1. INTRODUCTION

One of the most interesting features of Earth, in the perspective of common people, is the ever-changing distribution of clouds. They are as natural as anything we encounter in our daily lives. As they float above us, we hardly give their presence a second thought. And yet, clouds have an enormous influence on Earth's energy balance, climate, and weather. My work is for recognize the type of cloud and to predict rainfall or thunder storming. Rainfall plays an important role in the water cycle by providing water to the surface of the Earth. Rain sustains agriculture and provides water to streams, which is important for aquatic life and navigation.

Clouds are important phenomena strongly affecting the total incoming radiance at a given point in the Earth surface. For a growing number of applications in diverse fields such as agriculture, forestry or energy production and management, being able to accurately estimate and predict solar radiation at a given ground location and at short time scales, is becoming an extremely important task as solar radiation strongly influences the relevant processes and energy balances. By the better understanding of the clouds, better models can be developed for the prediction of different atmospheric disturbances such as typhoons, hurricanes, dust storms and etc.

Traditionally, rainfall has been observed using rain gauges at the surface. Gauges provide ground truth data, albeit with several known sources of error. Rain gauge amounts are read either manually or by automatic weather station (AWS). The frequency of readings will depend on the requirements of the collection agency. Rain gauges are also limited because they are observations at a single point and are often spaced sparsely and report data at infrequent intervals.

Weather radars measure reflectivity, which can be related to rainfall rate. Rainfall vary greatly in space and time, and using a single conversion from reflectivity to rain rate over an entire radar scanning area for a whole event may not produce reasonable estimates of rainfall.

Existing techniques for interpreting cloud images have concentrated largely on satellite images. Large-scale cloud information is available from several satellites, but such data is provided in a

low resolution and may contain errors. Low or thin clouds and surface are frequently confused because of their similar brightness and temperature.

Nowadays ground-based imaging devices are commonly used to support satellite studies. Their Whole Sky Imagers are constructed to measure sky radiance at diverse wavelength bands across the whole hemisphere. Due to the high-quality components involved, these imagers are often too expensive for small research groups. Therefore, as a cost-effective alternative, a few research institutions in several countries have developed non-commercial sky cameras for their own requirements.

Ground-based assessment of cloud cover provides a much finer resolution in both space and time than is possible by satellite, and can provide valuable information concerning cloud at different heights. Coomes and Harrison developed a cloud estimation system using ground-based radiometric measurements, but although this method provides good estimates of total cover, it was unable to distinguish between different classes of cloud.

It has been demonstrated that both color and texture are useful features in the differentiation of cloud from sky, although neither approach has proved sufficient for complete analysis. A simple method to combine the features using a Bayesian classifier has shown an improvement in classification over the two independent approaches, but there are still problems to be resolved concerning: (i) cloud border accuracy, and (ii) the weighting of die classifier towards texture. Work by Parikh and Rosenfeld [2] used images from both the infra-red and visible regions of the spectrum. Their system extracted spectral and textural information, and reached a classification using a complex selection procedure.

Another recent paper by Calb´o and Sabburg, 2008 introduces some possible criteria for sky-images to classify eight predefined sky conditions, which include statistical features, features based on the Fourier transform, and features that need the prior distinction between clear and cloudy pixels. However, the classifier is based on a very simple classification method and achieves an accuracy of only 62%. Other publications handle simpler issues such as the estimation of cloud base height or the identification of thin and thick cloudiness (e.g. Seiz et al.,

2002; Kassianov et al., 2005; Long et al., 2006; Cazorla et al., 2008; Parisi et al., 2008). Parisi et al. (2008) in particular report that they were not able to classify cloud type.

Another automatic cloud classification [11] algorithm is based on a set of mainly statistical features describing the color as well as the texture of an image. The k-nearest neighbour classifier is used due to its high performance in solving complex issues, simplicity of implementation and low computational complexity. The random sample TEST yields an overall classification rate of 87.52%.

For developing an automatic ground-based cloud observation system, we have a number of possible methods to extract useful spectral (color), and textural (local spatial) features. The spectral-texture features used are an extension of those used by Laws, from monochrome to color images. A set of local linear transforms is applied to each spectral channel (RGB), and a macro statistic taken over a 32x32 window. Larger neighbourhood windows have been found to offer no improvement in classification, but introduce greater errors at boundaries between classes. Previous studies have usually concentrated on a limited set of samples taken from a single set of satellite images. This strategy is adequate for satellite images, which contain relatively little inter-image variation, but in ground-based data, is far more variable. The position of the sun, the time of day, and the angle with which the image was taken, all result in a very wide range of lighting conditions. For training we can use BPNN, but the major disadvantage is the lack of clear strategy for choosing between the many training paradigms. Different problems appear to require different training mechanisms, or different parameters for the same training mechanism. Finding the best parameters and training algorithms is a matter of experience, trials and patience.

Considering most of the drawbacks and parameters, in this project it will focus on cloud images and uses image processing techniques. The input image is digital cloud. When the input is given to the system, apply wavelet in order to split the status of sky from the input image. In previous papers law's texture description was taken into account to differentiate between the sky and cloud. Now evolve with the feature extracted image to represent the status of sky. After that the feature extracted images of sky status is processed with cloud mask algorithm and evolve with the feature extracted image to represent the status of cloud. Now k-means clustering is applied to

the image to find the type of cloud from its shape and density. After finding the cloud type the Information about cloud and the status of rain can be found.

The main focus is to recognize the type of cloud and estimate rainfall from the digital cloud images. Use wavelet to separate the points needed for the cluster and using k-means clustering to combine those points will provide a better performance than previous techniques used. The Scope of the work is predicting rainfall using the digital images rather than using the satellite images. The Satellite images costs lot so everyone can't get them easily. The steps used in this process are Data collection, Sky Status, Cloud Status, Cloud Type, Cloud Information, Rainfall Status and Performance measures.

2. RELATED WORK

Recently there has been lot of research and studies happening in the area of cloud characterization using machine learning algorithms. Researchers were successful to a great extent in case of satellite images. However when it comes to the digital image, the hurdles and technical limitations are much more as compared. A few reason's are given below:

- The extraction of features from 2D image.
- The frequently changeover from one cloud class to another class, a natural phenomenon that will always lead to some misclassifications of these classes using automatic methods.

Some of the other studies conducted are as listed below:

- Estimation of Cloud Cover using Colour and Texture [1].
- Cloud classification using BPNN[12].
- Automatic cloud classification of whole sky images[11].

2.1 Estimation of Cloud Cover using Colour and Texture.

In this, experiments with colour and texture analysis on cloud images obtained from a ground-based colour camera, and demonstrate that although each provides good discrimination neither is sufficient for complete analysis. Texture has been successfully employed to classify satellite images of cloud, and have seen that on the basis of intensity and colour there is little difference between transparent cloud (such as cirrus) and sky. Texture may provide a means to distinguish between them, and may also be useful for recognising different classes of cloud such as cumulus, stratus and cirrus.

A texture model is based on local linear transforms. This approach provides much better classification than with either of the other two approaches, although it does result in an n-fold increase in computation (where n is the number of local linear transforms).The image is first transformed using 9 local linear operators which were defined by Laws. These make use of a number of empirical linear transformations which are intended to extract elementary underlying structures such flats, spots and edges. For any given point an energy measure, such as the variance or some higher order statistical measure, is calculated over a local neighbourhood in

each of the transformed images. The local texture properties of that point are described by the vector of feature energy measures. Texture is useful, but there are two difficulties with the approach: (i) sky has no texture, so that the borders of cloud extend into regions of sky causing a small overestimation of cloud; (ii) some regions of cloud have little texture and are indistinguishable from sky, they are therefore consistently mislabelled.

Both colour and texture have been shown to be important indicators to the presence of cloud, but neither is sufficient for complete segmentation. Chromatically, haze is indistinguishable from cloud, and cirrus is indistinguishable from sky. Texturally, smooth regions of cloud are frequently mis-labelled as sky. There is a clear attraction in using a combination of both methods. Normalised colour space produces two chromaticity measures. As with the texture classification a Bayesian classifier can be employed to minimise the probability of mis-classification. The second training set of images was reprocessed to calculate the new class-mean feature vectors consisting of texture and colour measures. The classification procedure was based on that used for texture analysis.

The combination of colour and texture can greatly improve classification but is still imperfect. Obvious errors which persist from texture classification are the misclassification of smooth regions of cloud the extended classification of cloud at borders with sky. Both colour and texture are useful features in the differentiation of cloud from sky, although neither approach has proved sufficient for complete analysis. A simple method to combine the features using a Bayesian classifier has shown an improvement in classification over the two independent approaches, but there are still problems to be resolved concerning: (i) cloud border accuracy, and (ii) the weighting of die classifier towards texture.

This approach is suitable for simple models, but part of the success relies on the feature distributions being normal. This cannot be guaranteed in a problem domain such as this, since the radiation passing through the atmosphere is known to change throughout the day. In order to compensate for this, and deal with extreme situations such as sun-rise and sun-set, a more elaborate technique for combining colour and texture is required.

2.2 Cloud classification using BPNN.

The images were processed to extract spectral-texture features similar to Laws Texture measures. Each feature is the local energy of one of nine 3x3 linear transforms, applied to each colour channel separately. This simple colour extension to the monochromatic texture analysis used by Laws provides a total of 27 features. Other researchers such as Unser, have used Principal Component Analysis to identify sets of local linear transforms with better discriminating power, however similar experiments with our data showed no apparent improvement in classification.

Previous studies have usually concentrated on a limited set of samples taken from a single set of satellite images. This strategy is adequate for satellite images, which contain relatively little inter-image variation, but ground-based data is far more variable. The position of the sun, the time of day, and the angle with which the image was taken, all result in a very wide range of lighting conditions. The experimental design uses test and training data randomly taken from within a common set of images. Here they preselected independent sets of test and training images to study the abilities of the classifier in the face of both interand intra-image differences.

Since there were limited examples of cirrus available in the data set they biased the random samples to ensure that each training set contained some examples of cirrus. The number of samples for each image was also biased so that the relative number of examples for each class would be the same. This limited selection of cirrus images accounts for much of the poor classifier performance.

The BPNN was 58.4% correct compared to 59.4% for the Bayesian Classifier

2.3 Automatic cloud classification of whole sky images.

An automatic cloud classification based on a set of mainly statistical features describing the colour as well as the texture of an image. The k-nearestneighbour classifier is used due to its high performance in solving complex issues, simplicity of implementation and low computational complexity.Seven different sky conditions are distinguished: high thin clouds (cirrus and cirrostratus),high patched cumuliform clouds (cirrocumulus and altocumulus), stratocumulus

clouds, low cumuliform clouds, thick clouds (cumulonimbus and nimbostratus), stratiform clouds and clear sky.

