

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018



A PROJECT REPORT (15CSP85) ON

“A New Data Preparation Methodology in Machine Learning-based Haze Detection algorithm”

Submitted in Partial fulfillment of the Requirements for the Degree of
Bachelor of Engineering in Computer Science & Engineering

By

MANJULA K (1CR16CS082)

GEETHIKA PRIYA M (1CR16CS097)

RESHMA M (1CR16CS131)

VANDANA PRIYA B (1CR16CS175)

Under the Guidance of,

Mrs. NAVANEETHA.M

Assistant professor, Dept. of CSE



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

Certified that the project work entitled “**A New Data Preparation Methodology in Machine Learning-based Haze Detection algorithm**” carried out by **Ms. MANJULA K, USN 1CR16CS082, Ms. GEETHIKA PRIYA M, USN 1CR16CS097, Ms. RESHMA M, USN 1CR16CS131, Ms. VANDANA PRIYA B, USN 1CR16CS175**, bonafide students of CMR Institute of Technology, in partial fulfillment for the award of **Bachelor of Engineering** in Computer Science and Engineering of the Visveswaraiah Technological University, Belgaum during the year 2019-2020. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report as been approved as it satisfies the academic requirements in respect to Project work prescribed for the said Degree.

Mrs. Navaneetha M

Asst Professor

Dept. of CSE, CMRIT

Dr. Prem Kumar Ramesh

Professor & Head

Dept. of CSE, CMRIT

Dr. Sanjay Jain

Principal

CMRIT

External Viva

Name of the examiners

1.

2.

Signature with date

DECLARATION

We, the students of Computer Science and Engineering, CMR Institute of Technology, Bangalore declare that the work entitled " **A New Data Preparation Methodology in Machine Learning-based Haze Detection algorithm** " has been successfully completed under the guidance of **Asst Prof. Mrs. Navaneetha M**, Computer Science and Engineering Department, CMR Institute of technology, Bangalore. This dissertation work is submitted in partial fulfillment of the requirements for the award of Degree of Bachelor of Engineering in Computer Science and Engineering during the academic year 2019-2020. Further the matter embodied in the project report has not been submitted previously by anybody for the award of any degree or diploma to any university.

Place: Bengaluru

Date:

Team members:

MANJULA K (1CR16CS082)

GEETHIKA PRIYA M (1CR16CS097)

RESHMA M (1CR16CS131)

VANDANA PRIYA B(1CR16CS175)

ABSTRACT

Haze removal is an intellectually challenging object of scientific study. Haze removal approaches founded upon machine learning require a large and reliable training database. Researchers are currently using the synthetic database due to the complexity of real database acquisition. To introduce the synthetic haze into the clear images, they assume that the depth map is drawn from the standard uniform distribution. In this paper, we present a new methodology for preparing the synthetic training database, in which the proposed equidistribution is employed instead of standard uniform distribution.

ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude and respect to **CMR Institute of Technology, Bengaluru** for providing me a platform to pursue my studies and carry out my final year project.

I have a great pleasure in expressing my deep sense of gratitude to **Dr. Sanjay Jain**, Principal, CMRIT, Bengaluru, for his constant encouragement.

I would like to thank **Dr. Prem Kumar Ramesh**, HOD, Department of Computer Science and Engineering, CMRIT, Bangalore, who has been a constant support and encouragement throughout the course of this project.

I consider it a privilege and honor to express my sincere gratitude to my guide **Mrs. Navaneetha M**, Assistant Professor, Department of Computer Science and Engineering, for the valuable guidance throughout the tenure of the review.

I also extend my thanks to all the faculty of Computer Science and Engineering who directly or indirectly encouraged me.

Finally, I would like to thank my parents and friends for all their moral support they have given me during the completion of this work.

TABLE OF CONTENTS

	Page No.
Certificate	Ii
Declaration	Iii
Abstract	Iv
Acknowledgement	V
Table of contents	Vi
List of Figures	Viii
List of Abbreviations	Ix
1. INTRODUCTION	1
1.1 Relevance of the Project	1
1.2 Problem Statement	2
1.3 Objective	2
1.4 Scope of the Project	2
1.5 Methodology	2
1.6 Chapter Wise Summary	4
2. LITERATURE SURVEY	5
2.1 Technical and Research Papers	5
3. SYSTEM REQUIREMENTS SPECIFICATION	9
3.1 Functional Requirements	9
3.2 Non Functional Requirements	10
3.3 Hardware Requirements	10
3.4 Software Requirements	10

4. SYSTEM ANALYSIS AND DESIGN	11
4.1 System Architecture	11
4.2 Data Set	12
4.3 Data Flow for Use Case	13
4.4 Data Flow for Class Diagram	14
4.5 Data Flow for Sequence Diagram	14
5. IMPLEMENTATION	16
5.1 Gray Level Co-Occurrence Matrix	16
5.2 Principal Component Analysis	17
5.3 Support Vector Machine	20
6. RESULTS AND DISCUSSION	29
6.1 Training Model for Haze	29
6.2 Classifying the image	30
6.3 Accuracy	32
6.4 Results-Haze	33
7. TESTING	35
7.1 Software Testing Introduction	35
7.2 STLC (Software Testing Lifecycle)	35
7.2.1 Test Plan	35
7.2.2 Test Development	35
7.2.3 Test Execution	35
7.2.4 Analyze Result	36
7.3 Types of Testing	36
7.3.1 White Box Testing	36
7.3.2 Black Box Testing	36
7.3.3 Grey Box Testing	36

7.4 Level of testing used in project	37
7.4.1 Unit Testing	37
7.4.2 Integration Testing	37
7.4.3 System Testing	37
7.4.4 Functional Testing	37
7.4.5 Alpha Testing	37
7.4.6 Beta Testing	37
8. Conclusion and Future Scope	38
8.1 Conclusion	38
8.2 Contribution	38
8.3 Future Scope	38
REFERENCES	39

LIST OF FIGURES

	Page No.
Fig 1.1 Methodology	3
Fig 2.1 Atmosphere Scattering Model	6
Fig 2.2 Square in different weather	7
Fig 4.1 System Architecture	11
Fig 4.2 Datasets	13
Fig 4.3 Use Case Diagram	14
Fig 4.4 Class Diagram	14
Fig 4.5 Sequence Diagram	15
Fig 5.1 GLCM	16
Fig 5.2 Linear v/s Nonlinear	21
Fig 5.3 Kernel	24
Fig 5.4 Mapping	25
Fig 6.1 Training Model For Haze	29
Fig 6.2 K080-000009	30
Fig 6.3 M080-000050	30
Fig 6.4 U080-000042	31
Fig 6.5 L080-000040	31
Fig 6.6 LIMA-000009	32
Fig 6.7 Overall Accuracy	33
Fig 6.8 Sample Image	33
Fig 6.9 Single Image Output after Testing	34
Fig 7.1 Labels and Image	36

LIST OF ABBREVIATIONS

ABBREVIATIONS

GLCM

SVM

PCA

MEANING

Grey Level Co-Occurrence Matrix

Support Vector Machine

Principal Component Analysis

CHAPTER 1

INTRODUCTION

Haze is usually used to indicate the natural and anthropogenic aerosols, which are present in the atmosphere and are the major cause of the reduction in image clarity. Most of existing methods in the field of computer vision are based on the hypothesis/assumption that the weather in outdoor images or videos are clear. However, different weather conditions such as rain, snow or haze will cause complex visual effects in images or videos. Accordingly, haze removal holds many practical applications in various technology areas. Haze removal approaches founded upon machine learning require a large and reliable training database. The general idea behind most machine learning is that an algorithm learns to handle a task by studying from a set of examples. Then, it will perform the same task with new data it has not encountered before. Therefore, the training data is absolutely of crucial importance. For the haze removal problem, collecting a real database is extremely complicated So we are using synthetic database to train the machine learning algorithm.

In this methods, We are going to create a image having a synthetic haze which is helpful to remove the haze algorithm. The general idea behind most machine learning is that an algorithm learns to handle a task by studying from a set of examples. Then, it will perform the same task with new data it has not encountered before. Therefore, the training data is absolutely of crucial importance. However, for the haze removal problem, collecting a real database is extremely complicated. In their methods, the depth map is assumed to follow standard uniform distribution but most of current pseudo random number generators do not guarantee the uniform distribution. In this paper, we, hereby, propose the novel methodology for preparing the training database, in which the equidistribution is employed to guarantee that the artificially-generated depth map is of truly uniform distribution. Experimental results show that algorithms retrained with our database outperform the original ones.

1.1 RELEVANCE OF THE PROBLEM

- To be able to create a synthetic dataset for haze removal algorithm application
- Synthetic dataset using frida2dataset.
- Applying feature extraction using GLCM on synthetic Frida2 dataset and PCA for detecting the haze images and non free haze images.
- Saving the image and labels array for future use.

