.t. \ F(x_i, y_i) \geq \min_{y \epsilon Y}[F(x_i, y) + \Delta(y, y_i)] \forall i \tag{3.14}$$

where $\Delta(y, y_i)$ denotes the loss for predicting a labelling of y when y_i is the real label of y. This function is represented in equation 3.15.

$$\Delta(y, y_i) = L_{miss}. \textit{ (No. of missed detections) } + L_{fa}. \textit{ (No. of false alarms)} \tag{3.15}$$

where L_{miss} and L_{fa} control the formula to achieve high recall and high precision respectively.

Equation 3.14 which is a hard margin classifier is modified to a soft margin classifier in equation 3.16 so we can solve the problem of having non overlapping rectangles.

$$\begin{aligned} \min_{w,\xi} \frac{1}{2}||w||^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i \\ s.t. F(x_i, y_i) \geq \max_{y \epsilon Y}[F(Z_i, y), \Delta(y, y_i)] - \xi_i \ \forall \ i \\ \xi_i \geq 0 \ \forall \ i \end{aligned} \tag{3.16}$$

The max-margin hypothesis is illustrated in Figure 3.9. In the max-margin equation above, C is the usual support vector machine parameter and it controls the trade off between trying to fit the training data and obtaining a large margin. ξ_i represents the upper bound on the loss incurred by the training example (x_i, y_i). A convex upper bound of the average loss per training image is shown in equation 3.17.

$$\frac{C}{n}\sum_{i=1}^{n}\Delta(argmax_{y\epsilon Y}F(x_i,y,y_i)) \tag{3.17}$$

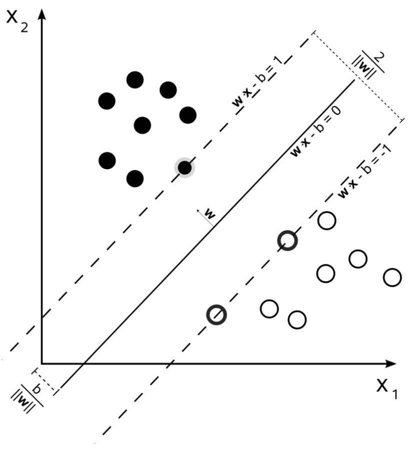


Figure 3.9: Max Margin Hyperplane[18]

This leads to the conclusion that if ξ_i in Equation 3.16 is minimized to 0, then the detector will produce a positive output for a training example.

The problem of finding the MMOD optimization by using the cutting plane method is shown in Figure 3.10. We know that a convex function will be bounded by tangent planes. The algorithm finds the minimum tangent by finding the intersection of these tangents and iteratively. The green line in Figure 3.10 represents the new tangent found from the original two tangents. The problem of max margin object detected can be represented by the equivalent unconstrained problem shown in Equation 3.18

$$\min_{w} J(w) = \frac{1}{2}||w||^2 + R_{emp}(w) \tag{3.18}$$

where $R_{emp}(w)$ is a convex function of w is represented by Equation 3.19

$$\frac{C}{n}\sum_{i=1}^{n}\max_{y\epsilon Y}[F(x_i,y)+\Delta(y,y_i)-F(x_i,y_i)] \tag{3.19}$$

If we consider δR_{emp} denote the subgradient of R_{emp} at a point w_t. The

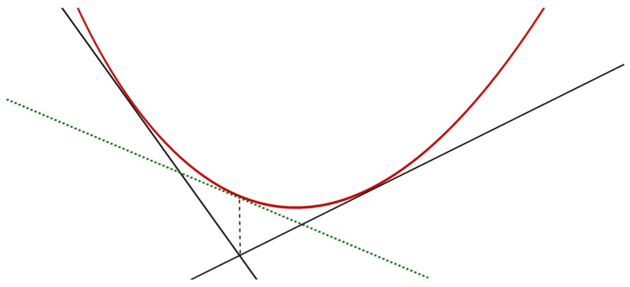


Figure 3.10: Cutting Plane Method[18]

tangent plane to R_{emp} at w_t is given by Equation 3.20 where a belongs to δR_{emp} (w_t)

$$\langle w, a\rangle + b \tag{3.20}$$

The bias b can be represented as shown in Equation 3.21

$$b = R_{emp}(w_t) - \langle w_t, a\rangle \tag{3.21}$$

With this information a lower bounding approximation is given in equation 3.22 where P is the set of cutting planes.

$$\frac{1}{2}||w||^2 + R_{emp} \geq \frac{1}{2}||w||^2 + max_{(a,b)\epsilon P}[\langle w, a\rangle + b] \tag{3.22}$$

Algorithm 2 shows the routine used to implementing Equations 3.18 to 3.22 The algorithm guarantees convergence to the optimal w^* to within

Algorithm 2 Object Detection[18]

```
1: procedure
2:     Input : $\epsilon \geq 0$
3:     $w_0 := 0, t_0 := 0 P = \{\}$
4:      repeat
5:      $t = t + 1$
6:      Compute Plane Tangent to $R_{emp}(w_{t\text{-}1})$
7:      select $a_t\ \epsilon \delta R_e mp(w_{t-1})$
8:      and $b_t := R_{emp}(W_{t-1}) - \langle w_{t-1}, a_t \rangle$
9:      $P_t := P_{t-1} \cup \{a_t,\ b_t\ \}$
10:     Let $K_t(w)\ = \frac{1}{2}||w||^2 + max_{(a_i,b_i)\epsilon P_t}[\langle w, a_i \rangle + b_i]$
11:     $w_t := argmin_w K_t(w)$
12:    until $\frac{1}{2}||w_t||^2 + R_{emp} w(t) - K_t(w_t) \leq \varepsilon$
13:    RETURN: $w_t$
```