Spectral features describe the average color and tonal variation of an image. In cloud classification they are useful to distinguish between thick dark clouds, such as cumulonimbus, and brighter clouds, such as high cumuliform clouds, and to separate high and transparent cirrus

clouds from others. The spectral features implemented in the algorithm are the following:

– Mean (R and B)

– Standard deviation (B)

–Skewness (B)

–Difference (R-G, R-B and G-B)

To describe the texture of an image, statistical measures computed from Grey Level Co-occurrence Matrices (GLCM) may be used. A GLCM is a square matrix for which the number of rows equals the number of grey levels in the considered image. To avoid dependency on image orientation, often an average matrix is calculated out of two or four matrices, expressing mutually perpendicular directions. Furthermore, because the computation of GLCMs strongly increases with increasing number of intensity levels G, it is advantageous to reduce the original number (G=256) of grey levels.

To classify the images the k-nearestneighbour (kNN) method is chosen. The assignment of an image to a specific class using kNN classifiers is performed by majority vote. After pre-processing, several spectral and textural features are extracted from an image. In the next step, the computed and normalized feature vector x is compared with the known feature vectors xi of each element in the training data by means of a distance measure. The choice of these features is based on their Fisher Distances Fxij, a selection criterion used in cloud classification work relating to satellite imagery. Spectral features describe the average colour and tonal variation of an image. In cloud classification they are useful to distinguish between thick dark clouds, such as cumulonimbus, and brighter clouds, such as high cumuliform clouds, and to separate high and transparent cirrus clouds from others.

To describe the texture of an image, statistical measures computed from Grey Level Co-occurrence Matrices (GLCM) may be used. In addition to the features described above, the cloud cover also computed.

The kNN classifier is often criticized for slow runtime performance and large memory requirements. Also misclassification can be happened.

3. RESEARCH QUESTIONS

The main objective of my work is to recognize rainfall and estimate it using image processing concepts. The main objective can be reached by defining following sub-objectives:

- To extract the features that characterize digital cloud image.
- Classify the clouds efficiently.
- If there are some gray levels in the image with very high frequencies, they dominate the other gray levels having lower frequencies. Address this problem.
- Finding one threshold that segments the image pixels into clear sky and cloud.

The focus of this research is to predict rainfall using texture, color, and density. Rather than using the Law's, here concentrating on image processing concepts and machine learning algorithms.

For address the above challenges, initially, use wavelet transform for calculating the sky status. Then for cloud status, dynamic histogram equalization and obviously cloud mask algorithm is used. Classification will be the major part, so for that use k-means clustering algorithm.

To enhance the performance, we can incorporate some other concepts also. RBF can be used for thresholding. The image thresholding approach, using radial basis function neural networks selected by means of multi- objective genetic algorithms will produce the best results achieving a decrease of about 50% in the cloud cover estimation error.

4. DESIGN – METHODS AND PROCEDURES

4.1 Brief Introduction of Haar wavelet transform.

The proposed system uses Haar wavelet transform as the preprocessing technique. Using this method, it efficiently separates sky and cloud region.

Wavelet transform divides the information of an image into approximation and detail subsignals. The approximation subsignal shows the general trend of pixel values and other three detail subsignals show the vertical, horizontal and diagonal details or changes in the images. If these details are very small (threshold) then they can be set to zero without significantly changing the image.

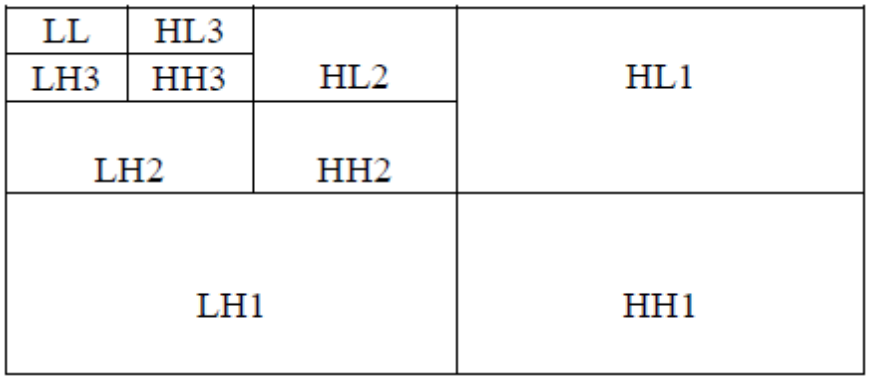


Figure 1: Structure of wavelet decomposition

To do this, first the average the pixels together, pairwise, is calculated to get the new lower resolution image with pixel values. Clearly, some information is lost in this averaging process. We need to store some detail coefficients to recover the original four pixel values from the two averaged values. This single number is used to recover the first two pixels of our original four-

pixel image. Thus, the original image is decomposed into a lower resolution (two-pixel) version and a pair of detail coefficients. Repeating this process recursively on the averages gives the full decomposition. Thus, for the one-dimensional Haar basis, the wavelet transform of the original four-pixel image. We call the way used to compute the wavelet transform by recursively averaging and differencing coefficients, filter bank. We can reconstruct the image to any resolution by recursively adding and subtracting the detail coefficients from the lower resolution versions.

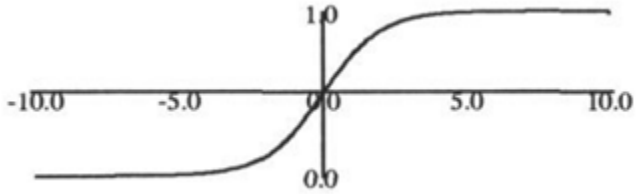
The transformation of the 2D image is a 2D generalization of the 1D wavelet transformed already discussed. It applies the 1D wavelet transform to each row of pixel values. This operation provides us an average value along with detail coefficients for each row. Next, these transformed rows are treated as if they were themselves an image and apply the 1D transform to each column. The resulting values are all detail coefficients except a single overall average co-efficient. In order to complete the transformation, this process is repeated recursively only on the quadrant containing averages.

i) Start with the matrix P representing the original image.
ii) Compute the transformed matrix T by the operation averaging and differencing (first for each row and then for each column)
iii) Choose a threshold method and apply that to find the new matrix say D.
iv) Use D to compute the compression ratio and other values and to reconstruct the image as well.

4.2 BPNN Architecture.

Neural Networks consist of a network of interconnected processing elements, with each element having its own set of inputs and associated weights. When in operation a cell calculates its response by summing the product of all inputs with their weights. A non-linear transfer function is applied to this sum, giving the actual output from the cell. The transfer function may be a simple threshold, or a function which simulates the non-linear characteristic of biological cells.

Functions such as the Sigmoidal Logistic function (Figure 2), and the Hyperbolic Tangent function (tanh), are frequently used. The necessity for the non-linearity, and the potential for representing arbitrary functions with multiple layer networks.



$$SIG = \frac{1}{(1 + exp(-SUM))}$$

Figure 2 : The sigmoid response function

A standard artificial Neural Network configuration consists of three types of layer: an input layer, an output layer, and one or more intermediate hidden layers (Figure 3). In operation, a set of inputs is presented at the input layer, and consecutive layers of the network are processed, with each layer's output forming the input to the following layer. The final layer provides the network's output. Alternative architectures such as recurrent networks, or networks with connections which skip layers have been suggested but are outside the scope of this paper.

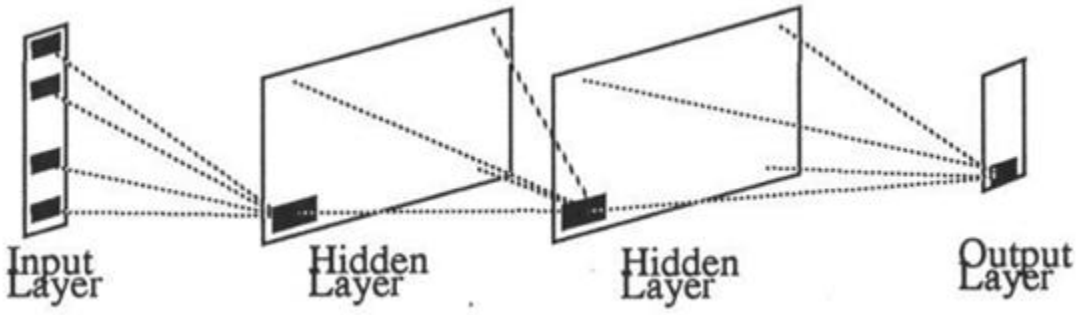


Figure 3: Neural Network consisting of four layers

The principal attraction of NNs is the ability to train the behaviour of a system in response to data of known origin. A number of Neural Network training schemes, both supervised and unsupervised, have been proposed.

Using backpropagation the error of the entire network to a given input is calculated at the output layer, and each cell is given its own error term 50utput. Usually for binary outputs this is just the difference between the target and the output (Equation 1).

$$\delta_{output} = \text{Target} - \text{Output} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

This error is passed back through the input weights to the previous layer. Each cell in this layer sums all of the errors from the cells in the output layer. This sum, multiplied by the derivative of the transfer function, represents the error of a hidden cell Shidden (Equation 2). This value will be subsequently passed back to other hidden layers, and so the error is propagated.

$$w_{lij}^{new} = w_{lij}^{old} - \alpha \lim_{n\to\infty} \sum_{k=1}^{N} o_{(l-1)j}^{k} \delta_{li}^{k} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

An identical weight updating process is used in both the hidden and output layers. The generalised delta rule learning law is guaranteed to converge to a minimum error state. Although this learning law always minimises the error, it takes an indefinite length of time to converge. For practical purposes a finite version of the rule, with a fixed batch size N, is often used.

$$\delta_{hidden} = \text{out}_{\text{hidden}} (1 - \text{out}_{\text{hidden}}) \sum_{\text{i}=1}^{\text{N}} w_{hidden,i} \delta_i \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

There are many possible variants on this basic architecture and learning law with various advantages and pitfalls. Unfortunately there is no clear mechanism for choosing between them, and the choice of learning law, parameters, and architecture must be selected to suite each particular problem.

4.3 Dataset- Training and Testing

Dataset used here is simple digital cloud images. In the data collection phase, collect digital cloud images and store the image in the file system. The dimension chosen for the images is 256x256. The 52 images were taken from internet and also with 10 megapixel camera. The images of four types of clouds are taken. They are fair weather cumulus, altostratus, cirrus and cumulonimbus.

4.4 Validation of Data

The input digital cloud image from each classification is given to the system. Check how many images are correctly classified and how many are misclassified by comparing it with output got from laws. The percentage of accuracy is the ratio of measure of correctly classified data to the total data.

4.5 Approach towards the solution

4.5.1 Overview

The input image is digital cloud image. When the input is given to the system, apply Haar filter in order to split the status of sky from the input image. In previous papers law's texture description was taken into account to differentiate between the sky and cloud. Now evolve with the feature extracted image to represent the status of sky. After that the feature extracted images of sky status is processed with cloud mask algorithm and evolve with the feature extracted image to represent the status of cloud. Now k-means clustering is applied to the image to find the type

of cloud from its shape and density. After finding the cloud type the Information about cloud and the status of rain can be found.