1.2 PROBLEM STATEMENT

- To be able to create a synthetic dataset for haze removal algorithm application.
- Use of Support Vector Machine classifier and machine-learnt features to classify hazy images from the normal ones.

1.3 OBJECTIVES

- Creation of synthetic dataset usingFrida2.
- Frida2-It contains Haze and Haze-free Images.
- Feature Extraction using GLCM and PCA on frida2dataset.
- Classification of haze using SVM (Support Vector Machine).

1.4 SCOPE OF THE PROJECT

- Training the model whether it is a haze images or non free haze images.
- To remove haze we can apply filters like Gaussian filter, Box filter, Average filter.

1.5 METHODOLOGY

- We propose the equidistribution, which closely resembles the shape of theoretical uniform distribution.

- The idea behind our methodology is somewhat simple. It takes the pseudo uniformly distributed random sequence as the input, creates histogram and sorts it into descending and ascending orders.
- Then ,one of two sorted histogram is chosen to subtract from their average, which better resembles the shape of theoretical uniform distribution,
- This helps in creating synthetic haze images.
- Further the normal and hazy images are given into a SVM classifier, after a feature extraction method employed will be GLCM.
- This helps in understanding the robustness of these algorithms towards haze in images and their ability to classify them as hazy images.

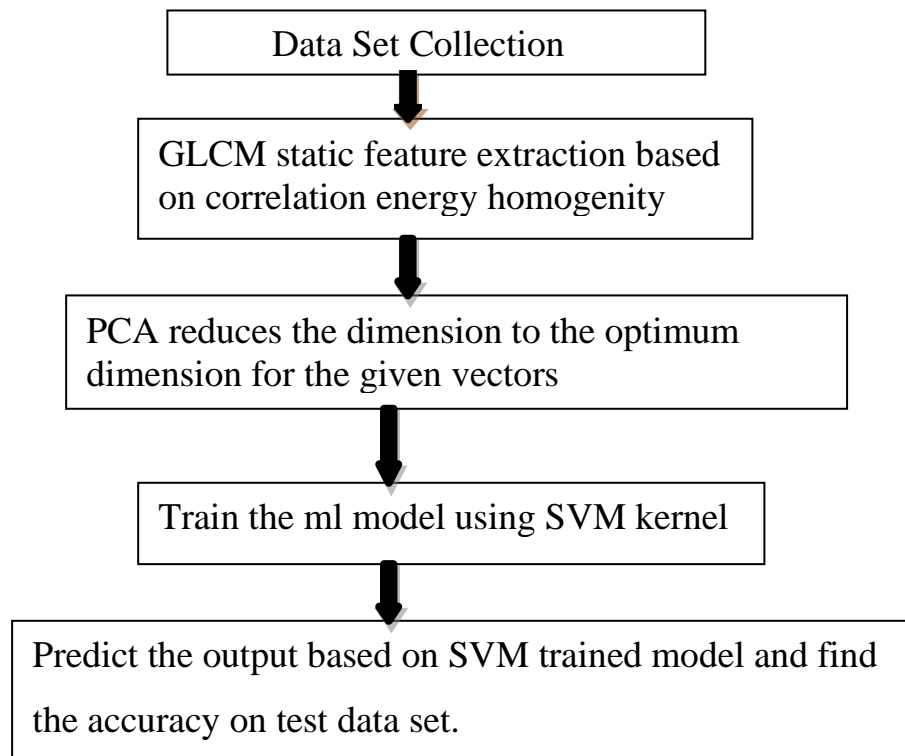


Fig 1.1: Methodology

1.6 CHAPTER WISE SUMMARY

Here is the chapter wise summary of the report.

- **Chapter 2: Literature Survey**

In this we have reviewed some research papers and compared them to find the best approach for our project.

- **Chapter 3: System Requirements Specification**

It contains all hardware and software requirements which we are using in our project.

- **Chapter 4: System Analysis and Design**

In this chapter we have described both the approaches for preparing our model i.e, SVM and GLCM.

- **Chapter 5: Implementation**

This chapter contains the code part and phases we are used.

- **Chapter 6: Results and Discussion**

This chapter contains the results we observed and obtained from the project.

- **Chapter 7: Testing**

This chapter contains the various testing methods used to test the project.

- **Chapter 8: Conclusion**

This chapter contains various conclusion drawn from our work.

Chapter 2

LITERATURE SURVEY

There are various papers published on prediction of haze detection using various methods. We have reviewed four papers and have drawn the following conclusions.

[1] Detecting Foggy Images and Estimating the Haze Degree Factor

This paper proposes Limited visibility in haze weather strongly influences the accuracy and the general functions of almost outdoor video surveillance or driver assistance systems. Actual weather condition is valuable information to invoke corresponding approaches. Based on the atmospheric scattering model analysis and the statistics of various outdoor images, for most foggy images, we find that the lowest and highest value in color channels tends to be the same value of atmospheric light. A function for estimating the haze degree is developed for the automatic detection of the foggy image with different haze degrees. Experimental results show that our haze classification method achieves high performance.

At present, most outdoor video-surveillance, driver-assistance and optical remote sensing systems have been designed to work under good visibility and weather conditions. Poor visibility often occurs in foggy or hazy weather conditions and can strongly influence the accuracy or even the general functionality of such vision systems. Based on the hazy weather classification, specialized approaches, such as a dehazing process, can be employed to improve recognition .Another proposed system can automatically label images with high confidence by assigning weather labels, such as sunny, or cloudy; however, manual input constraints are required. Against this background, the main aim of the current study is to develop a set of stable algorithms for the detecting foggy images and labeling the haze degree of images by using a factor with universal applications. In this paper, we propose a haze degree estimation function to automatically distinguish foggy images and label images with their corresponding haze degrees. We relied on the atmospheric scattering model analysis and statistics derived from various outdoor images in order to develop the estimation function.

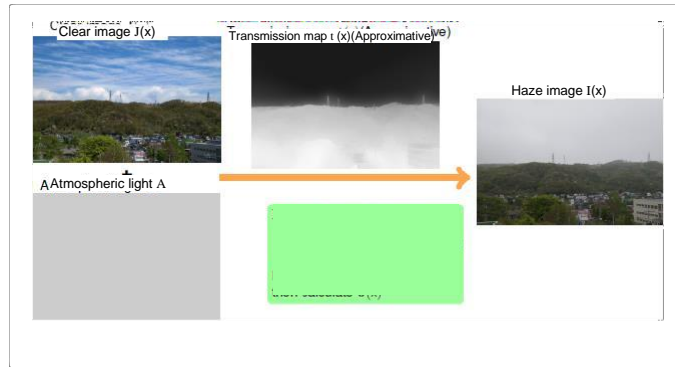


Fig 2.1: Atmospheric Scattering Model

[2] MULTI-CLASS WEATHER CLASSIFICATION ON SINGLE IMAGES.

Multi-class weather classification from single images is a fundamental operation in many outdoor computer vision applications. However, it remains difficult and the limited work is carried out for addressing the difficulty. Moreover, existing method is based on the fixed scene. In this paper we present a method for any scenario multi-class weather classification based on multiple weather features and multiple kernel learning. Our approach extracts multiple weather features and takes properly processing. By combining these features into high dimensional vectors, we utilize multiple kernel learning to learn an adaptive classifier. We collect an outdoor image set that contains 20K images called MWI (Multiclass Weather Image) set. Experimental results show that the proposed method can efficiently recognize weather on MWI dataset.

However, different weather conditions such as rain, snow or haze will cause complex visual effects in images or videos. Such effects may significantly degrade the performances of outdoor vision systems relying on image/video feature extraction or visual attention modeling. The applications are numerous, such as the detection of critical weather conditions, the observation of weather, the reliability improvement of video surveillance systems and rain or snow rendering. In this paper, we target at the problem of classifying multiple weather, such as sunny, rainy, snowy and haze from single images.



Fig 2.2: Square in different Weather

[3] The Effects of Haze on the Accuracy of Maximum Likelihood Classification Copy

This study aims to investigate the effects of haze on the accuracy of Maximum Likelihood classification. Data containing eleven land covers recorded from Landsat 5 TM satellite were used. Two ways of selecting training pixels were considered which are choosing from the haze-affected and haze-free data. The accuracy of Maximum Likelihood classification was computed based on confusion matrices where the accuracy of the individual classes and the overall accuracy were determined. The result of the study shows that classification accuracies declines with faster rate as visibility gets poorer when using training pixels from clear compared to hazy data.

One of the commonly used classification scheme for such purpose is Maximum Likelihood classification (ML), a supervised classification method. The performance of ML classification is very much influenced by the selection of the pixels used to train the ML classifier, or the so called training pixels. In investigating the effects of haze on land cover classification, there are two ways of selecting the training pixels that are to be fed to the ML classifier. The first option is selecting training pixels from the data that are affected by haze (i.e. hazy data) while the second option is selecting training pixels from the data that are free from haze (i.e. clear data).

