Image Processing and Analysis of Clouds Using Pattern Recognition for Supporting Weather Forecasting

Daniel Kellar

An undergraduate thesis submitted in partial fulfillment of the requirements for COSC4235

Algoma University

Thesis Advisor: Dr. Miguel Garcia-Ruiz

Abstract

Object recognition has become an enormous area of study in the field of computer vision allowing for the automated detection of known objects. The applications of this technology are vast, and this thesis looked to examine the application of a well-known approach to object detection as a method of automated identification of cloud formations: Haar classification. Using this technology, an application was developed which attempted to identify clouds within an image. The purpose of the research was to investigate the feasibility of Haar classification as a tool which could be used by weather watchers and forecasters for producing updated weather forecasts.

Acknowledgements

I would like to thank my friends and family for their support and encouragement through the process of researching and developing this thesis.

I would also like to thank my supervisor, Dr. Miguel Garcia Ruiz for his guidance and assistance in researching this topic, and Dr. Michael Biocchi his feedback as the second reader for this paper.

Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.0 Introduction

Current weather prediction models rely heavily on data captured from a variety of sources, including: satellites, radar, air craft, and various forms of ground based weather stations. This data is collected in a centralized location where it is then used to make weather and forecasting models.

Satellite and radar imagery are very powerful tools employed by forecasters, but they suffer from very distinct reliability issues. For example: satellites can produce a large scale image of cloud cover, but fail to deliver an accurate image of what is happening within those clouds, especially in the lower levels of cloud formations[1]. Doppler radar is susceptible to interference which can lead to inaccurate results.

To overcome these and many other issues, weather forecasters rely on weather watchers: people and groups who report ground based observations of weather events and conditions. Weather watchers gather data based on the current conditions and provide this data to forecasters and agencies (such as Environment Canada). With this data, forecasters are able to provide updated weather forecasts and issue weather related warnings and watches.

The goal of this thesis was to develop an automated ground based system for identifying cloud formations as a tool to potentially be used by weather watchers. This system uses captured images of the sky then uses pattern recognition to determine the type of clouds present in that image (if any such clouds are present). Such a system could be lightweight and fast enough for ease of use as well as portability.

1.1 Problem

Weather forecasting involves collecting data from various sources, such as Doppler radar, satellite imagery, wind speed and direction, and more. These forecasts provide a reliable means to predict weather patterns over a fairly large time period (Metro UK claimed that in 2014 they were able to provide a forecast for up to four days within some calculated error [2]). These forecasts, however, often cover large land masses (such as the City of Sault Ste. Marie or Algoma Manatoulin). As a weather system moves across these regions it can change (for example the severity of a thunderstorm can intensify, or rain can become freezing rain, and so on). Due to the ever this instability of natural weather events, weather forecasters have recruited weather watchers – people who collect data about weather activity in a region – which allows forecasters to produce updated regional forecasts.

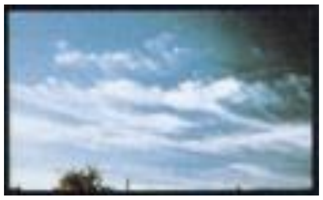
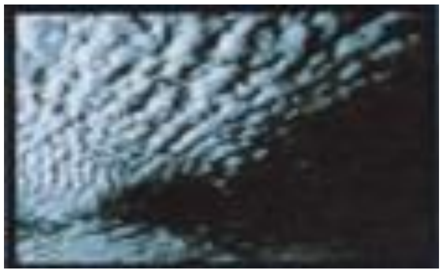
1.2 Literature Review

Clouds and cloud formations have been known for a long time to be indicators of weather activity. Clouds are categorized by ten classifications called Genera. These ten Genera are subcategorized based on altitude of the clouds in the atmosphere (table 1.2.1).