4.6 Solution implementation

In this project, we are trying to implement one application which will automatically detect rainfall by taking a digital cloud image. Also wants to enhance the performance by adding new techniques which will reduce the complexity of my application.

Once the image is fed to the system, initially it will undergo preprocessing. For the efficient removal of background, Haar wavelet transform is used. After this, get a filtered image and then converted to binary image. We get a processed image and this image is AND with the original gray scale image. For the computation purpose convert this image to color by AND with the original color image. After this cloud masking, extract the features sky density, cloud density and the cloud status. The intensities and centroids are calculated and passed to the k-means clustering. The cloud is classified according to the clusters which will return after the classification module.

After the classification we can made a conclusion about the cloud, ie. whether it is rainy cloud or not. If the cloud belongs to cumulonimbus or cirrus it is rainy. Actually cirrus is the parent cloud of rainy and thunder storming clouds.

4.7 Focus of the minor project

By implementing the above proposed method, creating a scenario wherein we can automatically recognize cloud and predict rainfall. The cloud classification and prediction of rainfall according to the classification is the focus of minor project. Also need to detect the presence of precipitation and cloud cover.

4.8 Resources

4.8.1 Hardware Requirements

Processor	:	Pentium IV or above
Memory	:	2GB or above
Hard Disk	:	40 GB or above

4.8.2 Software Requirements

Operating System	: Windows XP
IDE	: Net Beans
Java Version	: Java SE

5. TIME SCHEDULE

ID	Task Name	Date of Completion
1	Abstract Submission	19-09-2012
2	Literature Survey	18-10-2012
3	Preparation of Research Proposal	23-11-2012
4	Design- Methods and Procedures	23-11-2012
5	Data Collection	28-11-2012
6	Preprocessing Implementation	10-12-2012
7	Cloud Masking and Feature Extraction	23-12-2012
8	Classification	15-01-2013
9	Training using Neural network and integration	25-01-2013

Table 1: Time schedule

6. IMPLEMENTATION

6.1 System Description

In this section, the system architecture and each part of the architecture is explained Figure 6.1 represents the architecture of cloud classification.

Below is general overview of these modules, the elaborated discussion on these components is in detailed design section.

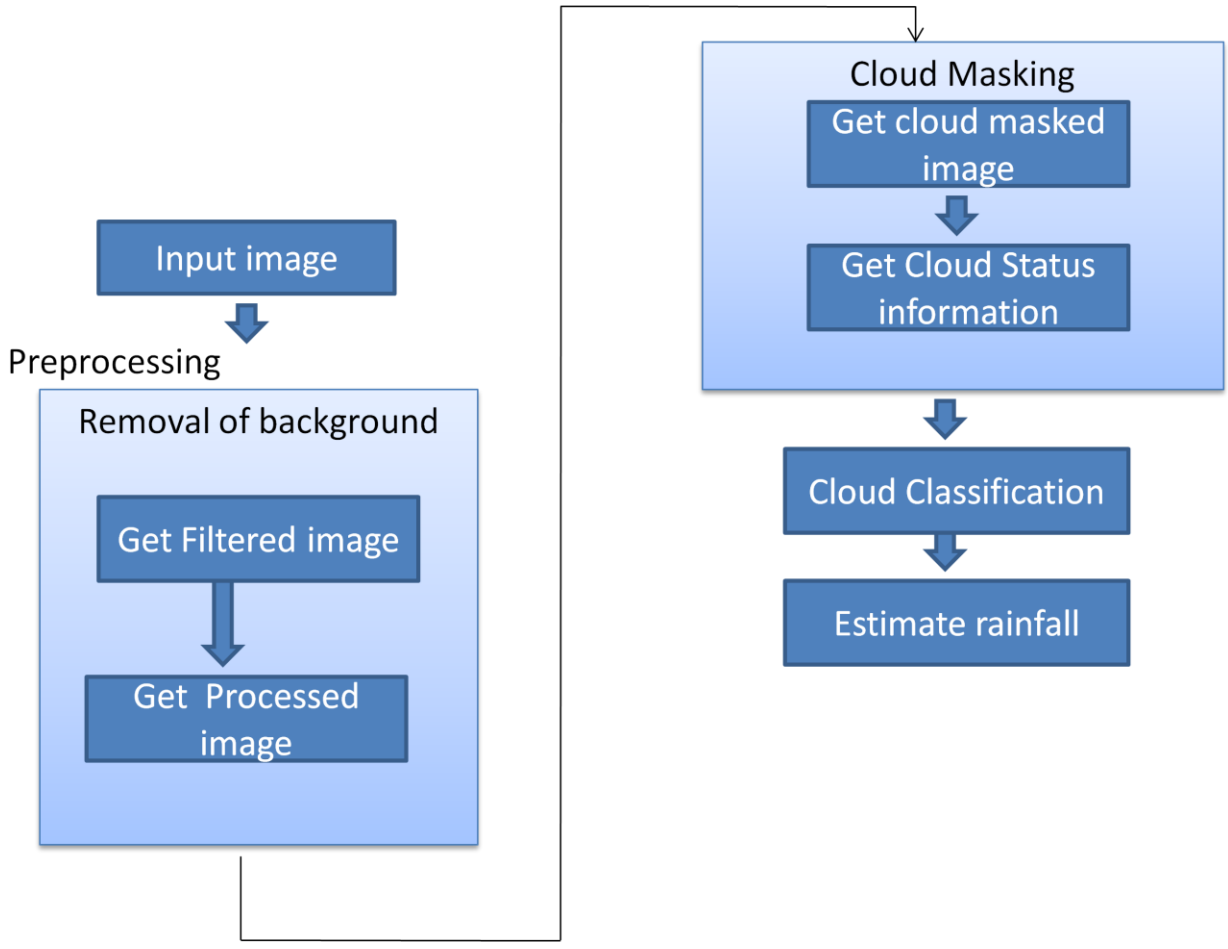


Figure 4: Design of Proposed System

6.2 Sky status

The first module is sky status. Use Haar filter for finding the sky status. It separates the point needed for the cluster. In wavelet separation points are used in the identification of the clouds or sky.

Wavelets provide a mathematical way of encoding information in such a way that it is layered according to level of detail. This layering facilitates approximations at various intermediate stages. These approximations can be stored using a lot less space than the original data. Here a low complex 2D image compression method using wavelets as the basis functions .The particular wavelet chosen and used here is the simplest wavelet form namely the Haar Wavelet.

We have to segment the cloud from other points. The wavelet threshold for the clouds is > 50 and <200. The formula to find the sky status is

$$Seg(n) = Segment(image[I,j])$$

$$\text{Sky Status} = \Sigma \text{ Highest Segmentation Value} + \text{Smallest Segmentation Value}$$

6.3 Cloud status

The second module is the cloud status. The cloud varies according to the thickness. So the high density is detected. The middle level density will be considered as the sky. In mild time or moody time sky is considered as cloud. To find the mask the histogram equalization is used the value with the highest weight will be considered as sky and the others will be considered as cloud. Also the cloud mask algorithm is used here. The formula to find the cloud status is given as follows.

$$\text{Cloud Status} = \text{Total No. of Segment} - \text{Sky Status}$$

The cloud mask algorithm consists of certain tests. Single pixel threshold tests are used first. Dynamic histogram analysis is used to get threshold. Thick High Clouds (Group 1): Thick high clouds are detected with threshold tests that rely on brightness temperature in infrared and water vapour bands. Thin Clouds (Group 2): Thin clouds tests rely on brightness temperature difference tests. Low Clouds (Group 3): Low clouds are best detected using solar reflectance test and brightness temperature difference. Spatial uniformity test is also used over land surface. High Thin Clouds (Group 4): This test is similar to Group1, but it is spectrally turned to detect the presence of thin cirrus. Brightness temperature difference test and spatial uniformity test are applied. Temporal uniformity test is also used.

6.4 Cloud Type

The major task here is to find the type of cloud as per the cloud status each and every cloud will be having its own shape and density and the values are matched accordingly. The type of cloud is identified by using clustering. Use k-means clustering to combine the pixels in order to differentiate the clouds. The thickness of the clouds will be in the base part. The colour, Shape and Texture are the concepts used in order to find the type of cloud. The formula to find the cloud type is shown as follows:

$$H(n) = \sum_{n=0}^{255} C[i,j]$$

Cloud id = Highest Density of Cloud Status

k-Means algorithm is an unsupervised clustering algorithm that classifies the input data points into multiple classes based on their inherent distance from each other. The algorithm assumes that the data features form a vector space and tries to find natural clustering in them. The points are clustered around centroids $\mu_i \forall$=1….k which are obtained by minimizing the objective

$$V = \sum_{i=1}^{k} \sum_{x_j \in s_j} (x_j - \mu_i)^2$$

where there are k clusters Si, i = 1, 2,…., k and μi is the centroid or mean point of all the points xj ϵ Si. As a part of this project, an iterative version of the algorithm was implemented. The algorithm takes a 2 dimensional image as input. Various steps in the algorithm are as follows:

(i) Compute the intensity distribution (also called the histogram) of the intensities.

(ii) Initialize the centroids with k random intensities.

(iii) Repeat the following steps until the cluster a label of the image does not change anymore.

(iv) Cluster the points based on distance of their intensities from the centroid intensities.

(v) Compute the

$$C^{(i)} = \arg min_j \left\| x^{(i)} - \mu_j \right\|^2$$

new centroid for each of the clusters.

$$\mu_i = \frac{\sum_{i=1}^{m} \{c_{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c_{(i)} = j\}}$$

where k is a parameter of the algorithm (the number of clusters to be found), i iterates over the all the intensities, j iterates over all the centroids and μ_i are the centroid intensities.

The next step is cloud information where the theoretical proof of the clouds that is the height, altitude, classification and appearance are given.

6.5 Rainfall Prediction

The major step is the prediction of rainfall; here the rainfall is estimated as per the type we recognize. Cumulonimbus and nimbostratus are the rainfall clouds. So we take the color and shape and also the width and find the rainfall status the temperature is also taken into account. The analysis part consists of Sky Status, Cloud Status, Sky Density and Cloud Density.