[4] Two-Class Weather Classification

Given a single outdoor image, this paper proposes a collaborative learning approach for labeling it as either sunny or cloudy. This two class classification problem is by no means trivial given the great variety of outdoor images. Our weather feature combines special cues after properly encoding them into feature vectors. They then work collaboratively in synergy under a unified optimization framework that is aware of the presence (or absence) of a given

weather cue during learning and classification. Extensive experiments and comparisons are performed to verify our method. We build a new weather image dataset consisting of 10K sunny and cloudy images, which is available online together with the executable. Rain and sunshine make up our everyday weather experience. The weather affects our daily lives in many ways, from solar technologies, outdoor sporting events, to the sort of clothes we wear and whether to stay indoors or not on weekend. While current accurate weather detection technologies rely on expensive sensors, for centuries weather observing tools consisted of the human eye (and various human senses as well).

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

The chapter describes about the requirements. It specifies the hardware and software requirements that are in order to run the application properly. The Software Requirement Specification (SRS) is explained in detail, which includes the overview of functional and non-functional requirements.

An SRS document describes the functional and behavioural requirements of the software under development. SRS is a fundamental document which forms the foundation of the software development process. It is the complete description of the behaviour of a system to be developed it is not only the list of requirements of a system but also has a description of its major features. Requirement Analysis in system engineering and software engineering encompasses those tasks that go into determining the need or conditions to meet for a new or altered product.

The SRS functions as a blueprint for completing a project. It is important to note that an SRS contains functional and non-functional requirements.

Thus, the goal of preparing the SRS document is to

- To firm foundation for the design phase
- Support system testing facilities
- Support project management and control
- Controlling the Evolution of system

3.1 Functional Requirements

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs/or conditions. These may include calculations, data manipulation and processing and other specific functionality. In these systems following are the functional requirements

- Input test case must not have compilation and runtime errors.
- The application must not stop working when kept running for even a longtime.
- The application should generate the output for a given input test case.

3.2 Non-Functional Requirements

Non-functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviours.

Given below are the non-functional requirements:

- Product requirements
- Organizational requirements
- Basic operational requirements

3.3 Hardware Requirements

- RAM-8GB
- Processor-Intel
- HardDisk-1TB
- Mouse
- Keyboard
- Monitor

3.4 Software Requirements

- Operating System: windows
- Programming Language: Python
- Additional platform: Google Cloudera
- Database: Frida2

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN

System Analysis is the process that decomposes a system into its component pieces for the purpose of defining how well those components interact to accomplish the set requirements. Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. System design is the process of defining and developing systems to satisfy specified requirements of the user. The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

Elements of a System

- **Architecture** - This is the conceptual model that defines the structure, behavior and more views of a system. We can use flowchart to represent and illustrate the architecture.
- **Modules** - These are components that handle one specific task in a system. A combination of the modules make up the system.
- **Components** - This provides a particular function or group of related functions. They are made up of modules.
- **Interfaces** - This is the shared boundary across which the components of a system exchange information and relate.
- **Data** - This is the management of the information and data flow.

4.1 System Architecture

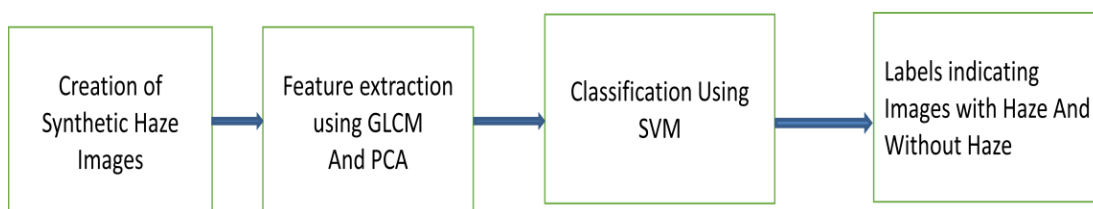


Fig: 4.1 System Architecture

Step 1:

To create a data or to collect a lot from the real world is very complex and expensive because we need to collect data with same resolution, same camera, same light. So we are using a Frida2 dataset which contains both the haze images and non free haze images

Step 2:

For this collected Frida2 database we are applying feature extraction using GLCM (Grey Level Co-Occurrence Matrix) and PCA (Principal Component Analysis). So that GLCM will extract the unique features and after the extracted unique features we are applying PCA to reduce the complexity and dimension.

Step 3:

Extracted data is sent to the training model to classify the data by using SVM.

Step 4:

Classified data contains the haze images and non free haze images.

4.2 Dataset

A **data set** (or **dataset**) is a collection of data. In the case of tabular data, a data set corresponds to one or more database tables, where every column of a table represents a particular variable, and each row corresponds to a given record of the data set in question. The data set lists values for each of the variables, such as height and weight of an object, for each member of the data set. Each value is known as a datum. Data sets can also consist of a collection of documents or files.

FRIDA2 are databases of numerical synthetic images easily usable to evaluate in a systematic way the performance of visibility and contrast restoration algorithms. Frida2 dataset in which the haze is synthetically equally distributed on all the random parts of the image.

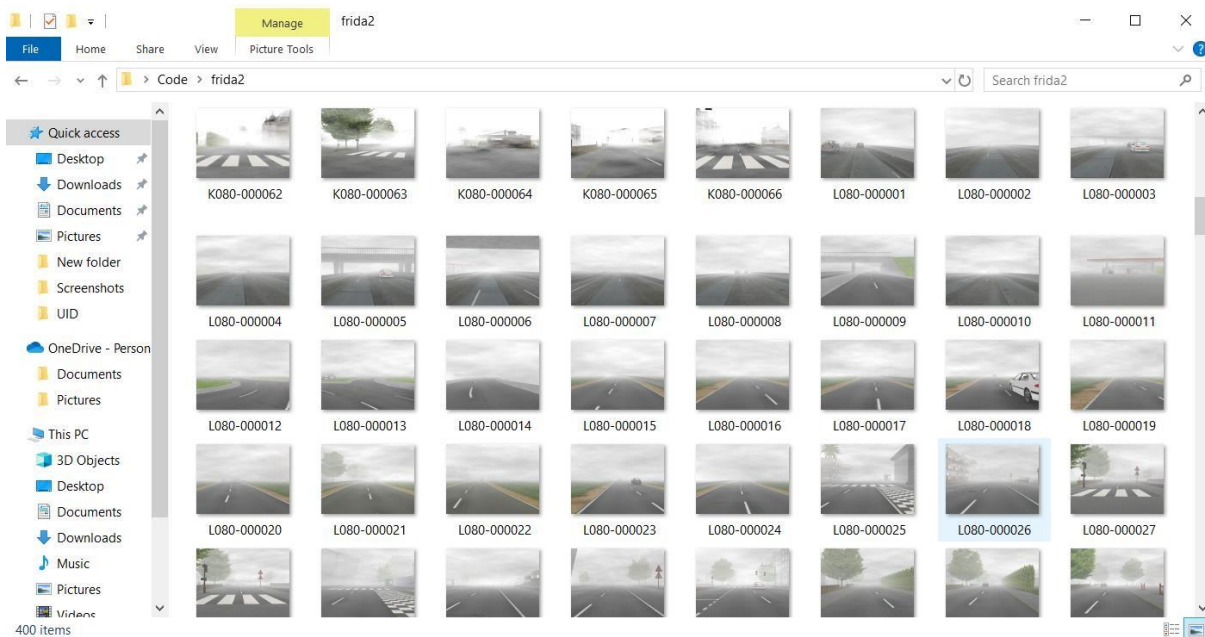


Fig 4.2 Datasets

4.3 Data Flow of Use Case

Dataflow is a fully managed streaming analytics service that minimizes latency, processing time, and cost through auto scaling and batch processing.

- In order to verify the effectiveness of our proposed methodology we are applying GLCM feature extraction on frida2 synthetic database.
- And it is used for extracting haze images and haze-free images.
- Haze images are indicated by 1.
- Haze free images are indicated by 0.

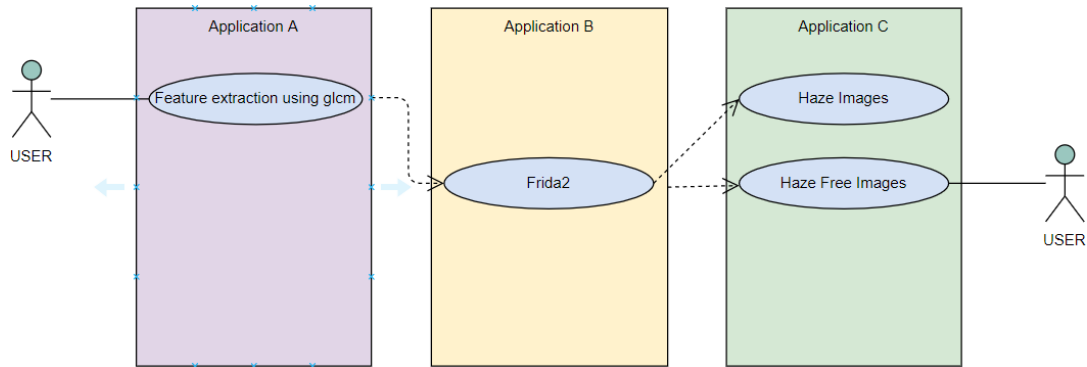


Fig: 4.3 Use Case Diagram

4.4 Data Flow of Class Diagram

- A class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, and the relationships between the classes.
- The class diagram has the following classes.
- FRIDA2: This is user interface class which has function for invoking train and predict.
- Feature extraction: This class has functions for extracting the features from the dataset and making it suitable for GLCM.
- GLCM: This function predicts whether it is Haze or Haze free images by taking output of the feature extraction as its input.