$\varepsilon$by terminating the algorithm. Equation 3.23

$$|J(w^*) - J(w_t)| < \epsilon \tag{3.23}$$

At the 11th step of Algorithm 2 an argmin must be solved. This step is represented as a quadratic program in Equation 3.24.

$$\begin{aligned} \min_{w,\xi} \frac{1}{2}||w||^2 + \xi \\ \textit{such that } \xi \geq \langle w, a_i \rangle + b_i, \forall (a_i, b_i) \epsilon P \end{aligned} \tag{3.24}$$

The set of variables being normalized w will have many more dimensions than the number of constraints. Therefore a dual problem solving Lagrangian function is considered. The Lagrangian of the quadratic program is in Equation 3.25

$$\begin{aligned} \max_{w,\xi,\lambda} L(w, \xi, \lambda) \\ s.t. \nabla_w L(w, \xi, \lambda) = 0, \\ \nabla_\xi L(w, \xi, \lambda) = 0, \\ \lambda_i \geq 0\ \forall\ i \end{aligned} \tag{3.25}$$

The Lagrangian is reduced to the following quadratic program in Equation 3.26.

$$\max_{y} \lambda^T b - \frac{1}{2}\lambda^T Q \lambda$$
$$\textit{such that } \lambda_i \geq 0, \sum_{i=1}^{|P|} \lambda_i = 1 \tag{3.26}$$

Iteratively a pair of Lagrangian multipliers (λ_b, λ_l)are selected which violate the Karush-Kuhn-Tucker (KKT) conditions. The pair is optimized using the Platt's sequential minimal optimization to reduce the dual quadratic problem. The iteration stops when the duality gap falls below a threshold. Upon solving for λ^*, the w_t which can be used in step 11 of Algorithm 2 is shown in equation 3.27.

$$w_t = -\sum_{i=1}^{|P|} \lambda^* a_i \tag{3.27}$$

The final stage of the algorithm is the calculation of R_{emp} and an element of it's subgradient. If we consider 3.19 as R_{emp}, then an element of the subgradient is given by equation 3.28

$$\delta R_e mp(w) = \frac{C}{n}[\sum_{r \epsilon y_i^*} \phi(x_i, r) - \sum_{r \epsilon y_i} \phi(x_i, r)] \tag{3.28}$$

where y^* is represented as

$$y_i^* = argmax_{y \epsilon Y}[\Delta(y, y_i) + \sum_{r \epsilon y} \langle w, \phi(x_i, r)\rangle] \tag{3.29}$$

Using this process to maximise margins, it is found that compared to the original approach shown in Algorithm 1, the optimized procedure would select a set of parameters which make the new algorithm perform significantly better.

4 Implementation

This chapter gives an overview of the technologies and the libraries used for the implementation of the Concept. A key consideration during the development process was to use open source libraries for easy portability and extensibility of the framework. The base language of programming is C++. Since C++ is a statically typed language, it is generally yields better performance than dynamically typed languages because the code is type-checked before it is executed. The performance of Python as a programming language is found to be significantly slower when compared to C++. The hardware specifications for the system used for the development process is; 4-Core 64-bit architecture processor running Windows 10 and 8GB of RAM. The Isolator Detection framework has two main module, the

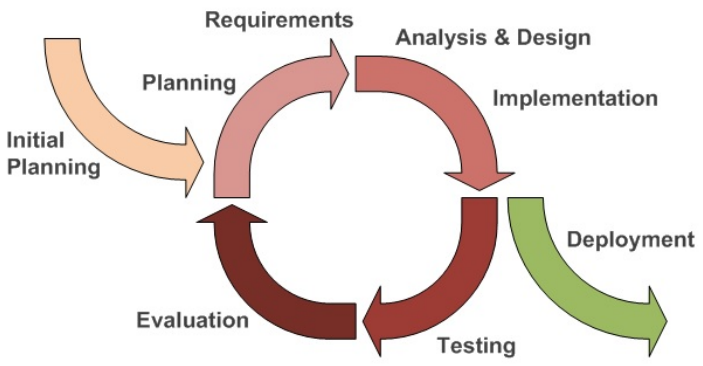
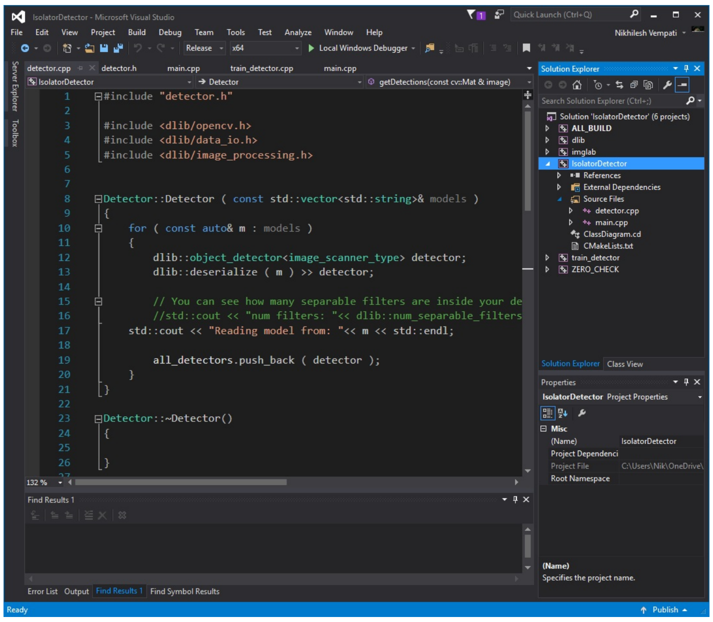