The cloud density and sky density is calculated using the formula

$$\text{Cloud Density} = \sum_{i}^{n}\sum_{j}^{m} C[i,j]$$

$$\text{Sky Density} = \sum_{i}^{n}\sum_{j}^{m} S[i,j]$$

The formula to find the sky status is shown as follows;

$$\mathbf{SS[n]} = \sum_{\mathbf{0}}^{\mathbf{255}} \mathbf{S[i,j]}$$

The formula to find the cloud status is shown as follows

$$TS[n] = \sum_{n=0}^{255} T[i,j]$$

The formula for wavelet, k-Means clutering and the performance measure are shown as follows:

$$\text{Wavelet} = 1 - \frac{\text{wavelet feature}}{\text{Total no. of pixels}}$$

$$\text{Clustering=1-(k-means Cluster value)}$$

$$\text{Performance}=\frac{\text{Total no.of Cloud values}}{\text{Total no.of pixels}} * 100$$

$$PSNR = 10\log_{10}\{255^{2}(\frac{1}{UV} * \sum_{i=0}^{U}\sum_{j=0}^{V}\|I(i,j) - O(i,j)\|^{2})^{-1}\}$$

Here, PSNR is the Peak Signal Noise Ratio, I (i,j) is the input and O(i,j) is the output. U,V represents the rows and columns respectively. In the measurement part we compare the two parts.

The data flow diagram to explain the process is shown as follows:

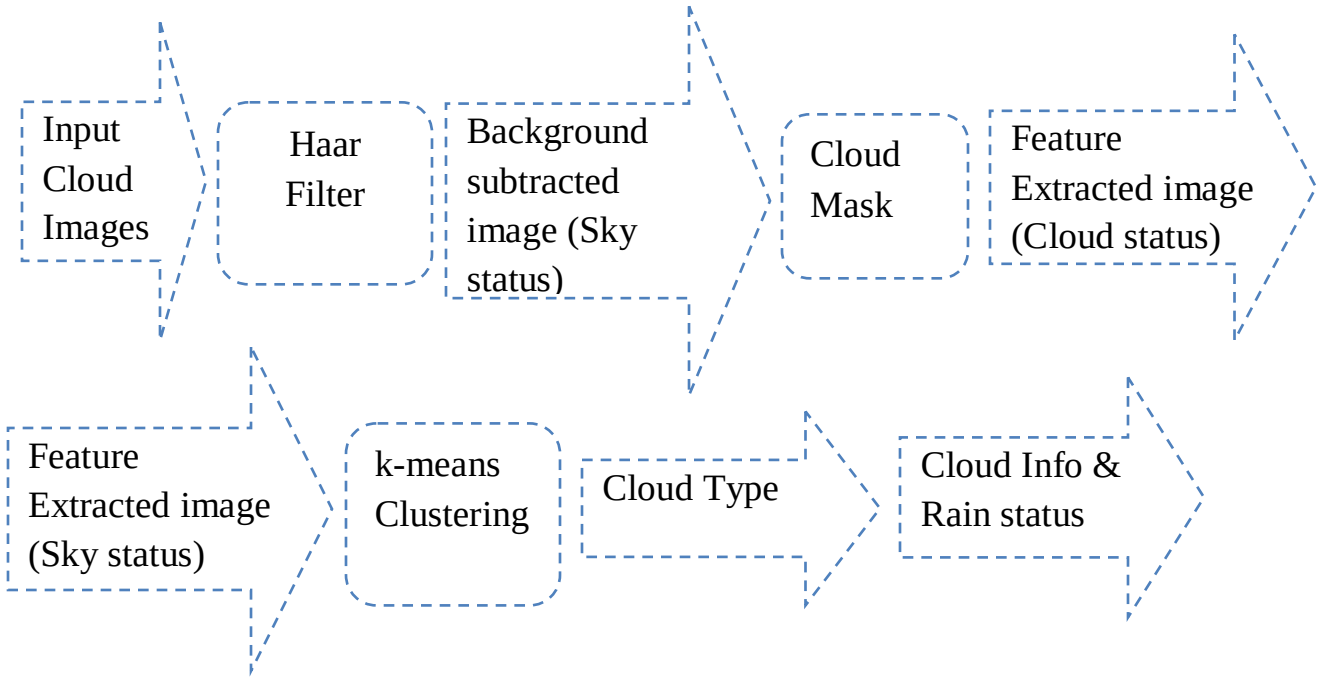


Figure 5: Data Flow

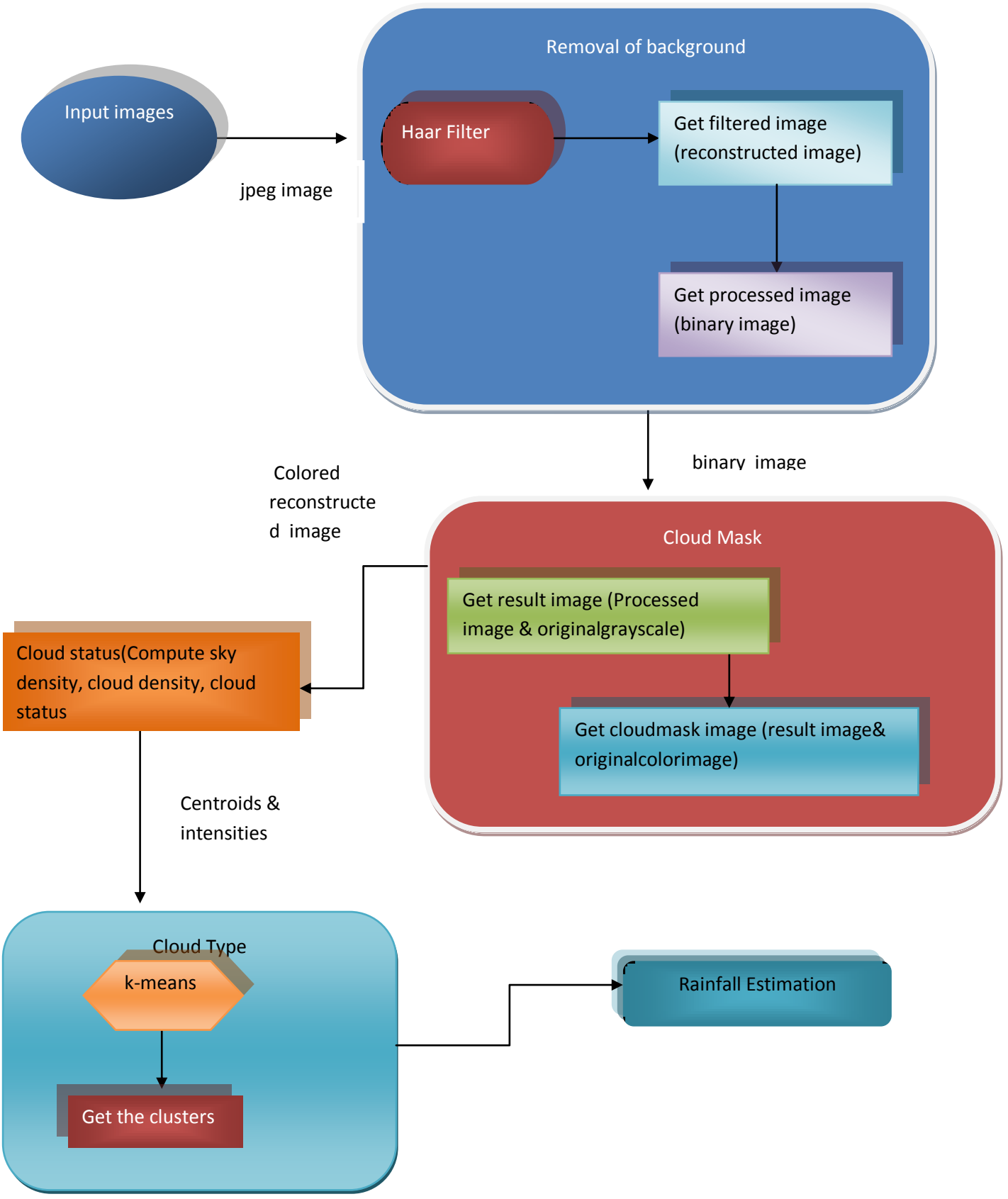


Figure 6: Architecture of Cloud Classification

OpenImage() //open the image from local depository

FeatureExtract() //convert to grayscale and pass to haar filter

ExtractWaveletFeatures()// do haar filter

CloudMask()// get result image(originalgray&processedimage)

CloudMaskMask()//get the cloud masked image(original color&rresultimage)

CloudStatus()// compute the densities,ie.cloud and sky

public class FilterProcessor()

//this is for doing haar filter, according to the values set by this class,haar filter compute its output

public class HaarFilter()//Two dimensional values stored in a byte array, methods for reversing the filter and converting filters and data into images.

public HaarFilter(int size, int iterations, int fractionalBits)

public int getIterations()//The number of filtering iterations performed, return the number of iterations performed during filtering

public int[] filter(byte[] values, int[] filter)// Filters the byte values supplied by applying the Haar wavelet for a number of iterations and stores the result in the specified integer array. Return the array which contains the filter values

//Here we are input the image and processed to get the Haar filtered output.

public class kmeans()

public static double[] intensities()

public static double[] initCentroids()

public static double[] kmeans()

// In this part, we are passing the intensities and centroids. And based on that classification is happening and output the clusters.

We get the type of clouds, for each cloud

6.6 Algorithm

1. Input image
2. Get the Filtered image
3. Get the Processed image
4. Get the Result image
5. Get Cloud masked image
6. Retrieve the cloud status information: Cloud density, Sky density, Highest density
7. Perform clustering:
    i. Input the intensities and centroids
    ii. Return clusters
8. Estimate rainfall from the information