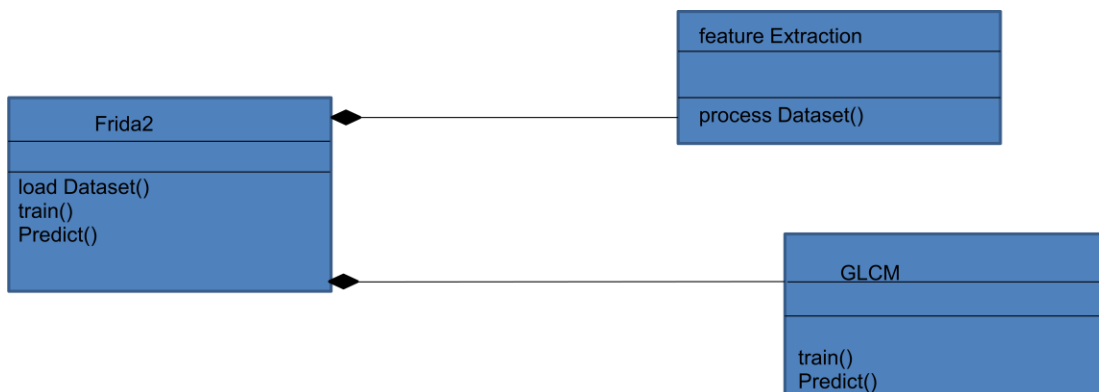


Fig 4.4: Class Diagram

4.5 Data flow of Sequence Diagram

- A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart.
- As shown in the sequence diagram GLCM invokes frida2 dataset to extract whether it is a haze images or haze free images.

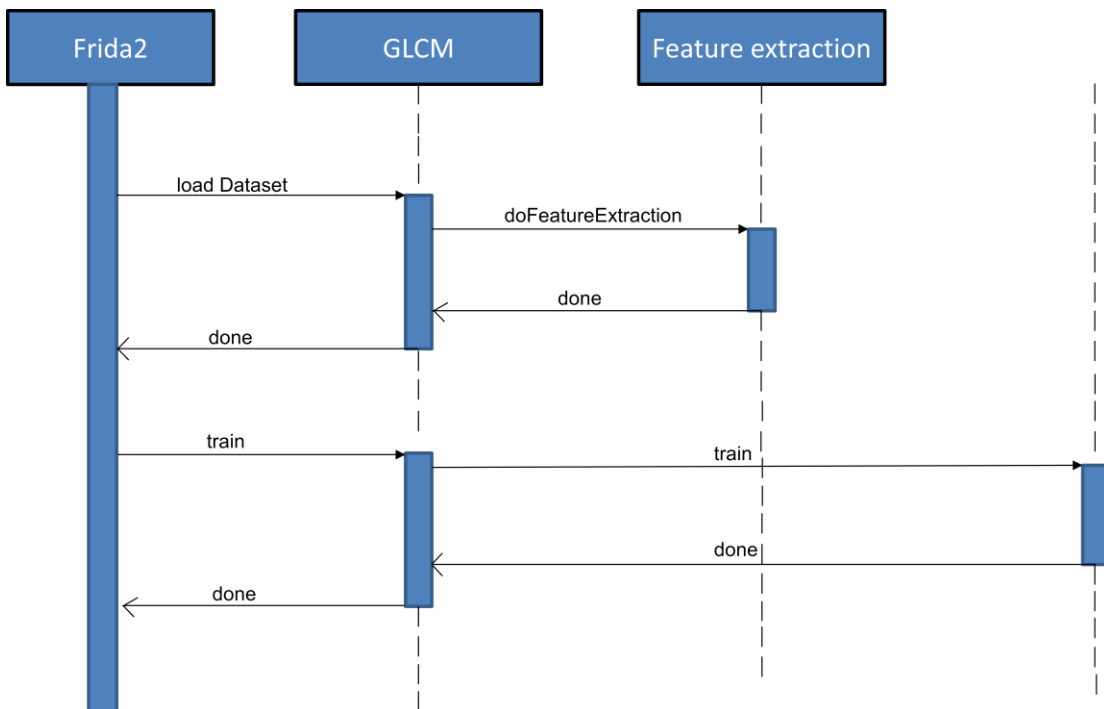


Fig 4.5: Sequence Diagram

CHAPTER 5

IMPLEMENTATION

5.1 Gray Level Co-Occurrence Matrix (GLCM)

Gray Level Co-Occurrence Matrix (GLCM): For feature extraction GLCM is useful for Haze classification since it looks for derivatives in the neighborhood of the current pixels. The **GLCM** functions characterize the texture of an image by calculating how often pairs of pixel with specific values and in a specified spatial relationship occur in an image, creating a **GLCM**, and then **extracting** statistical measures from this matrix.

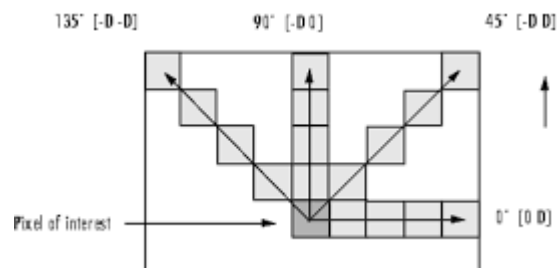


Fig 5.1: GLCM

EXTRACTION OF TEXTURE FEATURES OF IMAGE Gray Level Co-Occurrence Matrix (GLCM)

1. Angular Second Moment

Angular Second Moment is also known as Uniformity or Energy. It is the sum of squares of entries in the GLCM. Angular Second Moment measures the image homogeneity. Angular Second Moment is high when image has very good homogeneity or when pixels are very similar. In case of haze it is highly likely to have good homogeneity.

2. Inverse Difference Moment

Inverse Difference Moment (IDM) is the local homogeneity. It is high when local gray level is uniform and inverse GLCM is high.

3. Correlation

Correlation measures the linear dependency of grey levels of neighboring pixels. Digital Image Correlation is an optical method that employs tracking & image registration techniques for accurate 2D and 3D measurements of changes in images. This is often used to measure deformation, displacement, strain and optical flow, but it is widely applied in many areas of science and engineering. One very common application is for measuring the motion of an optical mouse.

5.2 Principal Component Analysis(PCA)

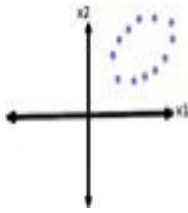
Principal Component Analysis (PCA) is a dimensionality-reduction technique that is often used to transform a high-dimensional dataset into a smaller-dimensional subspace prior to running a machine learning algorithm on the data.

It is often helpful to use a dimensionality-reduction technique such as PCA prior to performing machine learning because:

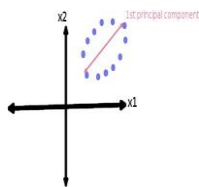
1. Reducing the dimensionality of the dataset reduces the size of the space on which k-nearest-neighbors (k\KNN) must calculate distance, which improve the performance of KNN.
2. Reducing the dimensionality of the dataset reduces the number of degrees of freedom of the hypothesis, which reduces the risk of over fitting.
3. Most algorithms will run significantly faster if they have fewer dimensions they need to look at.
4. Reducing the dimensionality via PCA can simplify the dataset, facilitating description, visualization, and insight.

Principal Component Analysis does just what it advertises; it finds the principal components of the dataset. PCA transforms the data into a new, lower-dimensional subspace-into a new coordinate system. In the new coordinate system, the first axis corresponds to the first principal component, which is the component that explains the greatest amount of the variance in the data.