Figure 4.1: Agile Development Process

training module and detection module. The Agile development process is used for the implementation such that the core functions are implemented first, and then further functions are added iteratively. The iterative agile development cycle is illustrated in Figure 4.1.

4.1 Microsoft Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It supports the development of computer programs with multiple programming languages. Since Visual Studio includes a code editor supporting auto code completion as well as code refactoring for faster development of the code. The integrated debugger works both as a source-



```
#include "detector.h"

#include <dlib/opencv.h>
#include <dlib/data_io.h>
#include <dlib/image_processing.h>


Detector::Detector ( const std::vector<std::string>& models )
{
    for ( const auto& m : models )
    {
        dlib::object_detector<image_scanner_type> detector;
        dlib::deserialize ( m ) >> detector;

        // You can see how many separable filters are inside your de
        //std::cout << "num filters: "<< dlib::num_separable_filters
    std::cout << "Reading model from: "<< m << std::endl;

        all_detectors.push_back ( detector );
    }
}

Detector::~Detector()
{

}
```

Figure 4.2: Visual Studio IDE

level debugger and a machine-level debugger. Microsoft Visual Studio Community edition is available free of cost and supports the development of C and C++ out of the box.

Microsoft Visual C++ is Microsoft's implementation of the C and C++

compiler and related language services and particular tools for use with the Visual Studio IDE. For C++, it takes after the ANSI C++ implementation alongside several C++11 features. It also supports the C++/CLI specification to write managed code, as well as mixed-mode code. Microsoft positions Visual C++ for development in native code or in code that contains both native as well as managed components. A Visual studio solution can be configured to work with external libraries to use with 3rd party and open-source APIs.

For easier navigation and control of a project modules, Visual Studio code editor supports setting bookmarks in code. Other navigational aids include collapsing code blocks and incremental search, in addition to normal text search and regex search. Code refactoring is also possible such as parameter reordering, variable and method renaming, interface extraction and encapsulation of class members inside properties. Visual studio features background compilation, to enable real-time type checking.

4.2 External Libraries

The external libraries used for the development of the framework are the opencv library and the dlib library. These are explained below.

4.2.1 Opencv

OpenCV is an open source computer vision and machine learning software library of more than 2500 optimized algorithms in C++, C, Python and Java. OpenCV is used extensively in real-time vision based applications and is natively described in C++. This implementation uses OpenCV version 2.4. OpenCV is licensed by Berkley Software Distribution, and it allows user to utilize and modify the code according to the requirements.

OpenCV can be used by programmers using a base programming language such as C++. OpenCV is available for platforms other than Windows such as Android, OS X, iOS and Linux. This is a very useful functionality because the software developed for one platform can be ported to other platforms as well. The goal of OpenCV is to provide simple to use

Figure 4.3: OpenCV

computer vision infrastructure that helps people build fairly sophisticated vision applications quickly. Since computer vision and machine learning work together in a lot of applications, OpenCV also contains a full general purpose Machine Learning Library(MLL). This sub library is focused on statistical pattern recognition and clustering.

4.2.2 Dlib

Dlib is a C++ library that contains machine learning algorithms and other tools for creating complex software in C++. It is utilized extensively in industry and academia in a wide range of domains such as embedded systems, mobile phones and vision-based applications. Dlib has an open source license therefore allowing the user to use it in any application.

Dlib provides complete documentation for every class and function in API for faster development. Also, the debugging mode allows for checking the preconditions for the functions which can help catch the vast majority of bugs caused by calling functions incorrectly or using objects in an incorrect manner. The other major advantage of Dlib is that there is no installation or configuration step needed before usage.

4.3 Isolator Detection Framework

The Isolator Detection Framework is developed in order to demonstrate the training and detection procedures researched in this master thesis. It consists of 2 major modules, one for training and one for detection.

The dataset used for the training consists of several images of isolators extracted from several videos. The data contains of isolators in several positions, orientations and scales and in total amount to close to 4500 samples. These two modules are explained in the sections below.

4.3.1 Training Module

The training module is the first of the two modules in the framework. It is developed for the purposes of training the machine with a training set. The algorithm for the training module is given in Algorithm 3.