The use of technologies such as sky-imaging allows for capture of continuous cloud cover on a local scale. [1]

Storm Spotters are invaluable resources in forecasting severe and non-severe weather events. Spotter Networks employ position based reporting which allows for real time tracking of such weather events. [3]

Table 1.2.1 – Cloud classifications and associated precipitation [4]

High Level (altitude > 6 Km)		
Cirrus – No precipitation 	Cirrocumulus – No precipitation	Cirrostratus – No precipitation
Middle Level (altitude 2.5 Km to 6 Km)		
Altocumulus – light showers 	Altostratus – rain or snow	Nimbostratus – heavy rain or snow

Low Level (altitude < 2.5 Km)		
Stratocumulus - drizzle	Stratus - drizzle	Cumulus – showers or snow
Cumulonimbus – shower or snow. Also potential for thunder/ lightning and squalls		

Weather prediction models have grown to the point where forecasts can be made in the short term on small scales, to seasonal forecasts over much larger regions. As computers and numerical weather models continue to evolve, it is possible that a limit will be reached of what these computers can and cannot process with respect to weather prediction. [5]

Storm prediction centres in Canada each cover roughly 1 million square kilometres [6]. With an estimated 500 volunteer weather watchers in Canada [6] this places responsibility for a large area on a small number of people.

NIMROD is a system that was developed in in the 1990's that was used to generate forecasts on a very short term [7]. The NIMROD System was ground based which collected information on several different metrics: including precipitation, general cloud cover, and visibility.

Automated cloud classification is a technique by which an image is analyzed and it is determined whether or not that image contains clouds, and what type of clouds formations they are.

Haar like features (figure 1.2.1) were initially described in 1998 as a method of feature extraction of images in facial detection software based on Haar wavelets [8]. This research was further developed and Haar like features were formally defined in 2001 [9]. In 2002 more features were added [10]. These features are designed as an abstract view of common facial features (figure 1.2.2).

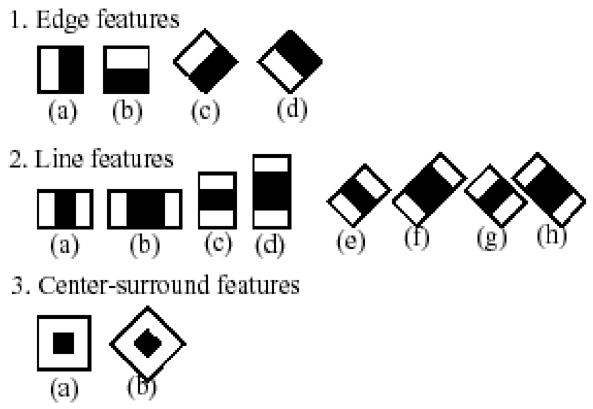


Figure 1.2.1 – Collection of Haar like features [11]

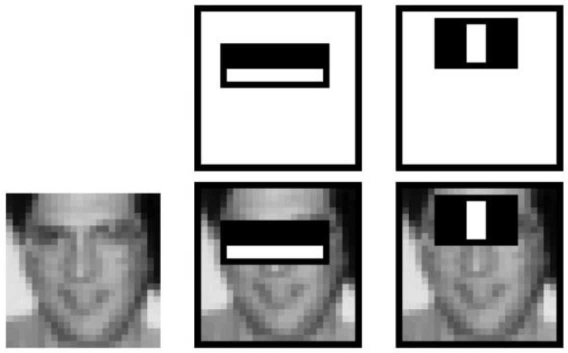
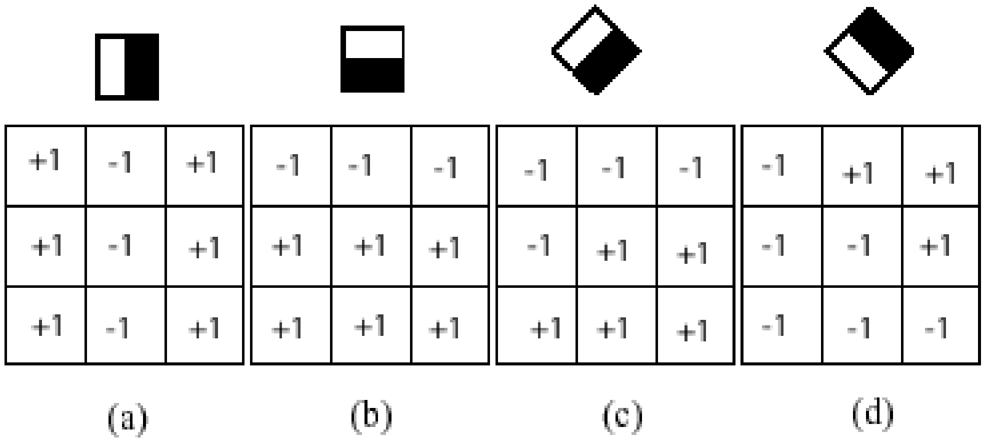