6.7 Implementation snap shots

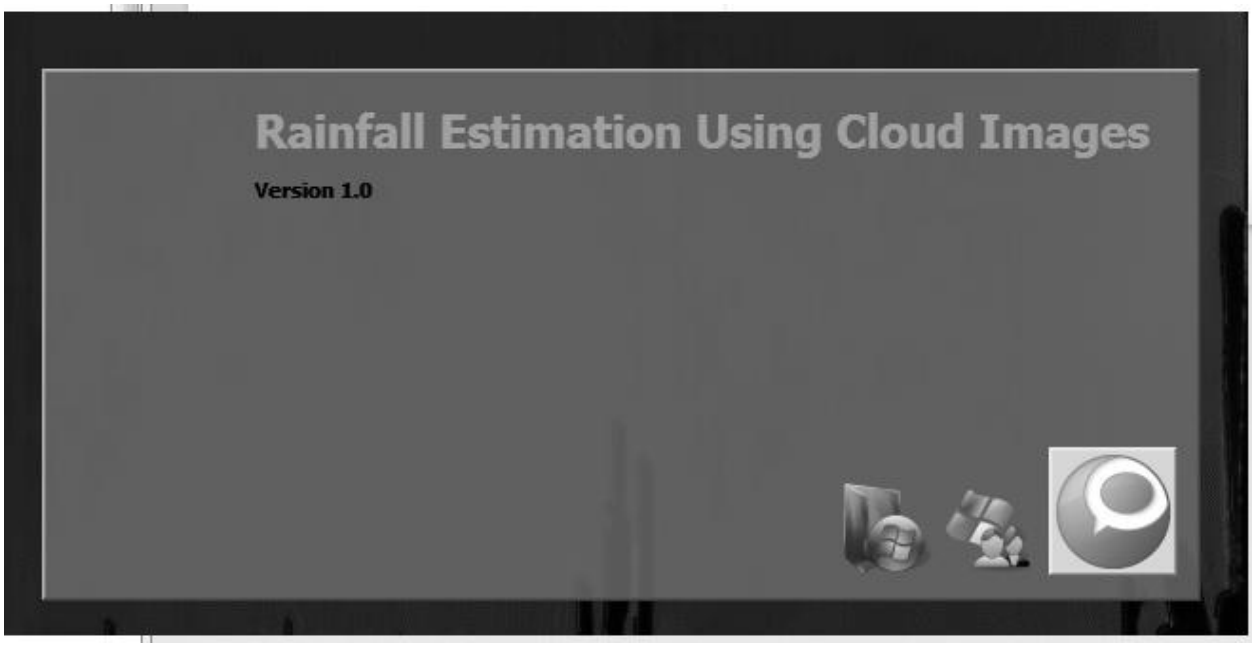


Figure 7: Project intialization screen

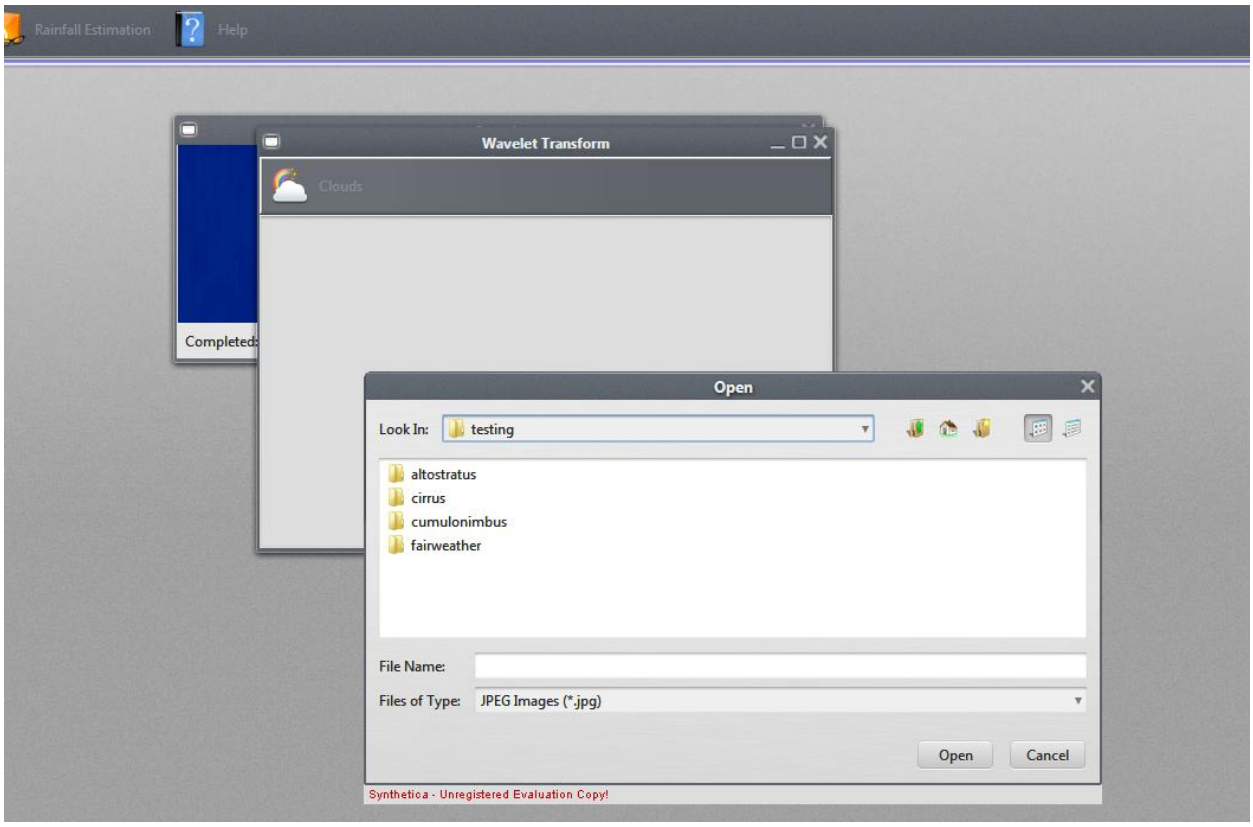


Figure 8: Open image Screen

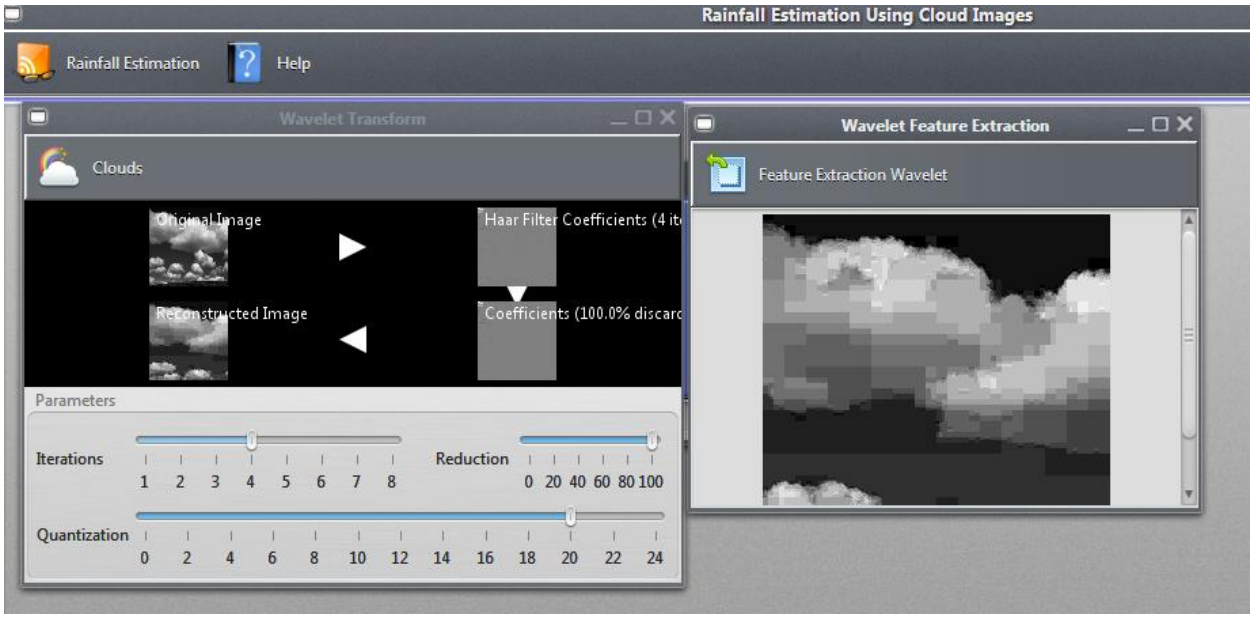


Figure 9: Preprocessing Screen

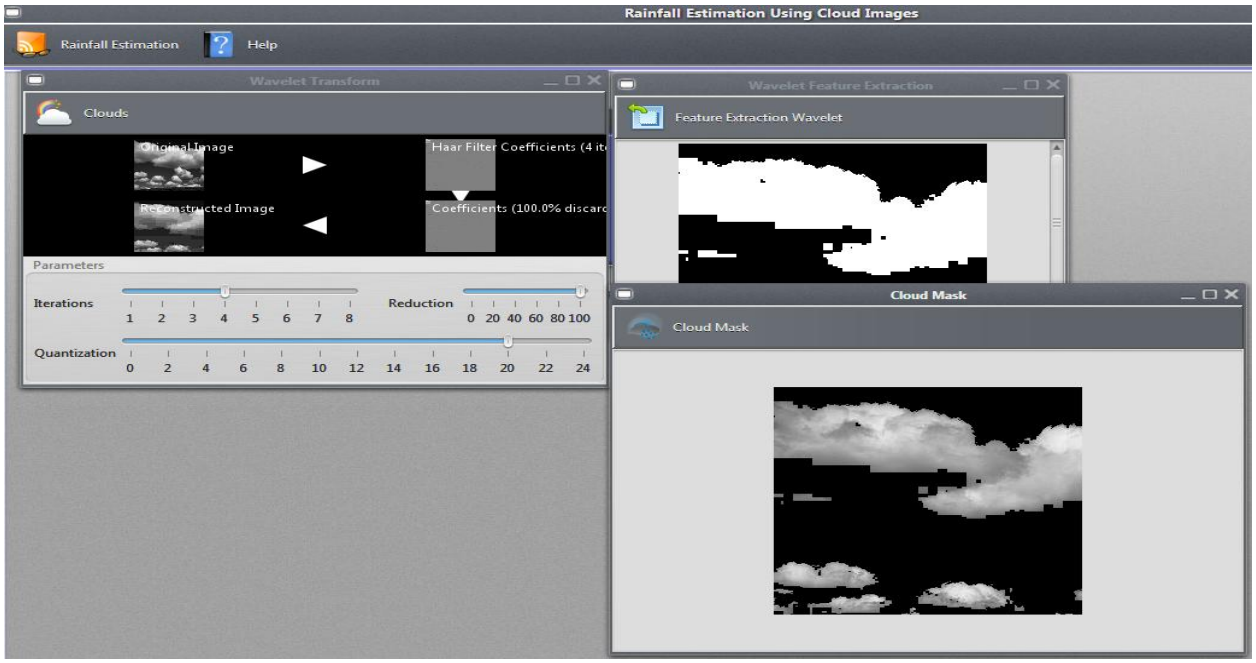


Figure 10: Cloud Masking Screen

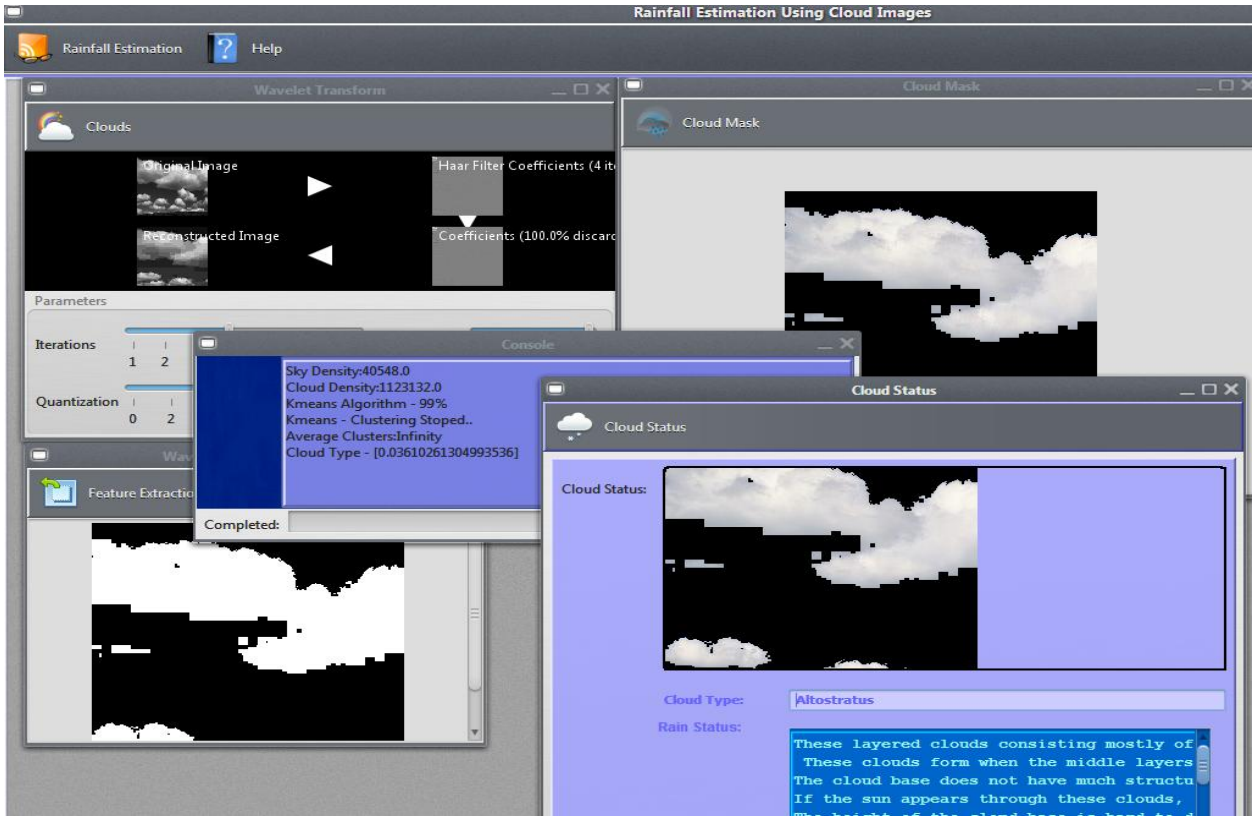


Figure 11: Output Screen

7. TESTING - RESULTS AND ANALYSIS

This work is mainly comprised of classification problem. So the most suitable evaluation method is the Receiver Operating Characteristic Convex Hull (ROCCH) method.

The Receiver Operating Characteristics (ROC) analysis is a method for evaluating and comparing a classifiers performance. It has been extensively used in signal detection, and it was introduced and extended in for the Machine Learning community. In ROC analysis, instead of a single value of accuracy, a pair of values is recorded for different class and cost conditions a classifier is learned. The values recorded are the False Positive rate (FP) and the True Positive rate (TP), defined in terms of the confusion matrix as:

$$FP=fp / (fp+tn)$$

$$TP=tp / (tp+fn)$$

In this formula's, "fp" is the number of false positives (misclassification), "tp" is the number of true positives (correct classification), etc. The TP rate is equivalent to the recall of the positive class, while the FP rate is equivalent to 1 less the recall of the negative class.

7.1 The (ROCA) % Evaluation Measure

There are several possible evaluation measures for comparing the classification performance of different filtering methods. Clearly, to maximize the number of True Positives (TPs), and the number of True Negatives (TNs). Furthermore, we would like to minimize the number of False Positives (FPs), and to minimize the number of False Negatives (FNs).

Many metrics have been used for performance evaluation of classifications. All of them are based on the confusion matrix as shown at Table2.

	predicted positives	predicted negatives
real positives	TP	FN
real negatives	FP	TN

Table 2: Confusion Matrix

To get optimal balanced classification ability, TP (sensitivity) and TN(specificity) are defined which is adopted to monitor classification performance on two classes separately. Notice that TP is sometimes called true positive rate or positive class accuracy while TN is called true negative rate or negative class accuracy. Based on these two metrics, g-mean has been proposed which is the geometric mean of classification accuracy on negative samples and classification accuracy on positive samples.