Algorithm 3 Training Module

```
 1: procedure TRAIN(images)
 2:     Input: user command, ImageList
 3:     Parse user command
 4:     Read  Input Image List
 5:     Sort Positive and Negative Samples
 6:     Set SVM Learning Parameters
 7:     Set number of processing threads
 8:     Set Training Window Size
 9:     CreateTraining Window ← Size
10:     Scale to multi-scale pyramid
11:     Call SVM
12:     while (!endof.ImageList) do
13:         Load Positive Images
14:         Load Negative Images
15:         flip Images
16:         Create or Modify Images Vector
17:     if (user command == t) then
18:         while (!100% Detection Rate) do
19:             Feature Extraction
20:             Train SVM(Image Features)
21:         Return:  Trained Model
22:     if (cross validation == 1) then
23:         Print:  Confusion Matrix
```

Images are fed to the system with annotations describing the location of the isolator in each image. Figure 4.4 shows the example of a training image with annotations around the isolator. The annotation on the images can be done by several annotating tools available on the internet. For our purposes we used Dlib's annotation tool imglab which generates an xml file with all annotations.

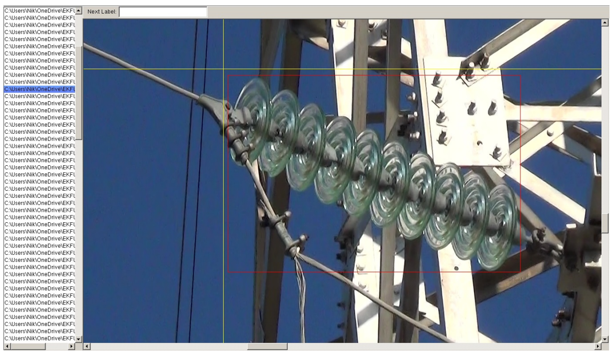


Figure 4.4: Annotations around Isolators

```
<--\\Training Data -->
<images>
  <image file='C:\Users\Isolator Dataset\horiso013.jpg'>
      <box top='323' left='565' width='390' height='278'/>
      <box top='729' left='266' width='275' height='173'/>
  </image>
</images>
```

The system expects the name of the xml file, along with a few more options in the console command, as shown below -

```
Train-Detector.exe images-list.xml -options
```

The learning parameters for the SVM are set to achieve high accuracy. For the purposes of detecting isolators, we found the best values to be C = 1.0, ε= 0.01 and the detection window size to be the default 80x80 pixels.

The number of processing threads are set to 4 for our 4-core processor.

```
const double C = get_option(ConIn, "c", 1.0);
const double eps = get_option(ConIn, "eps", 0.01);
const unsigned long target_size = get_option(ConIn,
   "target-size", 80*80);
const int threads = get_option(ConIn, "threads", 4);
```

A training window is created with the specified size and the images are scaled to a multi-scale pyramid. We have chosen 5 levels. The images are then flipped so each image can contribute twice to the training. Then on each image the training window is applied on all possible positions to extract the features. The SVM is called with the input as the features from the images labelled positive and negative. The training is performed on the training data and a trained svm model is generated.

The training of our images were done on different configurations of isolators, the three sets containing isolators of similar type in varying positions.

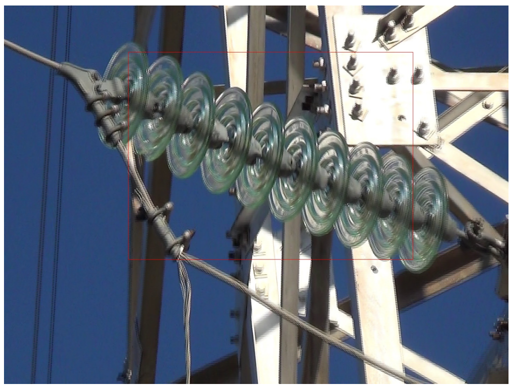


Figure 4.5: Single Detection

4.3.2 Detector Module

The second module of the framework is responsible for loading an input frame of video mark, the location of isolators and save the key-frames. The procedure for the detection module is given in Algorithm 4.

Algorithm 4 Detector Module

```
1: procedure DETECT($Video$)
2:     input : Video File
3:     input : SVM Trained Models
4:     read : Video File
5:     while !lastframe do
6:         load : Frame
7:         call : Detect(frame)
8:         if detection then
9:             draw : Detection
10:            save : Save to Hard-Drive
11:    end
```

The algorithm is initiated by running the detector executable. The code is pre-programmed to load the SVM models from *models.txt* and a list of input data from *input.txt*. This can be overriden by calling the application with a filename with its path.

```
IsolatorDetector.exe C:\IsolatorVideos\videoname.type
```

The SVM is initiated with the trained model and the video file is loaded into the memory.

Each frame in the video is scanned and its features are extracted and then scanned by the SVM. Detections are marked with an outlining box. Frames with detections are saved to a specified output folder. Figures 4.5