Figure 1.2.2 – Haar like features overlaying a face [12]

These features can be defined by a convolution matrix (kernel) (figure 1.2.3), where features of the image are determine by the convolution with the kernel [13]. Figure 1.2.4 depicts how such a feature may look overlaying a cloud where the light region of the feature corresponds to a region of the cloud; the dark region of the feature corresponds to a region outside of the cloud (in this case, the sky in the background).



$$\begin{array}{|c|c|c|}\hline +1 & -1 & +1 \\ \hline +1 & -1 & +1 \\ \hline +1 & -1 & +1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline +1 & +1 & +1 \\ \hline +1 & +1 & +1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & +1 & +1 \\ \hline +1 & +1 & +1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline -1 & +1 & +1 \\ \hline -1 & -1 & +1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

(a) (b) (c) (d)

Figure 1.2.3 – examples of convolution matrices

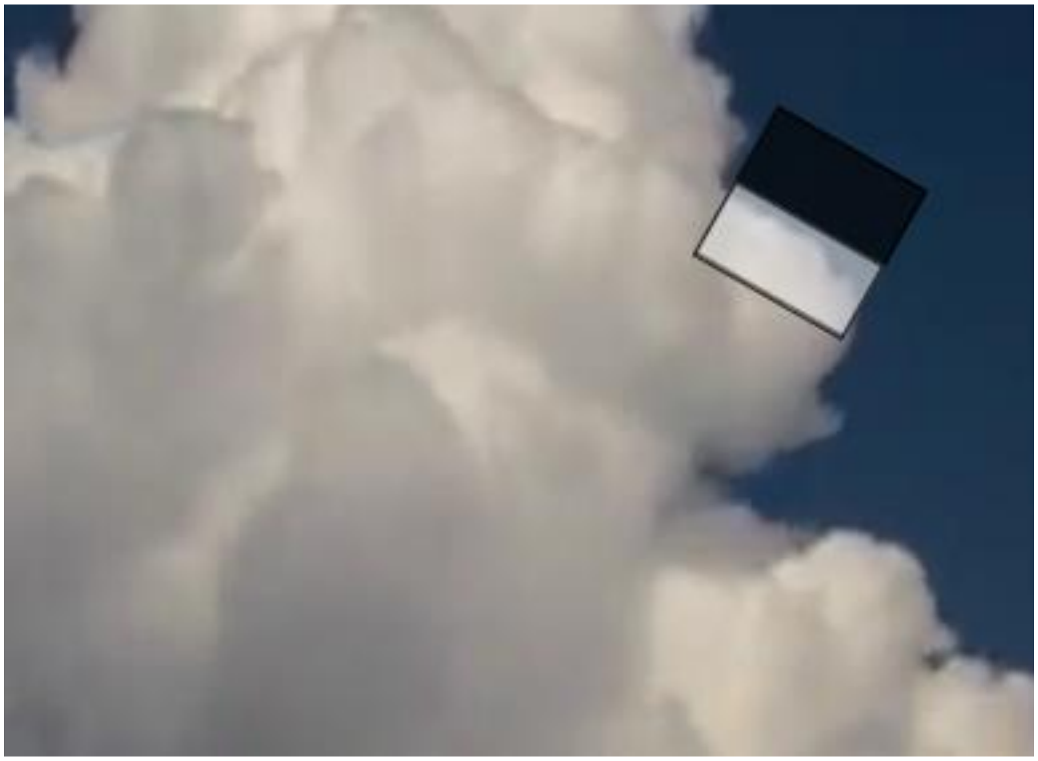


Figure 1.2.4 – a cloud with a Haar like feature overlaid

1.3 Objectives

There were two main objectives for this thesis:

i. To research existing systems of similar nature or otherwise relevant to the area of study, and

ii. To determine if Haar-like features can be used to identify and classify cloud formations

1.4 Rationale

The reason that I chose to research this topic is that I am interested in approaches used in weather research and forecasting. Through this research, I hope to find new ways to track weather events which will hopefully aid in a community's ability to prepare for these events.

I have chosen to use a machine learning approach because it offers a solution that is largely automated with the ability to recognize cloud formations that it is unfamiliar with using minimal human interaction. Haar like features were selected because the technique of using these features in pattern recognition has grown to incredible heights over the past decade. This technique has, in previous research covering various applications, proven to be very reliable and fast.