1. Let's say your original dataset has two variables, x_1 and x_2 :



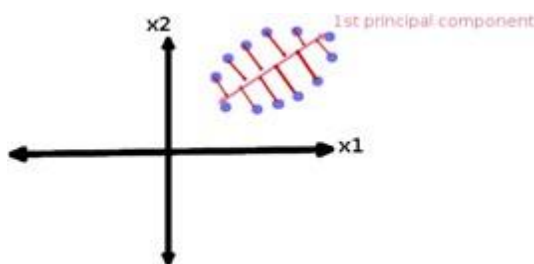
2. Now, we want to identify the first principal component that has explains the highest amount of variance. Graphically, if we draw a line that splits the oval lengthwise, that line signifies the component that explains the most variance:



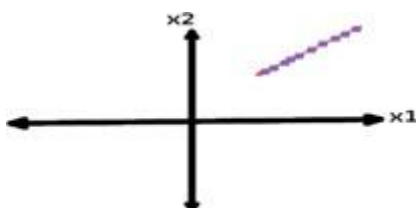
3. Let's say we just wanted to project the data onto the first principal component only. In other words, we wanted to use PCA to reduce our two-dimensional dataset onto a one-dimensional dataset.

4. Basically, we would collapse our dataset onto a single line (by projecting it onto that line). The single line is the first principal component.

Here is a picture:



5. Here is what the dataset will look like after being projected onto a single dimension corresponding to the first principal component:

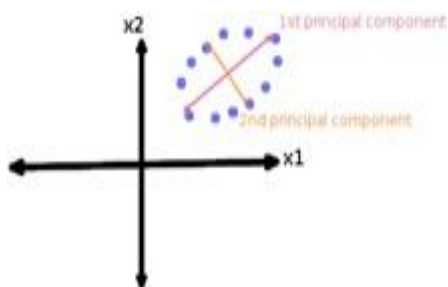


6. We can see that we have destroyed some of the original information when we went from a two-dimensional dataset to the one-dimensional projection.
7. You can think of this sort of like a shadow
8. Although we lost some information in the transformation, we did keep the most important axis, which incorporates information from both x_1 and x_2 .

Second principal component

1. The second principal component must be orthogonal to the first principal component. In otherworld's, it does its best to capture the variance in the data that is not captured by the first principal component. For our two-dimensional dataset, there can be only two principal components.

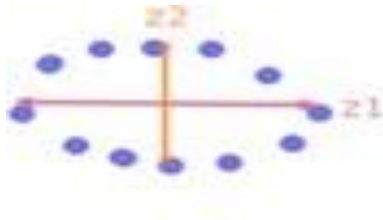
2. Here is a picture of the data and its first and second principal components:



3. Again, you can see that the two principal components are perpendicular to each other. They capture independent elements of the dataset.

If we were to perform PCA on this dataset and project the original dataset onto the first two principal components, then no information would be lost. (We are transforming from a two-dimensional dataset to a new two-dimensional dataset.) Instead, we would be merely rotating the data to use new dimensions.

1. Here is a picture of what the dataset looks like projected onto the first two principal components:



In general, the data will tend to follow the 80/20 rule. Most of the variance (interesting part of data) will be explained by a very small number of principal components. You might be able to explain 95% of the variance in your dataset using only 10% of the original number of attributes. However, this is entirely dependent on the dataset. Often, a good rule of thumb is to identify the principal components that explain 99% of the variance in the data. You cannot have more principal components than the number of attributes in the original dataset.

5.3 Supported Vector Machine (SVM)

Supported Vector Machine (SVM): It is suitable for our project since the classification is based on the energy and correlation of the images.

It is a supervised learning algorithm. It is mostly used for classification problems. Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. A schematic example is shown in the illustration below. In this example, the objects belong either to class GREEN or RED. The separating line defines a boundary on the right side of which all objects are GREEN and to the left of which all objects are RED. Any new object (white circle) falling to the right is labeled, i.e., classified, as GREEN (or classified as RED should it fall to the left of the separating line).

There are two types of Classifiers:

1. Linear SVM classifier
2. Non linear SVM classifier

Linear SVM: In linear SVM the data points are separated by an apparent gap. It predicts a straight "hyper plane" dividing by 2 classes. For Drawing the hyper plane is maximizing the distance from hype plane to the nearest data point of either class. The hyper plane is called as a "maximum margin hyper plane".

Non linear SVM: In non linear SVM data points plotted in a higher dimensional space. Here kernel trick is used to maximum margin hyper plane.

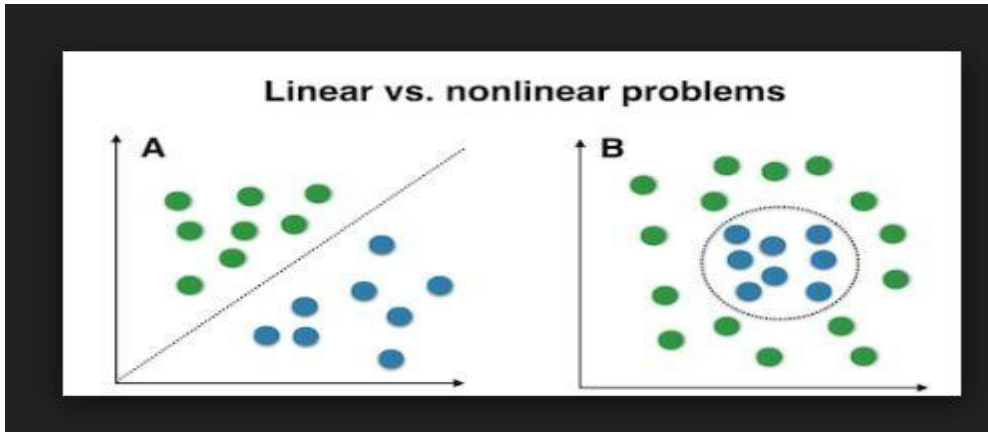
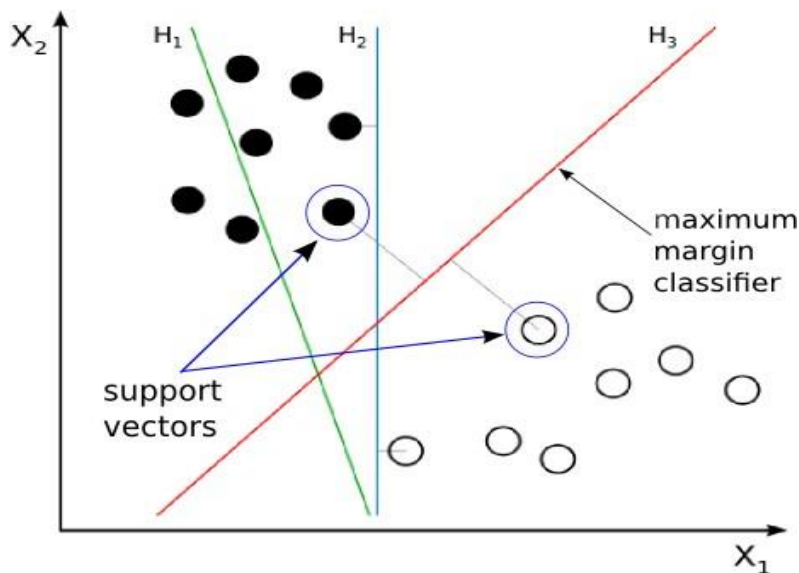


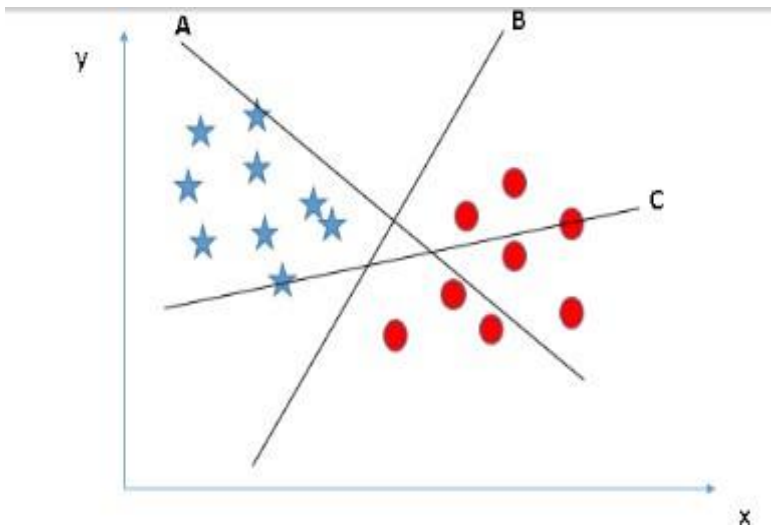
Fig 5.2 Liner v/s nonlinear

Margin is defined as the distance between the separating hyper plane (decision boundary) .

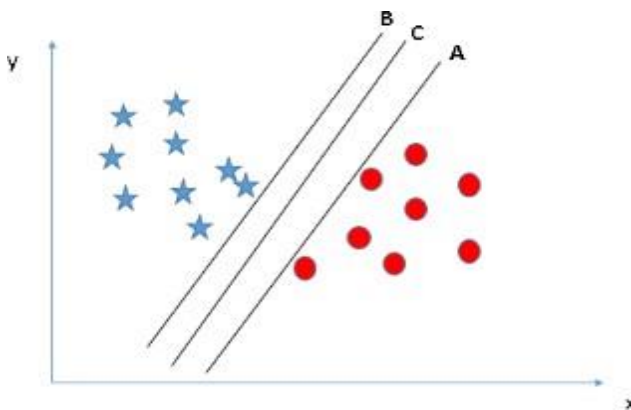


Here H1 does not separate the classes. H2 does, but only with a small margin. H3 separates them with the maximum margin.