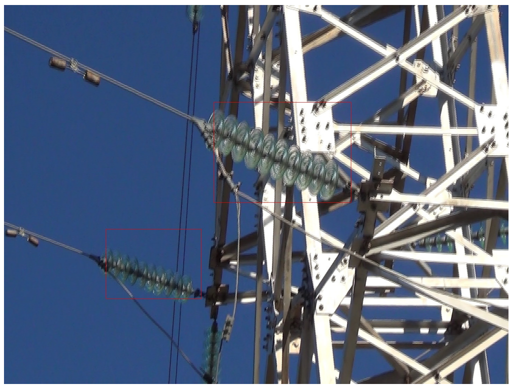


Figure 4.6: Multiple Detections

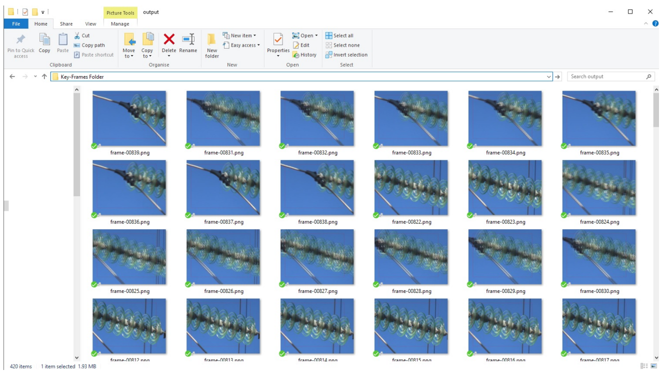


Figure 4.7: Key-Frames Folder

and 4.6 represent two detected frames representing single and multiple detections. Figure 4.7 shows the Key-Frame folder populated with positive detected frames.

5 Data Evaluation and Results

This chapter will focus on the experiments performed in order to test the framework described in the previous chapter. The acquired dataset used for the training and testing will be explained. The test conditions are explained first, followed by the results for each test condition.

5.1 Training Evaluation

The data used for this project was acquired from within *Technische Universität Chemnitz*'s database of videos taken from surveillance drones. This dataset when examined could be categorized into videos that were recorded from a stable position and those with a lot of instability. Isolator images were extracted from these videos and appended with a few taken from the internet. These isolators were then classified into 3 groups based on their orientation and position. The example for these three types can be seen in Figure 5.1. The training procedure was performed with 2-fold cross

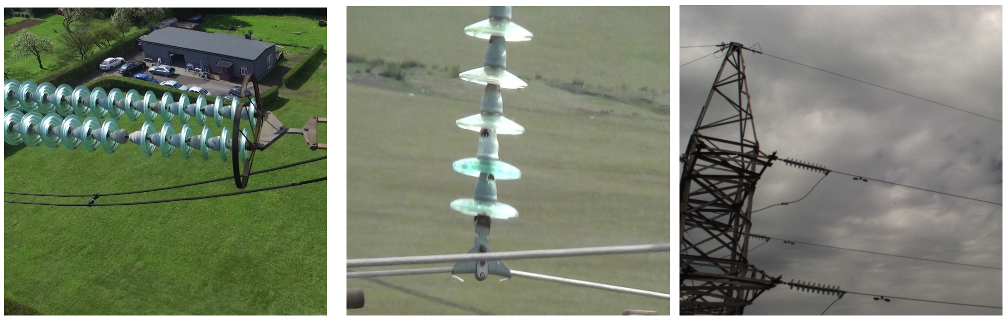

(a) Horizontal (b) Vertical (c) Low-Light

Figure 5.1: Training Data

validation till a model was generated taking into consideration over and under-fitting. Tables 1,2,3 illustrate the confusion matrix details for each training iteration for each model found with a threshold of -0.5.

Test Size	True Positives	False Positives	True Negatives	False Negatives	Precision	Recall	Fall Out
25	2	7	3	13	0.222	0.133	0.70
35	7	6	9	13	0.538	0.35	0.4
45	21	2	13	9	0.913	0.700	0.133
55	39	0	15	1	1	0.975	0
65	49	1	14	1	0.98	0.98	0.067
75	57	2	13	3	0.966	0.95	0.133
100	51	5	35	9	0.911	0.85	0.125
125	65	7	43	10	0.903	0.867	0.14
150	87	7	43	13	0.926	0.87	0.14

Table 5.1: Training Method for Horizontal Isolators

The training for the set of horizontal isolators reaches an optimal value of Recall and low Fall-Out at 55 test images which can be seen in the graph in Figure 5.2. The point in the graph where the precision and recall are

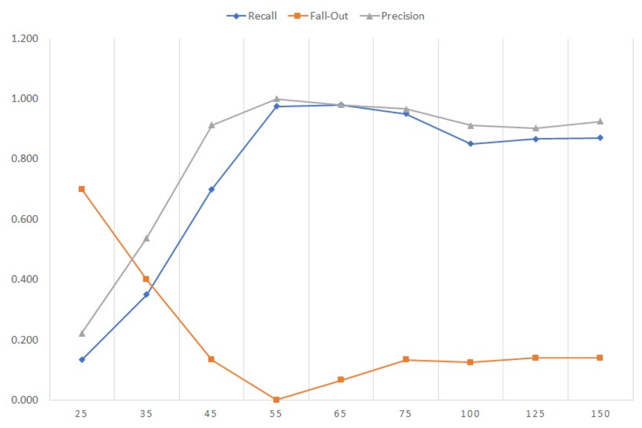


Figure 5.2: Precision-Recall Curve - Set 1

highest is found to be the ideal model in the training process.

The process is repeated for the category of vertical isolators and the precision,recall graphs are represented in Figure 5.3. The highest values for precision and recall are found at 275 Test Images. The generated model is taken as the model of vertical isolators.