As previously discussed, weather watchers play a pivotal role in producing updated weather forecasts. The resultant software developed for this thesis was intended to be fast, which is achieved through the use of Haar classifiers, and lightweight. The software used multiple libraries for preparing the images and training the classifier, as I will discuss in more depth later in this paper, but the software for actually detecting and classifying clouds is dependent on a smaller set off tools which require very little effort to set up. This was intended as a quick and reliable method for weather watchers to gather more data on current weather events.

1.5 Research Questions

What are the limitations of Haar classifiers?

Can a classifier based on Haar-like features be used to identify and classify cloud formations?

Is such an application beneficial in any way?

1.6 Scope

This thesis was concerned only with the research and development of an application for ground based analysis of clouds. The focus of this research was on the ability to identify and classify these clouds.

Since the focus of this research was on identification of clouds formations with a focus on the application in a weather reporting system, some other important information is not being gathered, but could potentially be added in future iterations of this project. For example: real time temperature monitoring, and wind speed and direction.

The research was also focused primarily on identifying clouds that produce weather events (such as rain, thunderstorms, and blizzards). In particular mid to low level clouds are often associated with precipitation, with Cumulonimbus being the primary focus of this research.

This research relied on existing technologies for pattern recognition, such as the classifiers provided by the OpenCV library. Though the techniques were researched, the focus of this research did not include the low level details about how those technologies worked, nor how they are implemented.

Chapter 2: Methods

2.0 Introduction

This chapter will introduce the software development methodology used in this thesis, the staged delivery model, providing an overview of that methodology and how it was applied in this research. Details about what software and hardware was used will be covered along with how those tools were employed. The final section of this chapter will cover how the software was implemented.

2.1 Methodology

The software was developed following a staged delivery model (figure 2.1.1). This began with the initial concept for the software – a system for identifying could types. This system's intended purpose was to use a machine learning approach, involving image processing and pattern recognition, to capture “live” images of clouds and determine what type of cloud was being observed.