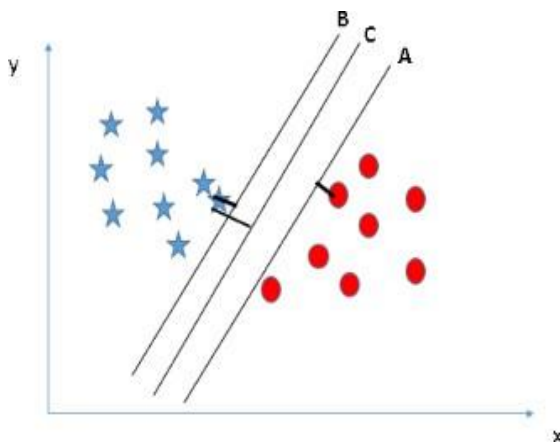
1. Identify the right hyper-plane (Scenario-1): Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.



2. Identify the right hyper-plane (Scenario-2): Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, how can we identify the right hyper-plane?

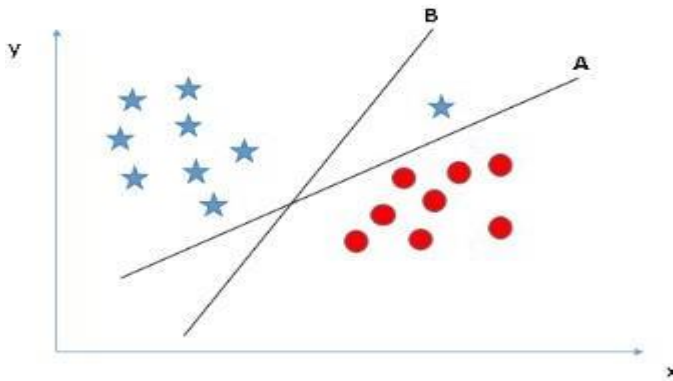


Here to find the maximum distance between the nearest data points for each class.

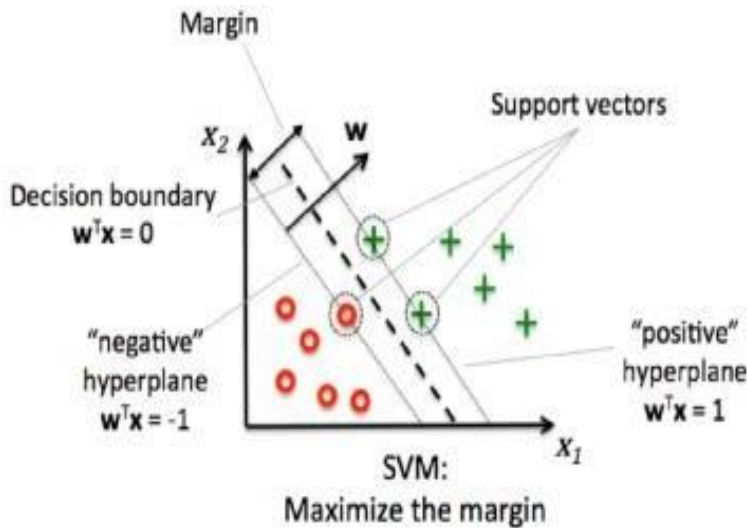


C is high as compared to both A and B. Hence, the right hyper-plane as C.

3. Identify the right hyper-plane(Scenario-3):



Here hyper-plane B as higher margin compared to A. But, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is A.



SVM needs only the support vectors to classify any new data instances, it is quite efficient. In other words, it uses a subset of training points in the decision function (support vectors), so it is also memory efficient.

Kernel method:

The kernel methods is to deal with linearly inseparable data, and to create nonlinear

combinations of the original features to project them onto a higher dimensional space via a mapping function $\phi ()$ where it becomes linearly separable.

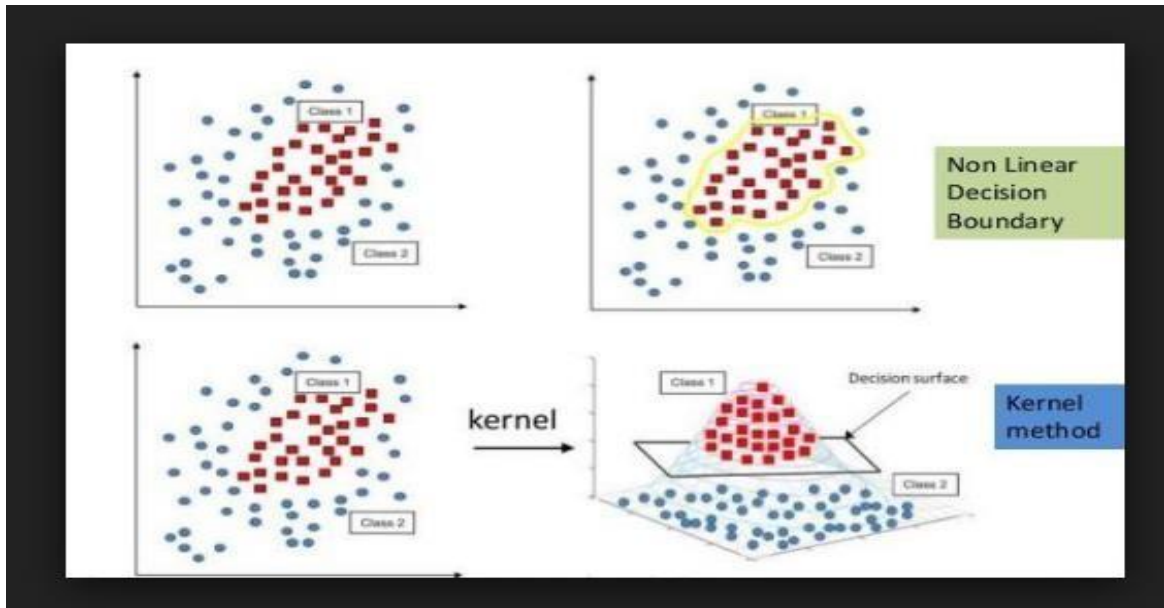


Fig 5.3: Kernel

We can transform a two-dimensional dataset onto a new three-dimensional feature space where the classes become separable via the following projection.

$$\phi(x_1, x_2) = (z_1, z_2, z_3) = (x_1, x_2, x_1^2 + x_2^2)$$

The mapping allows us to separate the two classes shown in the plot via a linear hyper plane that becomes a nonlinear decision boundary if we project it back (ϕ^{-1}) onto the original feature space.

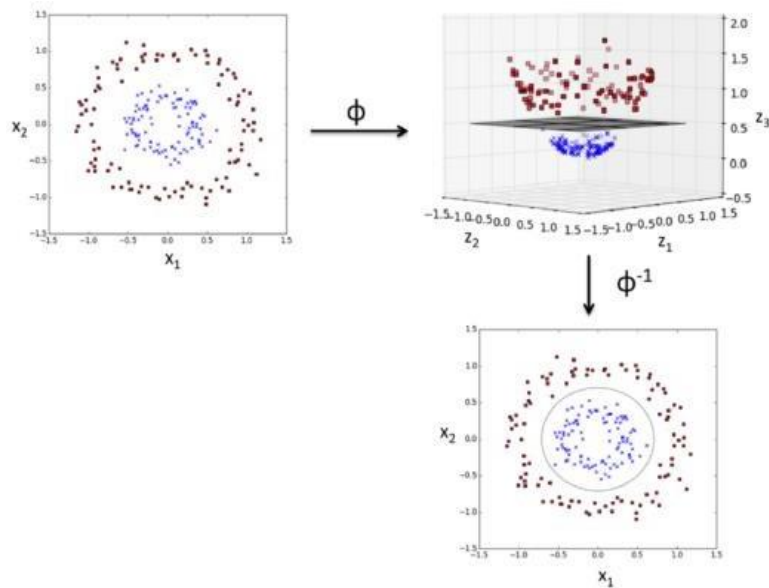


Fig 5.4: Mapping

Finding separating hyper planes via kernel trick:

To solve a nonlinear problem with SVM: We transform the training data onto a higher dimensional feature space via a mapping function $\phi()$.