Test Size	True Positives	False Positives	True Negatives	False Negatives	Precision	Recall	Fall Out
100	34	5	10	51	0.872	0.40	0.333
150	77	3	7	63	0.963	0.55	0.30
200	123	3	17	57	0.976	0.683	0.15
250	222	2	18	8	0.991	0.965	0.100
275	266	1	4	4	0.996	0.985	0.200
300	276	2	13	9	0.993	0.968	0.133
350	306	2	8	34	0.994	0.900	0.200

Table 5.2: Training Method for Vertical Isolators

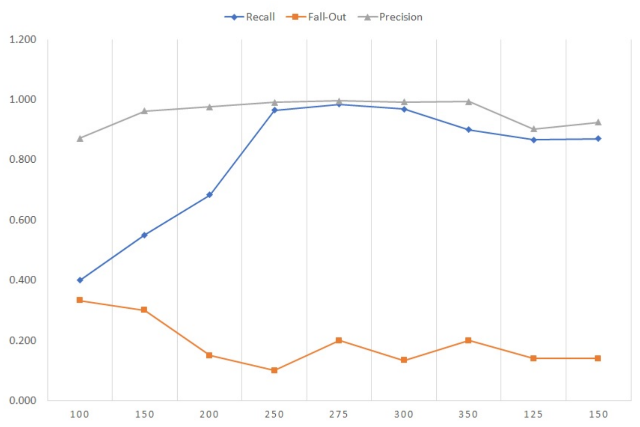


Figure 5.3: Precision-Recall Curve - Set 2

Test Size	True Positives	False Positives	True Negatives	False Negatives	Precision	Recall	Fall Out
100	29	15	35	21	0.659	0.58	0.30
150	90	3	7	50	0.968	0.643	0.30
200	82	16	64	38	0.837	0.683	0.20
250	191	1	4	54	0.995	0.78	0.20
275	208	3	12	52	0.986	0.8	0.20
300	242	3	22	33	0.988	0.88	0.13
350	286	3	7	54	0.990	0.841	0.30

Table 5.3: Training Method for Lowly-illuminated Isolators

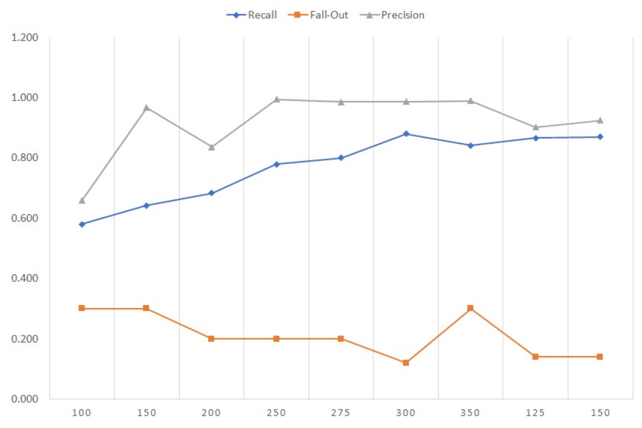


Figure 5.4: Precision-Recall Curve Set-3

The same procedure finds the ideal model for lowly illuminated isolators at 300 Test Images. See graph in Figure 5.4.

5.2 Test Procedure

To evaluate this procedure we manually choose 100 un-annotated test images that show different views of the isolators under differing lighting conditions and stability. We then feed the detector with these images and adjust the threshold value τ_d of the detection module in order to set the sensitivity of the detection system. We then measure the precision and recall for different threshold values.

Values less than *-2* and greater than *1* were found to be unusable and so the tests were bounded to within those values.

Threshold τ_d	True Positives	False Positives	True Negatives	False Negatives	Precision	Recall
-1.5	75	22	3	0	0.773	1
-1.25	75	18	7	0	0.806	1
-1	75	15	10	0	0.833	1
-0.75	75	7	18	0	0.914	1
-0.70	75	2	23	0	0.974	1
-0.65	75	0	25	0	1	1
-0.5	74	0	25	0	1	0.986
-0.25	73	0	25	2	1	0.973
0	70	0	25	5	1	0.933
0.25	63	0	25	12	1	0.84
0.5	52	0	25	23	1	0.693

Table 5.4: Measurements with varying τ_d

The value of *-0.65* for the Threshold,τ_d is found to be the point with the best precision and recall rating for our detection module.

Once the optimal threshold value has been found. The detection module is then run on two different types of video footage. One which is stable and one which is unstable. For the purposes of detection it is also imperative to have the videos in a progressive format by performing deinterlacing.