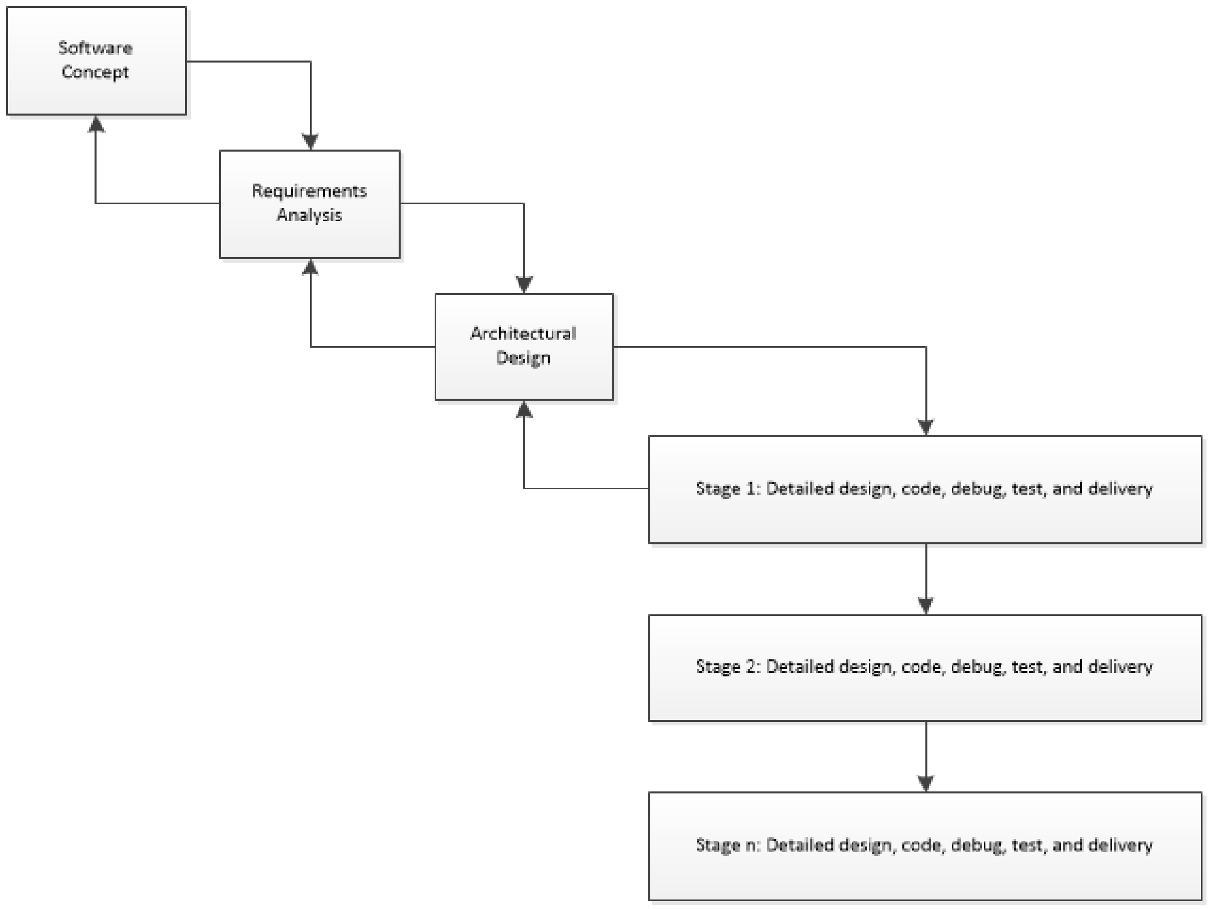

Software Concept
Requirements Analysis
Architectural Design
Stage 1: Detailed design, code, debug, test, and delivery
Stage 2: Detailed design, code, debug, test, and delivery
Stage n: Detailed design, code, debug, test, and delivery

Figure 2.1.1 – Staged delivery model [14]

The next phase of the model was analyzing requirements. The initial requirements were: train the system using known pattern recognition techniques based on a set of categorized images (reference images), capture images of clouds and determine what type those clouds were. It was determined that, due to time constraints, collecting enough reference images to train the classifier would not be feasible, since this process can require hundreds or even thousands of images with and without cloud formations in them. Through this analysis an alternative approach was considered: extract still image sequences from time-lapse videos of clouds.

The final stages of this methodology were to create detailed designs and implement, debug, and test these designs. There were two iterations of this stage: the goal of the first was to produce a simple prototype which would be able to identify clouds in images, but not classify those clouds by type. The second iteration was a refinement of the prototype which would be able to classify clouds based on type.

2.2 Materials

2.2.1 Software:

The application was developed use the Python language (version 2.7) using the Open Computer Vision Library (OpenCV) and the Python Image Library (PIL) – which has since been superseded by a “fork” of the original PIL source, called Pillow.

Images were taken from time lapse videos, exported to image sequences using Adobe After Effects and Adobe Media Encoder.

2.2.1.a About OpenCV

OpenCV (http://opencv.org/) is an open source computer vision library with binding in several languages including C++ and Python. OpenCV was used throughout this research for the following tasks:

- Generating samples
- Training the classifiers
- Opening the classifiers and test images and detecting objects in the test images

2.2.1.b About Python Imaging Library

Python Imaging Library (http://www.pythonware.com/products/pil/) is an open source library for Python which provides the ability to manipulate images (such as by altering individual pixels). PIL was used in this research to convert images to greyscale (from RGBA).

2.2.2 Hardware:

Development and testing of the software was completed on two desktop machines: one machine was used for creating samples and generating the classifier, and another machine was used to gather test data (this will be discussed later in this chapter as well as in the following chapters on the results and discussion). The technical specifications are outlined below:

Desktop 1 – used create samples and train classifier:

- Operating System – Ubuntu Server 15.10
- CPU – Intel Core 2 6300 @ 1.86 GHz
- Memory – 2 GB

Desktop 2 – used to gather test data:

- Operating System – Ubuntu 15.10
- CPU – Intel Core i7-3770 @ 3.40 GHz
- Memory – 8 GB

2.2.3 Classifier:

The goal of this thesis was to investigate the use of classifiers based on Haar-like features. The topic of such classifiers themselves has been researched to great extents and many tools exist for generating the classifier. Thus, this research used the utilities provided by the OpenCV library, namely the opencv_createsamples and the opencv_traincascade utilities.