1. We transform the training data onto a higher dimensional feature space via a mapping function $\phi()$.
2. We train a linear SVM model to classify the data in this new feature space.
3. Then, we can use the same mapping function $\phi()$ to transform unseen data to classify it using the linear SVM model.
4. The kernel trick avoids the explicit mapping that is needed to get linear learning algorithms to learn a nonlinear function or decision boundary.
5. To train an SVM, in practice, all we need is to replace the dot product $x^{(i)T}x^{(j)}$ by $\phi(x^{(i)})^T\phi(x^{(j)})$.
6. In order to avoid the expensive step of calculating this dot product between two points explicitly, we define a so-called **kernel function**.

$$k\left(x^{(i)}, x^{(j)}\right) = \phi\left(x^{(i)}\right)^T \phi\left(x^{(j)}\right)$$

Importing all the modules

```
import pickle
import glob
import cv2
import os
import numpy as np
from sklearn import svm

from skimage import io, color, img_as_ubyte

from skimage.feature import greycomatrix, greycoprops
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
```

Reading the frida2dataset

```
if name__ == 'main':

    if not os.path.exists('haze_images.pk'):
        os.chdir('frida2')
```

Extraction of texture features of image Gray Level Co-Occurrence Matrix (GLCM)

Gray Level Co-Occurrence Matrix (GLCM): For feature extraction GLCM is useful for Haze classification since it looks for derivatives in the neighborhood of the current pixels.

```
for image_path in glob.glob("*.png"):
    print('Reading image: ', image_path)
    rgb = cv2.imread(image_path)

    grayImg = img_as_ubyte(color.rgb2gray(rgb))
    distances = [1, 2, 3]
    angles = [0, np.pi/4, np.pi/2, 3*np.pi/4]
    properties = ['energy', 'correlation', 'homogeneity']
    glcm = greycomatrix(grayImg, distances=distances, angles=angles, symmetric=True,
normed=True)
```

Converting all the rows into columns by using ravel()

```
feats = np.hstack([greycoprops(glcm, prop).ravel() for prop in properties])  
image.append(feats.ravel())
```

Identifying all the images

whether it is a heterogeneous images or homogeneous or cloudy heterogeneous images or cloudy homogeneous images or haze free images.

```
if ('M080' in os.path.basename(image_path)):  
    labs.extend([1])  
elif ('U080' in os.path.basename(image_path)):  
    labs.extend([1])  
elif ('K080' in os.path.basename(image_path)):  
    labs.extend([1])  
elif ('L080' in os.path.basename(image_path)):  
    labs.extend([1])  
elif ('LIma' in os.path.basename(image_path)):  
    labs.extend([0])
```

PCA(Principal Components Analysis)

PCA is applied to the extracted GLCM features where it reduces the number of features by eliminating the features which are not useful from the GLCM and reduces the dimension. PCA transforms the data into a new, lower-dimensional subspace.

```
pca = PCA(n_components=4)  
X = _pca.fit_transform(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, labs, test_size=0.1, random_state=4)
```

Supported Vector Machine (SVM)

SVM since the classification is based on the energy and correlation of the images, To train the model.

```
model = svm.SVC(kernel='rbf')  
model.fit(X_train, y_train) #Model training  
labs_pred = model.predict(X_test)
```

The accuracy of the test data and train data of the trained model

```
print("\n\nTrain score:', round(model.score(X_train, y_train),2))  
print("Test score:',round( model.score(X_test, y_test),2))
```

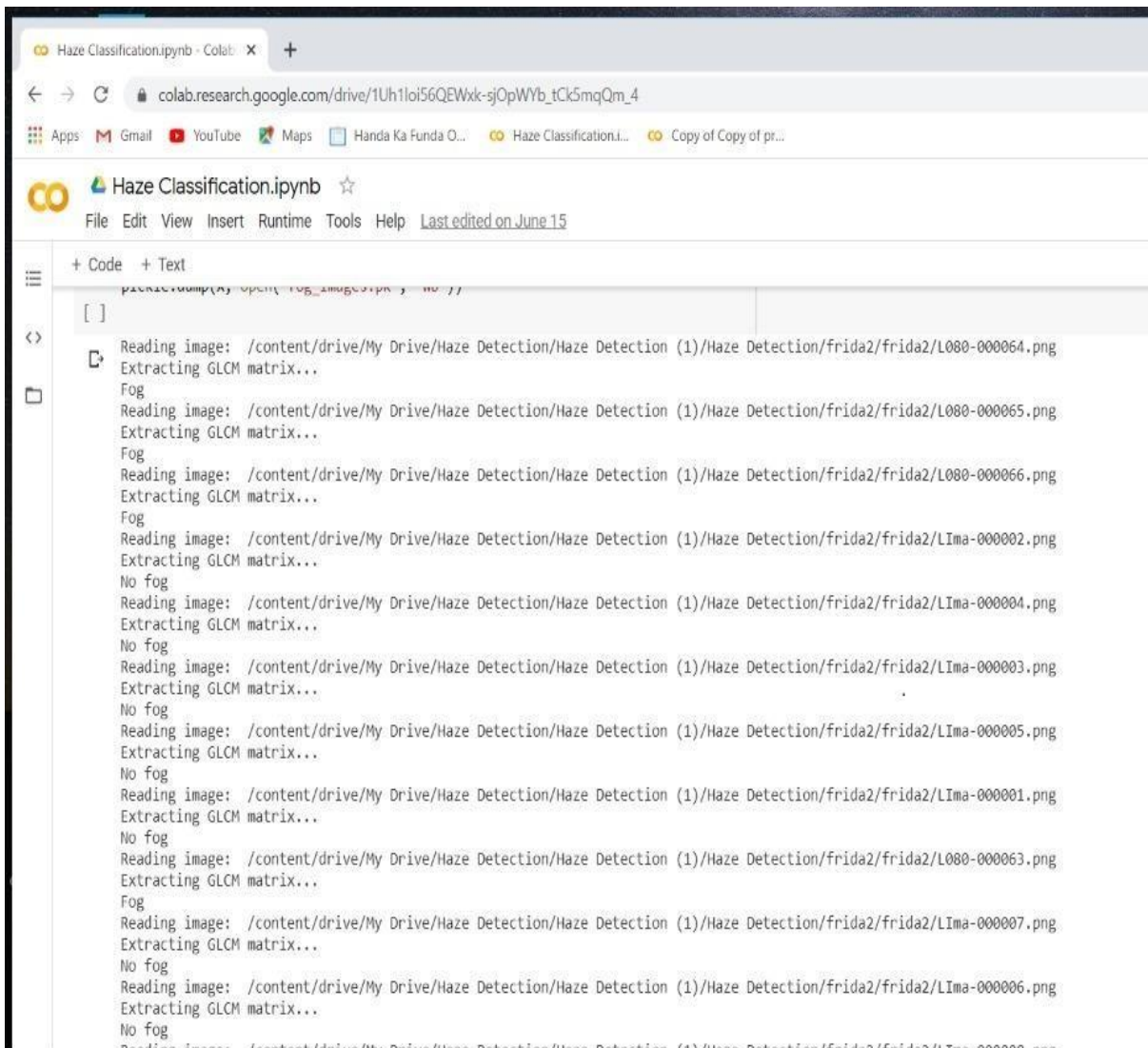
Passing the single image for classification to find out haze image or non free haze image

```
Name='frida2/K080-000010.png'  
Println("\n\nPrediction for individual image:',name,""\tPrediction:',test(name))
```

CHAPTER 6

RESULTS AND DISCUSSION

6.1 Training model for Haze



```

+ Code + Text
[ ]
Reading image: /content/drive/My Drive/Haze Detection/Haze Detection (1)/Haze Detection/frida2/frida2/L080-000064.png
Extracting GLCM matrix...
Fog
Reading image: /content/drive/My Drive/Haze Detection/Haze Detection (1)/Haze Detection/frida2/frida2/L080-000065.png
Extracting GLCM matrix...
Fog
Reading image: /content/drive/My Drive/Haze Detection/Haze Detection (1)/Haze Detection/frida2/frida2/L080-000066.png
Extracting GLCM matrix...
Fog
Reading image: /content/drive/My Drive/Haze Detection/Haze Detection (1)/Haze Detection/frida2/frida2/L1ma-000002.png
Extracting GLCM matrix...
No fog
Reading image: /content/drive/My Drive/Haze Detection/Haze Detection (1)/Haze Detection/frida2/frida2/L1ma-000004.png
Extracting GLCM matrix...
No fog
Reading image: /content/drive/My Drive/Haze Detection/Haze Detection (1)/Haze Detection/frida2/frida2/L1ma-000003.png
Extracting GLCM matrix...
No fog
Reading image: /content/drive/My Drive/Haze Detection/Haze Detection (1)/Haze Detection/frida2/frida2/L1ma-000005.png
Extracting GLCM matrix...
No fog
Reading image: /content/drive/My Drive/Haze Detection/Haze Detection (1)/Haze Detection/frida2/frida2/L1ma-000001.png
Extracting GLCM matrix...
No fog
Reading image: /content/drive/My Drive/Haze Detection/Haze Detection (1)/Haze Detection/frida2/frida2/L080-000063.png
Extracting GLCM matrix...
Fog
Reading image: /content/drive/My Drive/Haze Detection/Haze Detection (1)/Haze Detection/frida2/frida2/L1ma-000007.png
Extracting GLCM matrix...
No fog
Reading image: /content/drive/My Drive/Haze Detection/Haze Detection (1)/Haze Detection/frida2/frida2/L1ma-000006.png
Extracting GLCM matrix...
No fog
Reading image: /content/drive/My Drive/Haze Detection/Haze Detection (1)/Haze Detection/frida2/frida2/L1ma-000008.png
Extracting GLCM matrix...
No fog

```

Fig 6.1 Training Model for Haze

6.2 Classifying the images

Classifying the images based on the labels of image whether it is a haze image or haze free image.