$$\text{g -mean} = \sqrt{\text{sensitivity}.\,\text{specificity}}$$

Both g-mean and ROC can be used if the target is to optimize classification performance with balanced positive class accuracy and negative class accuracy.

On the other hand, sometimes effective detection ability is for only one class. For example, for credit card fraud detection problem, the target is detecting fraudulent transactions. For diagnosing a rare disease, what we are especially interested in is to find patients with this disease. For such problems, another pair of metrics, precision and recall as defined is often adopted. Here, recall is the same as sensitivity. The f-value defined, used to integrate precision and recall into a single metric for convenience of modeling. Similarly, the Area under Precision/Recall Curve is also used to indicate the detection ability of a classifier between precision and recall as a function of varying a classification threshold.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{f-value} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Because our target is to classify cloud, metrics concerning one-class detection ability are more suitable than metrics concerning balanced classification ability. However, if the precision is too low, the classifier has no practical usefulness.

7.2 EXPERIMENTAL RESULTS

From the test, we acquire the following values

	Tp	Fp	Fn	Tn	TPR	FPR
FWC	**11**	**2**	**1**	**34**	**0.85**	**0.91**
AS	**10**	**1**	**2**	**35**	**0.9**	**0.028**
C	**11**	**1**	**1**	**35**	**0.91**	**0.028**
CN	**11**	**1**	**1**	**35**	**0.91**	**0.028**

Table 3: Tp, Fp, Tn, TPR, FPR

	Precision	Recall
FWC	0.85	0.91
AS	0.9	0.83
C	0.91	0.91
CN	0.91	0.91

Table 4: Precision, Recall

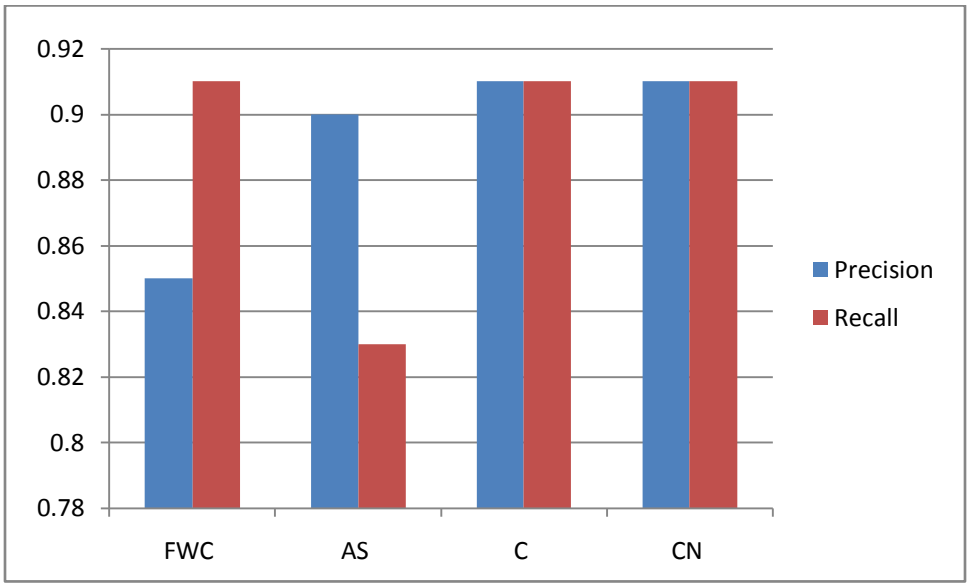


Figure 12: Precision and recall graphical presentation

As we observe in the Figure 12, the precision rate is obtained around 0.89%. The high value of the precision, confirms that our approach is rigid and robust. In terms of cloud classification the precision and recall value is of great importance to acquire the expected accuracy.