2.2.4 Images:

In order to train the classifier two sets of images were required: a set of “positive images” which were of the object which was to be identified by the classifier - clouds, and a set of negative images which contained anything but the object which was to be identified. These two sets needed to be very large (Leindhart et Al describe using a total of approximately 8000 images [5000 positive; 3000 negative] in training a Haar classifier for use in facial detection [9]). Due to time constraints images used in this research were taken from time lapse video of clouds. This made it possible to get thousands of images using only a couple of minutes of video. Initial tests of training the classifier based on images taken from a small very number of videos yielded inaccurate results, therefore images were taken from a larger sample of videos. This is discussed in further detail later in the paper.

Once the training was completed another set of images were used to test the resultant classifier. This second image set contained both positive and negative images. This was intended as a way to better measure potential misclassification by the classifier.

2.3 Procedure

The software was broken into two iterations: a prototype which would simply determine if there was a cloud in an image, and a refined version of the prototype which would be able to classify any Cumulonimbus clouds found in images.

Before implementing any of the software, several tasks needed to be completed as a prerequisite (figure 2.1.1). These steps were automated through the use of several python scripts:

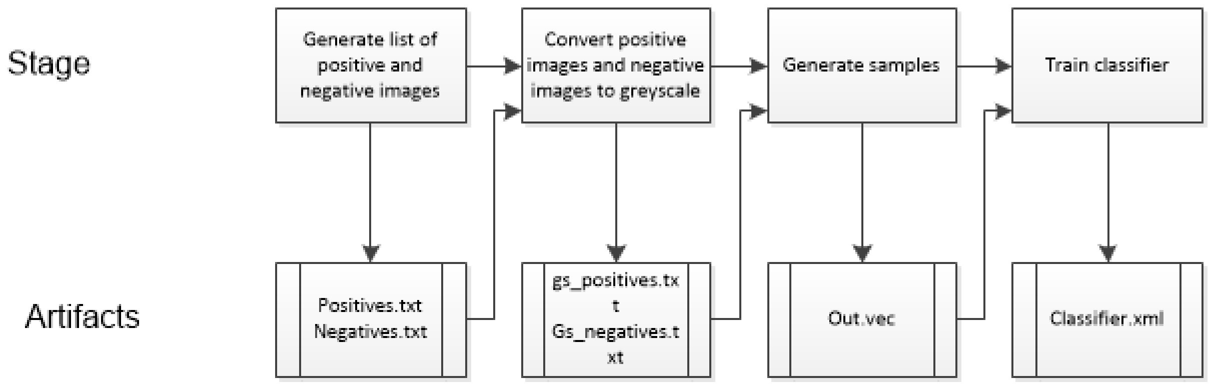


Figure 2.1.1 – A general overview of the procedures followed and the artifacts that were created and where they were used (note: images are omitted).

i) Locating and organizing images. Two sets of images are needed in order to train the classifier: a set of “positive” images which contain clouds, and a set of “negative” images that do not have clouds in them. These image sets were separated into two directories for further processing.

Images were taken from online time-lapse videos of cloud cover. This provided several benefits: a short video could produce a large image set (for example: a 60 second video at 30 frames per second could yield 900 images), the clouds were constantly changing shape and size over the duration of the video which provided a decent amount of variability to the image set. This was accomplished with Adobe After Effects and Adobe Media Encoder. First the videos were imported into After Effects then cropped down to eliminate as much of the background scenery as possible. The After Effects project was then imported into Media Encoder and exported to image sequences.

ii) Generate lists of the positive and negative images. To make the following processes easier, two files containing a list of the positive and negative images was created.

Due to time constraints and the computationally complex task of the following stages, a selection of 406 positive images and 822 negative images were used for training both for both the initial prototype and final software

iii) Convert to greyscale. Each of the positive and negative images needed to be converted to greyscale (from colour).

iv) Create samples from the positive and negative image sets. First text files were generated for both the positive and negative image sets. These text files simply contained a list of file names of the corresponding images (negatives.txt contained a list of the file names for all of the negative images). These text files were used by a Python script which called the opencv_createsamples (see appendix B.1) utility for each of the images in the positives file. The output of the create samples utility consisted of a “vector” file for each image, placed in a subdirectory. These vector files were then merged into a single vector file for use in the next stage.

v) Generate the classifier from the samples. The opencv_traincascade utility was used which took in the directory containing the vector files from the previous step, the text file containing the negative image names, and several other parameters (see appendix B.2).