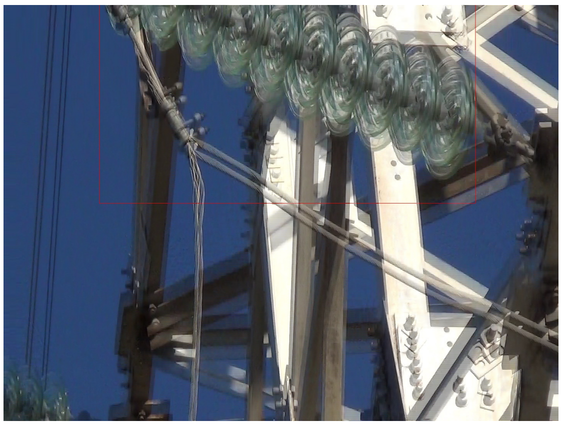


Figure 5.5: Interlacing caused motion artifacts

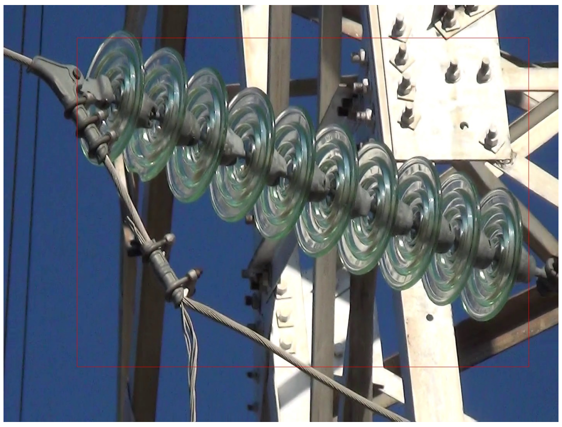


Figure 5.6: Deinterlaced frames without motion artifacts

Since most video cameras record by the superposition of 2 images to form a frame, an individual frame can cause motion based artifacts seen in Figure 5.5. Figure 5.6 shows the same isolator frame free of motion artifacts after deinterlacing has been performed.

5.3 Unstable Video Feed

The goal of the isolator detection framework is to extract the frames that contain isolators. The algorithm loads the video and three frames are selected for each second of the video. These frames are then evaluated for isolators. For every positive detection, the frame is saved to a designated location. This set of saved frames are termed as the key-frames.

Video	Length (in minutes)	Frames with Isolators	Video Type	Key-Frames Extracted
1	00:57	75%	Stable	60
2	04:01	15%	Unstable	47
3	07:14	10%	Unstable	72
4	09:22	4%	Unstable	24
5	0:42	0%	Stable	2

Table 5.5: Key-frames extracted

Table 5.5 represents the key-frames extracted for several videos. It is worth noting that when presented with unstable videos, the system extracts key-frames while omitting the parts of the video where no isolators are detected. This leads to a significant saving in time when compared to manually going through the entire video.

6 Conclusion and Future Work

A modern power transmission system must stay online in order to provide uninterrupted electricity. The inspection of power lines and isolators, though long been a manual process, is now being replaced with drone based surveillance. Since this technology is still in it's infancy, the scope for improvement is quite large. This chapter summarizes the results of the isolator detector framework and how improvements can be made in the next stages. In this thesis we presented a framework for isolator detection.

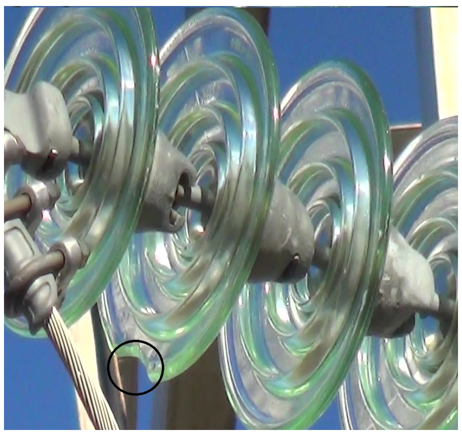


Figure 6.1: Fault Detection

Two different modules were implemented after careful comparison of several state of art object detection algorithms. The deformable parts model is found to provide very good results in terms of detection quality and elimination of false alarms. Like all object detection algorithms, even this model is not perfect and several features could be added to the framework to make it perform better.

The framework automatically saves the frames of a video that contain isolators. The system proves itself robust to varying positions and configurations of isolators. The system performs with a lower accuracy when in low-light situations where only the silhouette of the isolator can be seen, however simple changes in illumination and scale are dealt with easily. The extracted key-frames are found to significantly reduce the data the user needs to analyse, therefore making the inspection process faster and efficient.

In the future the vision framework can be extended to feature not just isolators but also transmission towers, power cables and binders. Such a framework would make the inspection of transmission lines much more streamlined. For each extracted key-frame the object is scanned for damage and if the damage is greater than a certain threshold, the frame is marked and saved accordingly. An example sample is shown in Figure 6.1.

Finally, this vision based application can be modified to run on embedded controllers leading to SOCs in drones that manage the tasks of flying, recording and detection. This means lower power consumption, higher data throughput if using digital signal processors and most importantly ease of use.

Bibliography

[1] Automated power line inspection (2016), https://www.tu-chemnitz.de/informatik/ce/projects/projects.php.en#apoli

[2] Failure of transmission line insulators (22/08/2014), http://www.inmr.com/failure-transmission-line-insulators/

[3] Ajmal, Ashraf, Shakir, Abbas, Shah: Video summarization: Techniques and classification (2012), http://link.springer.com/chapter/10.1007/978-3-642-33564-8_1

[4] Barhoumi, Zagrouba: On-the-fly extraction of key frames for efficient video summarization (2013), http://ac.els-cdn.com/S2212671613000140/1-s2.0-S2212671613000140-main.pdf?_tid=47802910-2dc0-11e7-9ed2-00000aacb362&acdnat=1493569131_7e682c72b3013bf99ed980ee4b6bd78e

[5] Bay, Tuytelaars, Gool, V.: Surf: Speeded up robust features (11/2004), http://www.vision.ee.ethz.ch/~surf/eccv06.pdf