Cloudy Homogeneous Image



Fig 6.2: K080-000009

Cloudy Heterogeneous Image



Fig 6.3: M080-000050

Homogeneous Image



Fig 6.4: U080-000042

Heterogeneous Image



Fig 6.5: L080-000040

Haze Free Image



Fig 6.6: LIMA-000009

When the images with M080 (Cloudy Heterogeneous image), L080 (Heterogeneous image), K080 (Cloudy Homogeneous image), U080 (Homogeneous image) is detected, it is classified as Haze images and LIMA (Haze free image) is detected then it is classified as Haze free images by GLCM and PCA.

When the single image (K080-000010) is passed to the trained model it predicts the haze image.

6.3 Accuracy

This data is trained by SVM classify the model. Then the trained data 90% and test data 10% is passed to model to test the accuracy. In this output the training data of accuracy 96% and the testing data of 91% accuracy is obtained.

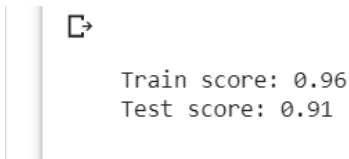


Fig 6.7: Overall Accuracy

6.4 Results-Haze

Single Image classification

```

# For single image classification

def test(image_path):
    classes = ['No haze', 'Hazy']
    test_image = []
    X = []
    rgb = cv2.imread(image_path)

    grayImg = img_as_ubyte(color.rgb2gray(rgb))

    distances = [1, 2, 3]
    angles = [0, np.pi/4, np.pi/2, 3*np.pi/4]
    properties = ['energy', 'correlation', 'homogeneity']

    glcm = greycomatrix(grayImg,
                        distances=distances,
                        angles=angles,
                        symmetric=True,
                        normed=True)

    feats = np.hstack([greycomprops(glcm, prop).ravel() for prop in properties])

    test_image.append(feats)
    X = np.asarray(test_image)

    X = X.reshape(X.shape[0], -1)
    X_test = _pca.transform(X)
    labs_test = model.predict(X_test)
    return classes[ labs_test[0] ]

name = 'frida2/K000-000010.png'
print('\n\nPrediction for individual image:', name, '\tPrediction:', test(name))
    
```

Fig 6.8: Sample Image: Testing for Haze



Fig 6.9: Single image output after Testing

CHAPTER 7

TESTING

7.1 Software testing introduction

Software testing is a process used to help identify the correctness, completeness and quality of developed computer software. Software testing is the process used to measure the quality of developed software. Testing is the process of executing a program with the intent of finding errors. Software testing is often referred to as verification and validation.

7.2 STLC (Software Testing Life Cycle)

Testing itself has many phases i.e. is called as STLC. STLC is part of SDLC

- Test Plan
- Test Development
- Test Execution
- Analyze Result

7.2.1 Test Plan

It is a document which describes the testing environment, purpose, scope, objectives, test strategy, schedules, mile stones, testing tool, roles and responsibilities, risks, training, staffing and who is going to test the application, what type of tests should be performed and how it will track the defects.

7.2.2 Test Development

Preparing test cases, test data, preparing test procedure, Preparing test scenario, Writing test script.

7.2.3 Test Execution

In this phase we execute the documents those are prepared in test development phase.

The frida2 database is first tested for saving the labels and the images for future use.

```
Shape of label array: (330,)  
Shape of preprocessed image array: (330, 36)  
Saving the image and label array for future use  
[ ]
```

Fig 7.1 label and image

7.2.4 Analyze Result

Once executed documents with get results either haze or haze free image. We need to analyze the results during this phase

We tested the dataset and found out which is haze images and non haze free images indicated as hazy and non hazy.

We calculated the accuracy of the predicted output.

7.3 TYPES OF TESTING:

- White Box Testing
- Black Box Testing
- Grey box testing

7.3.1 White Box Testing

White box testing as the name suggests gives the internal view of the software. This type of testing is also known as structural testing or glass box testing as well, as the interest lies in what lies inside the box.

7.3.2 Black Box Testing

Its also called as behavioral testing. It focuses on the functional requirements of the software. Testing either functional or non functional without reference to the internal structure of the component or system is called black box testing.

7.3.3 Grey Box Testing

Grey box testing is the combination of black box and white box testing. Intention of this testing is to find out defects related to bad design or bad implementation of the system.

7.4 LEVEL OF TESTING USED IN PROJECT

7.4.1 Unit testing

Initialization testing is the first level of dynamic testing and is the first responsibility of developers and then that of the test engineers. Unit testing is performed after the expected test results are met or differences are explainable/acceptable.

7.4.2 Integration testing

All module which make application are tested . Integration testing is to make sure that the interaction of two or more components produces results that satisfy functional requirement.

7.4.3 System testing

To test the complete system in terms of functionality and non functionality. It is black box testing, performed by the Test Team, and at the start of the system testing the complete system is configured in a controlled environment.

7.4.4 Functional testing

The outgoing links from all the pages from specific domain under test. Test all internal links. Test links jumping on the same pages. Check for the default values of fields. Wrong inputs to the fields in the forms.

7.4.5 Alpha testing

Alpha testing is final testing before the software is released to the general public. This testing is conducted at the developer site and in a controlled environment by the end user of the software.

7.4.6 Beta testing

The beta test is conducted at one or more customer sites by the end user of the software. The beta test is conducted at one or more customer sites by the end user of the software

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

It can be concluded that using our proposed methodology for preparing the training database. We are going to create an synthetic database and applying feature extraction to classify by SVM Whether it is haze images or non free haze images.

8.2 Contribution

By using our proposed methodology we can classify the images into haze and haze free images which further reduces the complexity of haze removal from the images. Since visibility plays a more important role than relative humidity in affecting the detection accuracy.

8.3 Future Scope

The project gives barrier-free access to the literature of research. It increases convenience, reach and retrieval power. This puts rich and poor on an equal footing for these key resources.

Training the model whether it is a haze images or non free haze images .

To remove haze we can apply filters like Gaussian filter, Box filter, Average filter .

REFERENCES

In IEEE style <https://www.sciencedirect.com/science/article/pii/S2001037014000464> Papers:

- [1] Z. Xu, X. Liu, and N. Ji, “Fog Removal from Color Images using Contrast Limited Adaptive Histogram Equalization,” in 2009 2nd International Congress on Image and Signal Processing, 2009, pp.1–5.
- [2] R. T. Tan, “Visibility in bad weather from a single image,” in 2008 IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp.1–8.
- [3] Q. Zhu, J. Mai, and L. Shao, “A Fast Single Image Haze Removal Algorithm Using Color Attenuation Prior,” IEEE Transactions on Image Processing, vol. 24, no. 11, pp. 3522–3533, Nov. 2015.
- [4] B. Cai, X. Xu, K. Jia, C. Qing, and D. Tao, “DehazeNet: An End-to-End System for Single Image Haze Removal,” IEEE Transactions on Image Processing, vol. 25, no. 11, pp. 5187–5198, Nov.2016.
- [5] P. Louridas and C. Ebert, “Machine Learning,” IEEE Software, vol. 33, no. 5, pp. 110–115, Sep.2016.
- [6] K. Ma, W. Liu, and Z. Wang, “Perceptual evaluation of single image dehazing algorithms,” in 2015 IEEE International Conference on Image Processing (ICIP), 2015, pp. 3600–3604.
- [7] J. P. Tarel, N. Hautiere, L. Caraffa, A. Cord, H. Halmaoui, and D. Gruyer, “Vision Enhancement in Homogeneous and Heterogeneous Fog,” IEEE Intelligent Transportation Systems Magazine, vol. 4, no. 2, pp. 6–20, Summer2012.
- [8] N. Hautière, J.-P. Tarel, D. Aubert, and É. Dumont, “BLIND CONTRAST ENHANCEMENT ASSESSMENT BY GRADIENT RATIOING AT VISIBLE EDGES,” Image Analysis & Stereology, vol. 27, no. 2, pp. 87–95, May2011.
- [9] H. Yeganeh and Z. Wang, “Objective Quality Assessment of Tone Mapped Images,” IEEE Transactions on Image Processing, vol. 22, no. 2, pp. 657–667, Feb. 2013.