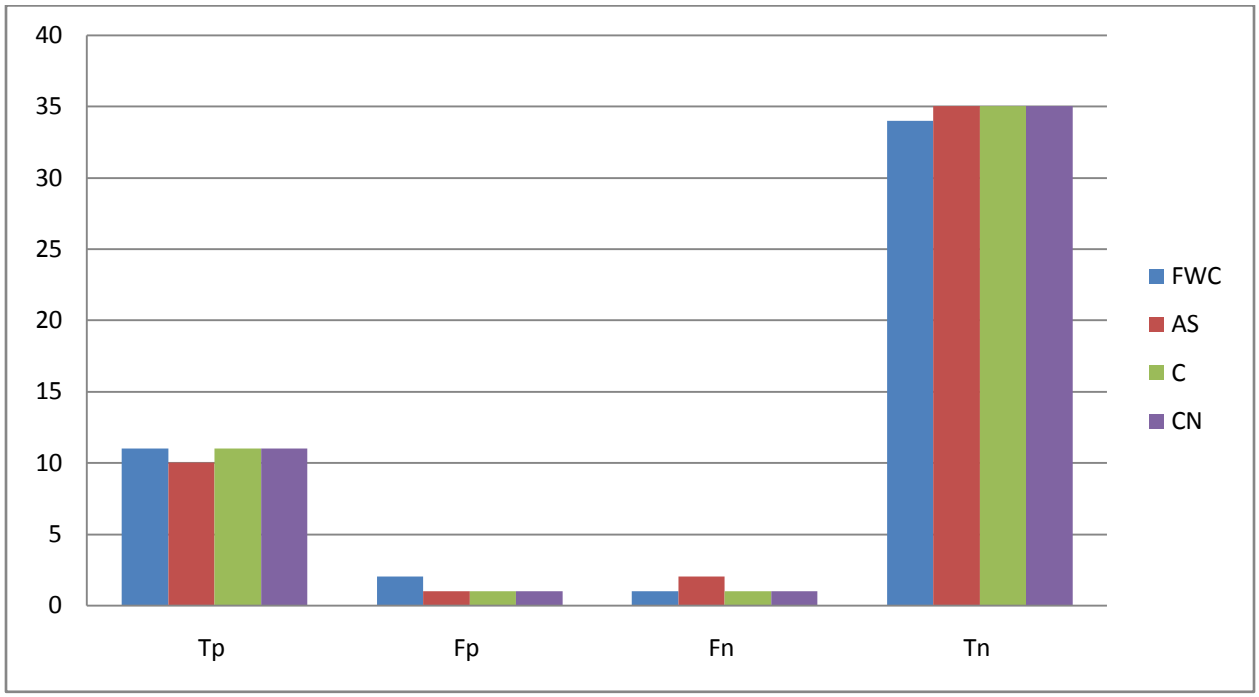


Figure 13: Accuracy on the basis of TP,FP, TN, FN

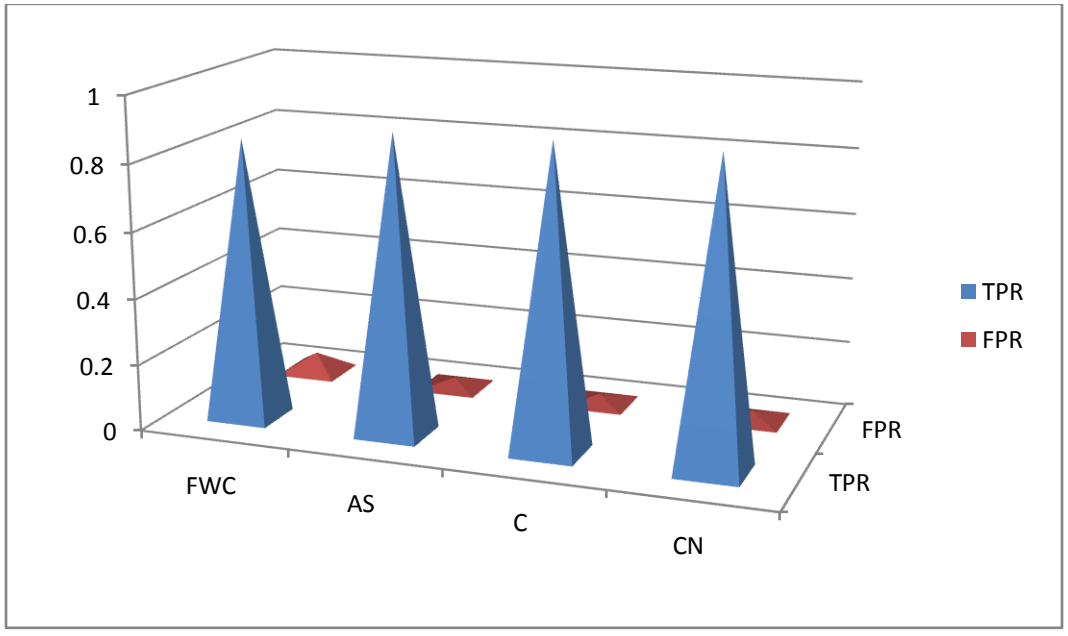


Figure 14: Comparison between TPR and FPR

7.3 Analysis

The analysis of the results using precision and recall have been shown in the above section.The average accuracy of other existing cloud classification method is 50% to 75% whereas this proposed method has up to 89%. The performance is almost 15% better. After doing the training using BPNN, the performance grows this much. The performance can be increased by adding some more features.

8. CONCLUSION

This paper proposed a classification of clouds and according to the type we can conclude about the characteristics of clouds, whether it is rainy or not. This system contains dataset, preprocessing, feature extraction, classification and also training. The system takes input images and after processing come to the conclusion about the clouds. It can be used as a supporting device to the satellite assessment. The experimental results applied on 48 images show that (on average) overall accuracy of the proposed system is 89% respectively.

9. REFERENCES

[1] K. Richards and G.D. Sullivan," Estimation of Cloud Cover using Color and Texture" Intelligent Systems Group, University of Reading, RG6 2AY,2006.

[2] J.A. Parikh and A. Rosenfeld, "Automatic Segmentation and Classification of Infrared Meteorological Satellite Data", IEEE Transactions on Systems, Man, and Cybernetics, SMC-8, No.10, 1978. pp 736-743.

[3] Z.Q. Go et al, "Comparison of techniques for measuring cloud texture in remotely sensed satellite meteorological image data", IEE Proceedings, Vol. 136, Pt F, No. 5, October 1989, pp 232-238.

[4] Igor A. C. Martins _ Pedro M. Ferreira and Antonio E. Ruano, "Estimation and Prediction of Cloudiness from Ground-Based All-Sky Hemispherical Digital Images. "

[5] C.A. Comes and A.W. Harrison, "Radiometric Estimation of Cloud Cover", Journal of Atmospheric and Oceanic Technology, Vol. 2, 1984, pp 482-490.

[6] Malay K. Kundu and Priyank Bagrecha,"Color Image Retrieval Using M-Band Wavelet Transform Based Color-Texture Feature".

[7] Kuo-Lin Hsu, X. Gao, and Soroosh Sorooshian,"Rainfall Estimation Using Cloud Texture Classification Mapping."

[8] Liu Jian and Xu Jianmin," An Automated, Dynamic Threshold Cloud Detection Algorithm for FY-2C Images", National Satellite Meteorological Center, Beijing, 100081, China.

[9] Shou Yixuan, Li Shenshen, Shou Shaowen and Zhao Zhongming, "Application of a cloud texture analysis scheme to the cloud cluster structure recognition and rainfall estimation in a mesoscale rainstorm process", Advances in Atmospheric Sciences, Science Press, copublished with Springer-Verlag GmbH, 0256-1530 (Print) 1861-9533 (Online), Volume 23, Number 5 / October, 2006, 767-774.

[10] Wei Shangguan; Yanling Hao; Zhizhong Lu; Peng Wu, "The Research of Satellite Cloud Image Recognition Base on Variational Method and Texture Feature Analysis", Industrial Electronics and Applications, 2007. ICIEA 2007. 2nd IEEE Conference on Volume, Issue, 23-25 May 2007.

[11] A. Heinle1, A. Macke2, and A. Srivastav1, "Automatic cloud classification of whole sky images" The Future Ocean", Department of Computer Science, Kiel University, Kiel, Germany, Published in Atmos. Meas. Tech., 6 May 2010.

[12] K. Richards and G.D. Sullivan," Colour and Texture in Cloud Identification: An Experimental Comparison of Neural Network and Bayesian Methods" Intelligent Systems Group, University of Reading.

10 .APPENDIX

10.1 Installation and User Guidelines

JAVA Download and installation in Windows

Downloading and installing Java is easy and free. There are a couple ways by which you can get Java for Windows

- Online download
- Offline download

Online

Manual installation downloads an IFTW (Install from the Web) executable program file and requires minimum user intervention. When you run this program, it fetches all the required files from the web, so you must remain connected to the Internet during the installation.

Administrative permission is required in order to install Java on Microsoft Windows. Before you proceed with online installation you may want to disable your Internet firewall. In some cases the default firewall settings are set to reject all automatic or online installations such as the Java online installation. If the firewall is not configured appropriately it may stall the download/install operation of Java under certain conditions.

- Go to the Manual download page in www.java.com.
- Click on **Windows Online**
- The File Download dialog box appears prompting you to run or save the download file
    - To run the installer, click **Run**.
    - To save the file for later installation, click **Save**. Choose the folder location and save the file to your local system. Tip: Save the file to a known location on your computer, for example, to your desktop.
      Double-click on the saved file to start the installation process.

- The installation process starts. Click the **Install** button to accept the license terms and to continue with the installation.

Oracle has partnered with companies that offer various products. The installer may present you with option to install these programs when you install Java. After ensuring that the desired programs are selected, click the **Next** button to continue the installation. A few brief dialogs confirm the last steps of the installation process. Click **Close** on the last dialog. This will complete Java installation process.

NOTE: You may need to restart (close and re-open) your browser to enable the Java installation in your browser.

Offline

Offline installation requires you to download an executable file available at the manual Java download page, which includes all the files needed for the complete installation at the user's discretion. There is no need to remain connected to the Internet during the installation. The file can also be copied to and installed on another computer that is not connected to the Internet. This process requires you to download an executable file that includes all the files needed for the complete installation. Administrative permission is required in order to install Java on Microsoft Windows.

- Go to the Manual download page in www.java.com.
- Click on **Windows Offline**.
- The File Download dialog box appears prompting you to run or save the download file Click **Save** to download the file to your local system. Tip: Save the file to a known location on your computer, for example, to your desktop.
- Close all applications including the browser.
- Double-click on the saved file to start the installation process.
- The installation process starts. Click the **Install** button to accept the license terms and to continue with the installation.

Oracle has partnered with companies that offer various products. The installer may present you with option to install these programs when you install Java. After ensuring that the desired programs are selected, click the **Next** button to continue the installation. A few brief dialogs confirm the last steps of the installation process; click **Close** on the last dialog. This will complete Java installation process.

NOTE: You may need to restart (close and re-open) your browser to enable the Java installation in your browser.

NETBEANS DOWNLOAD AND INSTALLATION

Starting the download

1. Go to http://download.netbeans.org/netbeans/7.1.2
2. In the upper right area of the page, select the language and platform from the drop-down list. You can also choose to download and use the platform-independent zip file.
3. Click the Download button for the download option that you want to install.
4. Save the installer file to your system.

To install the software:

1. After the download completes, run the installer.
    - For Windows, the installer executable file has the .exe extension. Double-click the installer file to run it.
    - For Solaris and Linux platforms, the installer file has the .sh extension. For these platforms, you need to make the installer files executable by using the following command: chmod +x <*installer-file-name*>
2. If you downloaded the **All** bundle, you can customize your installation. Perform the following steps at the Welcome page of the installation wizard:
    a. Click Customize.
    b. In the Customize Installation dialog box, make your selections.
    c. Click OK.

At the Welcome page of the installation wizard, click Next.

At the License agreement page, review the license agreement, click the acceptance check box, and click Next.

At the JUnit License Agreement page, decide if you want to install JUnit and click the appropriate option, click Next.

At the NetBeans IDE installation page, do the following:

1. Accept the default installation directory for the NetBeans IDE or specify another directory.

   Note: The installation directory must be empty and the user profile you are using to run the installer must have read/write permissions for this directory.

2. Accept the default JDK installation to use with the NetBeans IDE or select a different installation from the drop-down list. If the installation wizard did not find a compatible JDK installation to use with the NetBeans IDE, your JDK is not installed in the default location. In this case, specify the path to an installed JDK and click next, or cancel the current installation. After installing the required JDK version you can restart the installation.

If the GlassFish Server Open Source Edition 3.1.2 installation page opens, accept the default installation directory or specify another installation location.

If you are installing Apache Tomcat, on its installation page, accept the default installation directory or specify another installation location. Click Next.

At the Summary page, verify that the list of components to be installed is correct and that you have adequate space on your system for the installation.

Click Install to begin the installation.

At the Setup Complete page, provide anonymous usage data if desired, and click Finish.

Note: If you encounter problems successfully completing the software installation, see Troubleshooting for descriptions and suggested workarounds for unresolved issues that could affect the installation process.

After installing the software, start Netbeans. Click on File and open new project. Select java application and click on next. Add the project name, project location, project folder and click on Finish. Thus, new project gets created. Add the program files to that project. The class name should match with the filename. To run the program, open the file named svd.java and click on the run button.