[6] Dalal, Triggs: Histograms of oriented gradients for human detection (07/2005), http://ieeexplore.ieee.org/document/1467360/metrics

[7] Dunning: Accurate methods for the statistics of surprise and coincidence (03/1993), http://dl.acm.org/citation.cfm?id=972454

[8] Evans: The drone network of tomorrow(it's closer than you think) (last viewed on: 30/04/2017), http://dronelife.com/2017/02/03/drone-network-tomorrow-closer-think/

[9] Felzenszwalb, Girshick, McAllestar, Ramanan: Object detection with discriminatively trained part-based models (11/2010), http://ieeexplore.ieee.org/document/5255236/

[10] Furini, Ghini: An audio-video summarization scheme based on audio and video analysis (01/2006), http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1593230

[11] Hall, Coyne: The political economy of drones (2013), http://www.tandfonline.com/doi/pdf/10.1080/10242694.2013.833369?needAccess=true

[12] Han, Hu, Wang, Wu, Yoshigahara: Enhanced sports video shot boundary detection based on middle level features and a unified model (08/2007), http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4341601

[13] Harris, Stephens: A combined corner and edge detector (1988), http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.231.1604

[14] Hassaballah, Abdelmgeid, Alshazly: Image features detection, description and matching (2016), http://www.springer.com/cda/content/document/cda_downloaddocument/9783319288529-c2.pdf?SGWID=0-0-45-1549477-p177983508

[15] Hersha, Johnson, Meredith, Pittsley, Tameesh, Wilson: Drone monitoring of power lines (2015), http://www.egr.msu.edu/classes/ece480/capstone/spring15/group14/uploads/4/2/0/3/42036453/ece_480_senior_design_final_report_.pdf

[16] Jain, Dubes: Algorithms for clustering data (1998), http://homepages.inf.ed.ac.uk/rbf/BOOKS/JAIN/Clustering_Jain_Dubes.pdf

[17] Khattabi, Tabii, Benkaddour: Video summarization: Techniques and application (2012), http://waset.org/publications/10000964/video-summarization-techniques-and-applications

[18] King, D.: Max-margin object detection (01/2015), `https://arxiv.org/pdf/1502.00046.pdf`

[19] Liu, Hua, Chen: A hierarchical visual model for video object summarization (12/2010), `http://ieeexplore.ieee.org/document/5401164/`

[20] Lowe: Distinctive image features from scale-invariant keypoints (11/2004), `https://link.springer.com/article/10.1023%2FB%3AVISI.0000029664.99615.94`

[21] Mikolajczyk, Schmid: A performance evaluation of local descriptors (10/2005), `https://www.robots.ox.ac.uk/~vgg/research/affine/det_eval_files/mikolajczyk_pami2004.pdf`

[22] Morevic: Towards automatic visual obstacle avoidance (1977), `http://dl.acm.org/citation.cfm?id=1622947`

[23] Nezhad, Gholami, Jalilian, Hassanzahdeh: Performance improvement of insulator string in polluted conditions (2016), `http://journal.esrgroups.org/jes/papers/4_3_8.pdf`

[24] Ojala, Pietikainen, Maenpaa: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns (07/2002), `http://dl.acm.org/citation.cfm?id=628808`

[25] Pritch, Ratovitch, Hendel, Peleg: Clustered synopsis of surveillance video (2009), `http://www.cs.huji.ac.il/~peleg/papers/avss09-ClusteredSynopsis.pdf`

[26] (Publisher), D.M.: News on drones: Cebit (last viewed on: 23/04/2017), `http://www.cebit.de/de/news/thema/drones`

[27] Rosten, Drummond: Fusing points and lines for high performance tracking (10/2005), `http://ieeexplore.ieee.org/document/1544896/`

[28] Smith, Brady: Susan - a new approach to low level image processing (1995), `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.27.8270&rep=rep1&type=pdf`

[29] Varah: Image processing on live video streams should be more than just mustaches and swapping faces (26/05/2016), https://www.linkedin.com/pulse/image-processing-live-video-streams-should-more-than-just-sean-varah

[30] Wang, Tang: Dual-space linear discriminant analysis for face recognition (2004), http://www.ee.cuhk.edu.hk/~xgwang/papers/wangTcvpr04_dualspace.pdf

[31] Wolf: Key frame selection by motion analysis (05/1996), http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=543588

[32] Yeo, Lui: Rapid scene analysis on compressed video (12/1995), http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=475896

[33] Zabih, Miller, Mai: A feature based algorithm for detecting and classifying scene breaks (11/1995), http://www.cs.cornell.edu/~rdz/Papers/ZMM-MM95.pdf

[34] Zanaty: Support vector machines (svms) versus multilayer perception (mlp) in data classification (08/2012), http://ac.els-cdn.com/S1110866512000345/1-s2.0-S1110866512000345-main.pdf?_tid=379edde4-43e0-11e7-9237-00000aab0f01&acdnat=1496001774_eebc0fd8172547175389c3c73bc87787

[35] Zhang, Zhong, Smoliar: An integrated system for content-based video retrieval and browsing (07/1996), http://ac.els-cdn.com/S0031320396001094/1-s2.0-S0031320396001094-main.pdf?_tid=dc7dc074-3d95-11e7-9da0-00000aacb362&acdnat=1495310131_1c6026040139f8e3180a9cde0f6974db