Chapter 3: Results

3.0 Introduction

This chapter will present the results of testing the software. It will contain a brief overview of how the data was collected. It will also describe the raw data collected from the experiments and prepare the reader for the following chapter, Chapter 4: Discussion.

3.1 Main Findings

As mentioned previously, a set of images were selected for testing purpose and boxes bounding the clouds in the images were manually created (figure 3.1.1) and stored in text files. The software would run and create a box around each cloud it found in the image (figure 3.1.2) and write those boxes to a file using the same format as the files that were manually created. Testing was done in two phases – one for each iteration of the software (detecting clouds and identifying Cumulonimbus clouds)

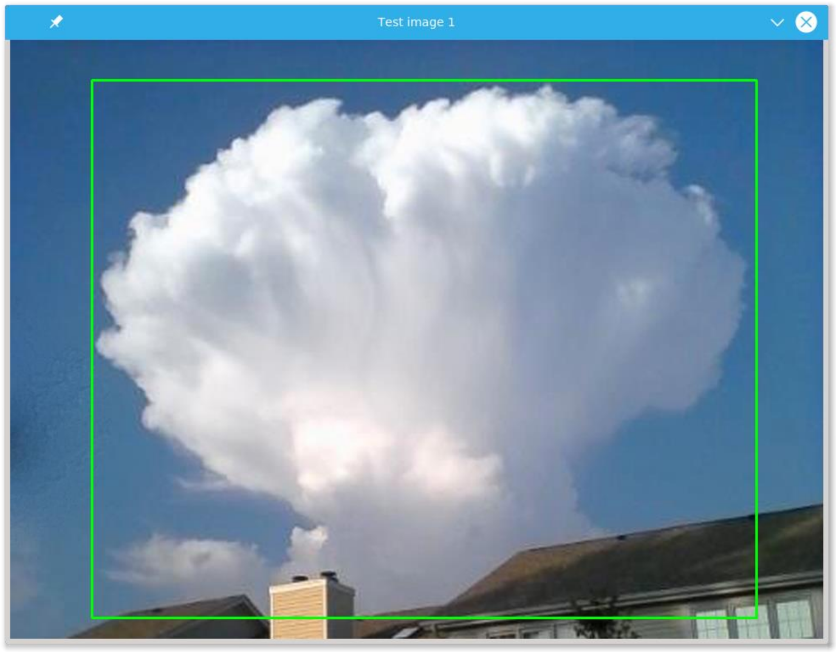


Figure 3.1.1 – image of a cloud with the expected bounding box outlined

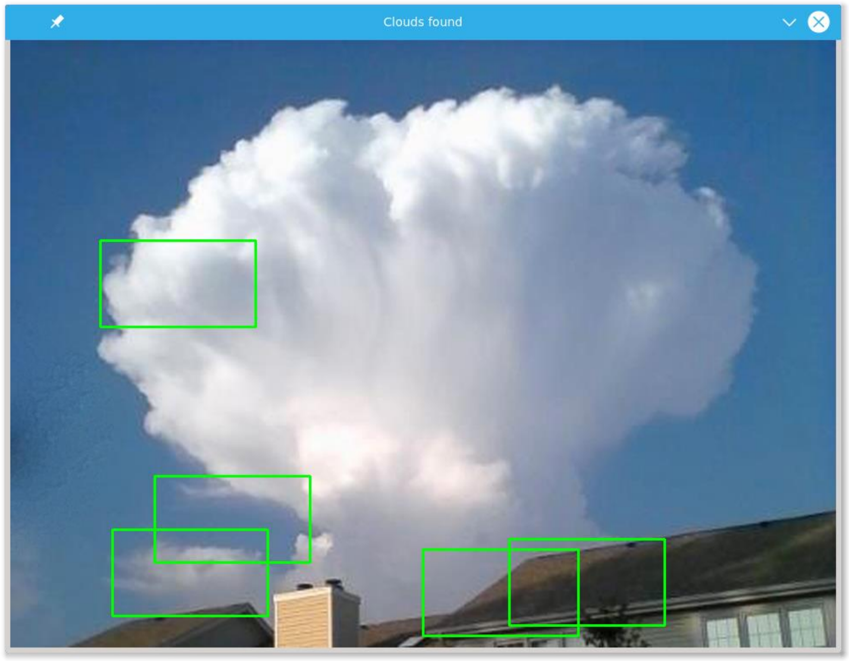


Figure 3.1.2 – image of a cloud with the detected “clouds” outlined

3.1.1 Cloud detection

Testing of the initial prototype used seventeen test images: twelve containing clouds and five which did not have clouds present in the image. Table 3.1.1 lists the expected and actual results for each image (see Appendix C: Raw Data for complete results) and figure 3.1.1 shows the same comparison for just the positive test images.