10.2 SCREENSHOTS

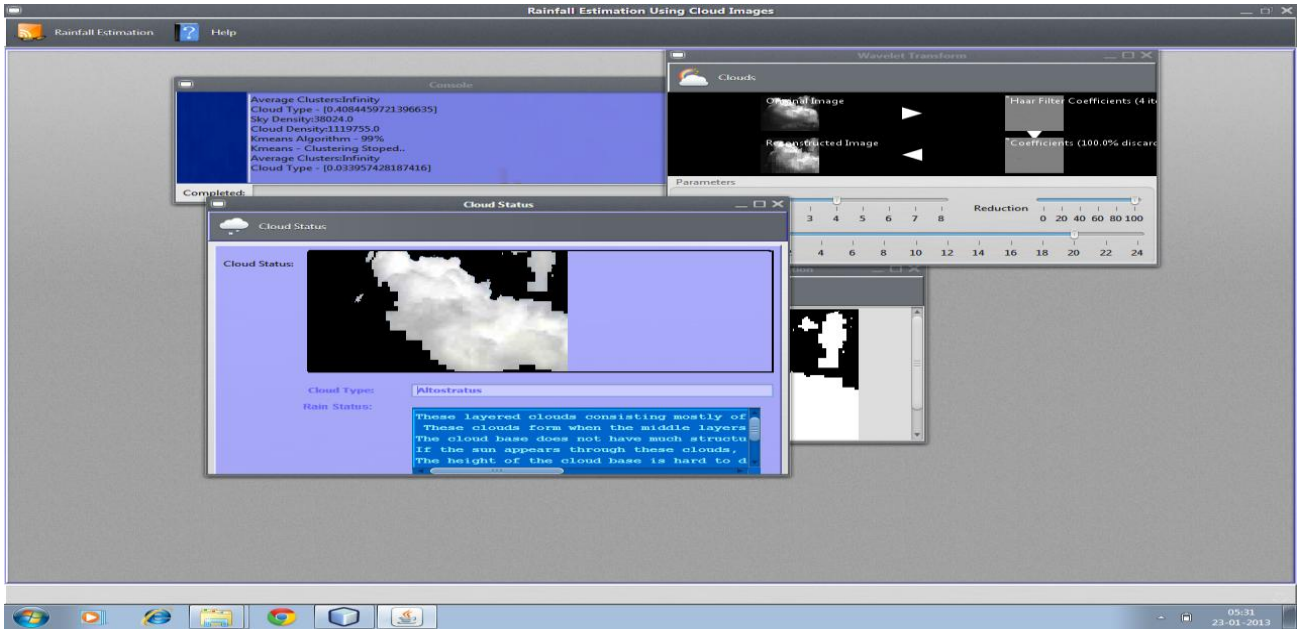


Figure 15: Altostratus cloud

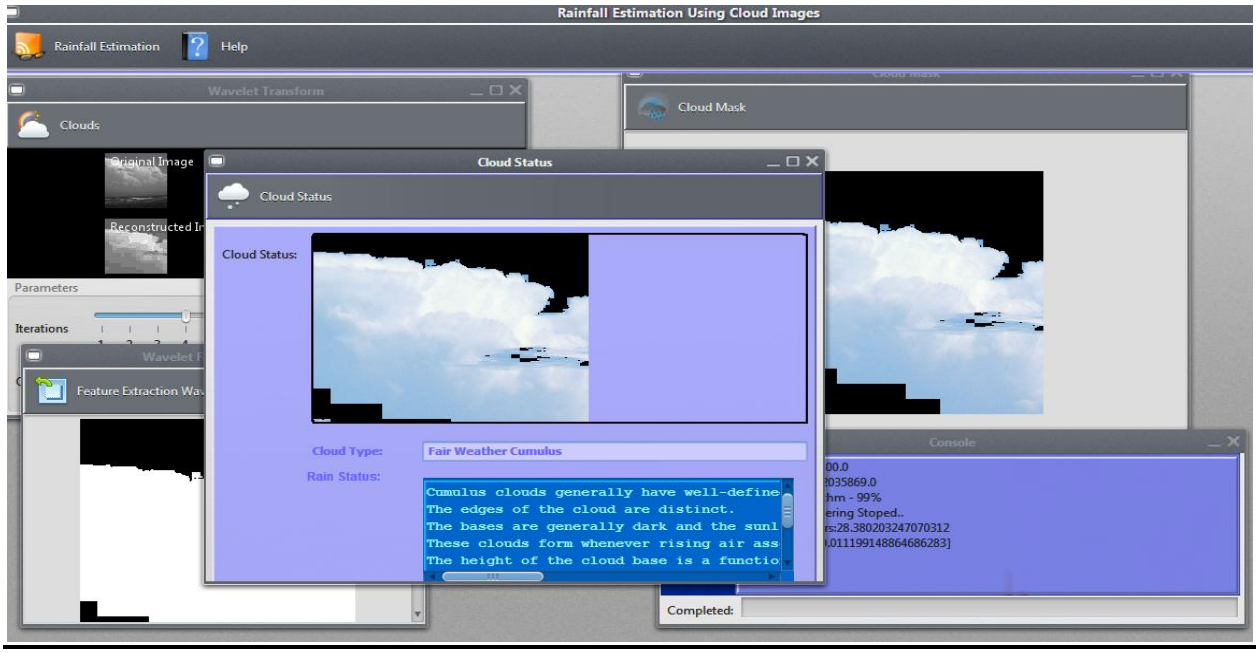


Figure 16: Fair weather cumulus

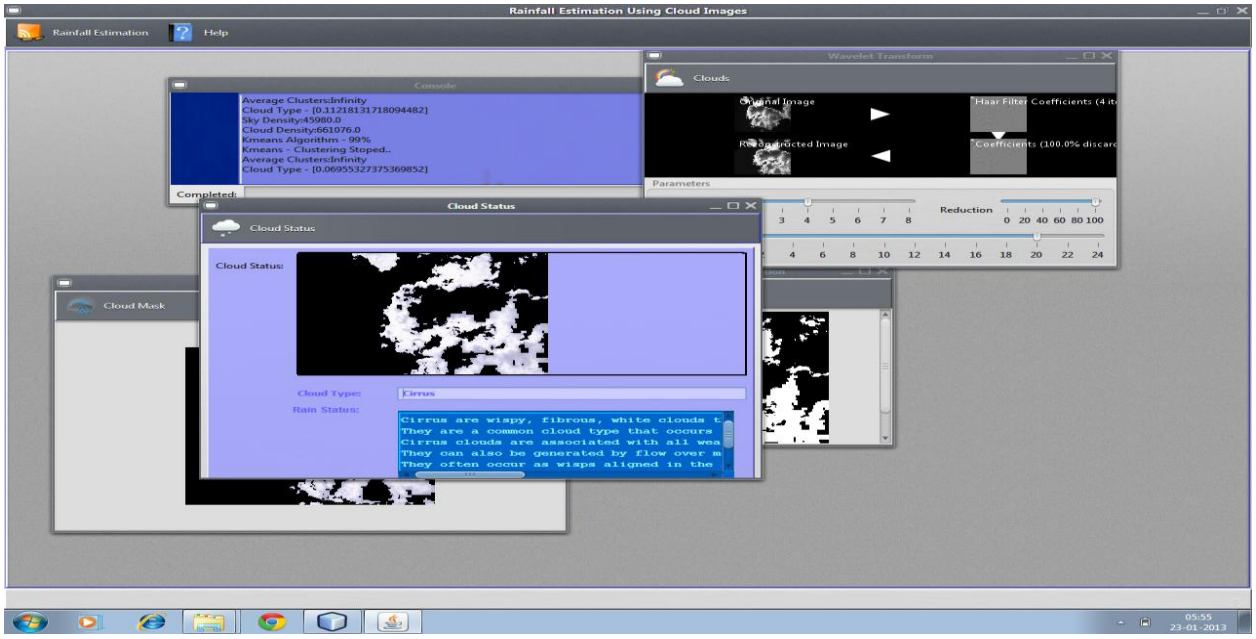


Figure 17: Cirrus cloud

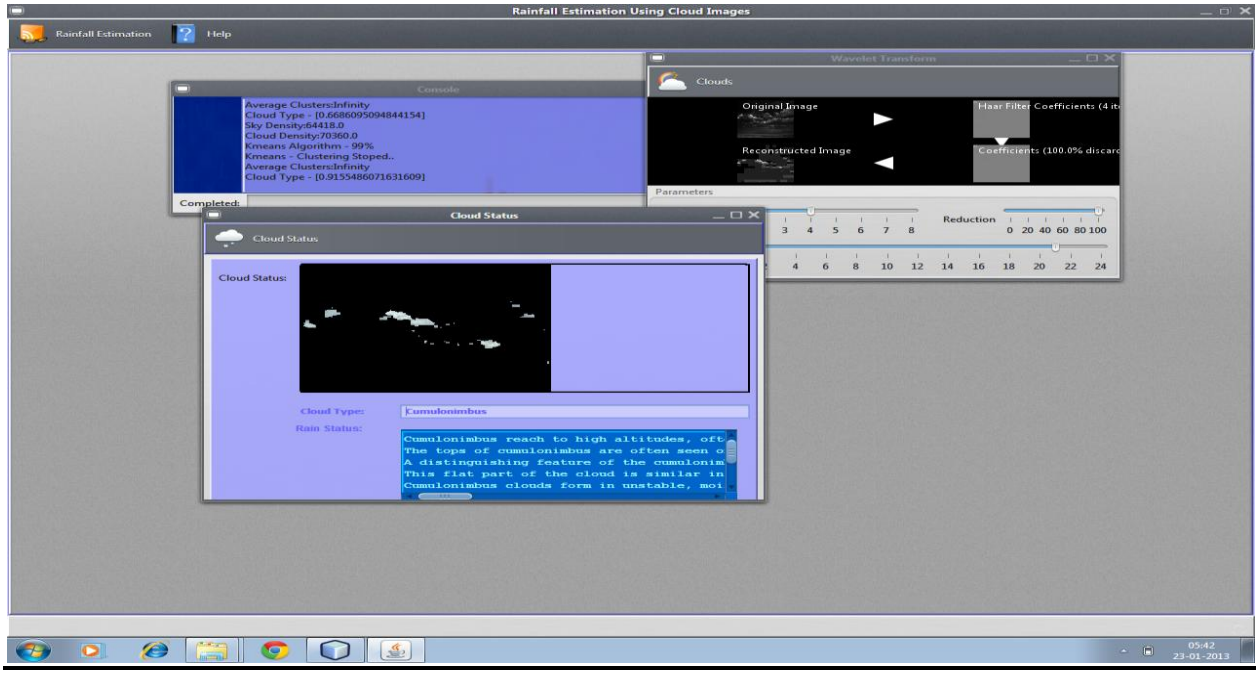


Figure 18: Cumulonimbus cloud

11. FUTURE WORK

The prime focus on extending this project is to enhance the classification criteria i.e. more features have to be extracted to conclude about the cloud characteristics. Features like temperature, wind direction, humidity and precipitation can be included to increase the performance. This can be achieved only with the help of satellite assessment of clouds. The accuracy can also be increased by using other transforms like curvelet, contourlet etc.