Table 3.1.1 – number of clouds expected to be detected vs cloud actually detected

Image name	Expected #of clouds to detect	#of clouds detected
Neg1.jpg	0	35
Neg2.jpg	0	15
Neg3.jpg	0	7
Neg4.jpg	0	16
Neg5.jpg	0	8
Pos1.jpg	1	68
Pos2.jpg	2	31
Pos3.jpg	6	2
Pos4.jpg	3	9
Pos5.jpg	1	12
Pos6.jpg	1	25

Pos7.jpg	1	20
Pos8.jpg	1	16
Pos9.jpg	1	56
Pos10.jpg	1	25
Pos11.jpg	1	45
Pos12.jpg	1	17

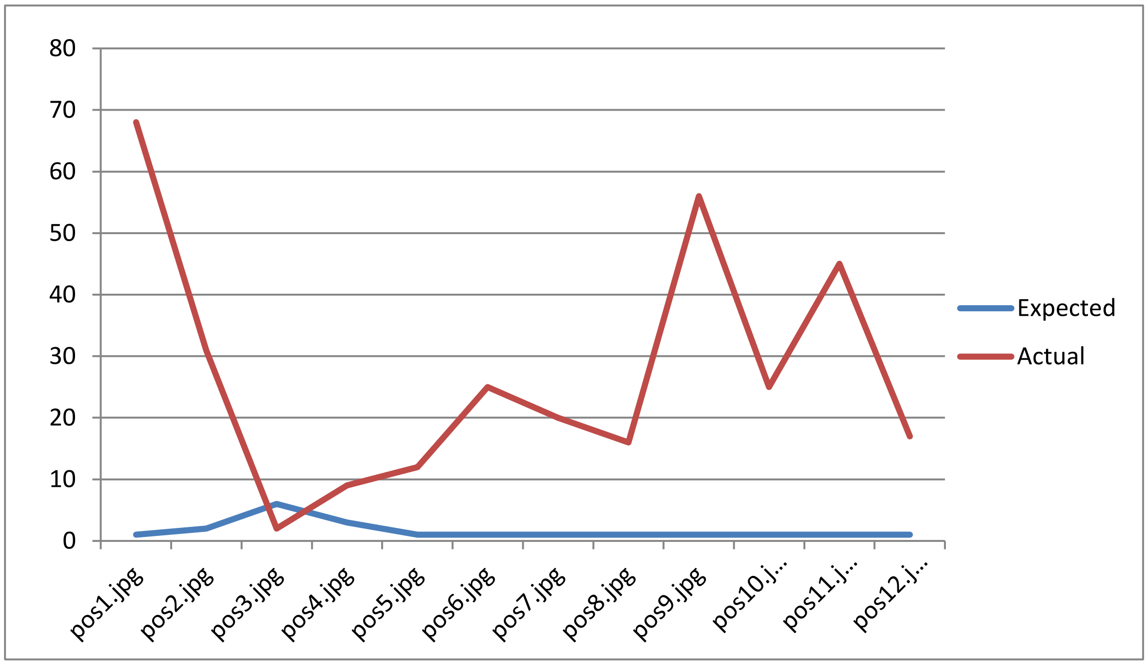


Figure 3.1.3 – number of clouds detected vs actual number of clouds in positive images

3.1.2 Cloud Identifier

Testing this iteration of the software used eighteen test images: ten containing Cumulonimbus clouds, eight which were a mixture of no clouds and clouds that are not

Cumulonimbus. For all test images, no clouds were identified. This will be discussed in the following chapter.

Chapter 4: Discussion

4.0 Introduction

This chapter will discuss in detail the results outlined in the previous chapter as well as summarize the research completed and provide insight into future research involving this topic. It will cover the areas in which the software failed and potential reasons for that failure.

4.1 Discussion of Results

It is quite clear that the results are not favourable. The first iteration of the software detected clouds in images that did not contain clouds which clearly proves it did not function properly. However, it was able to pick out parts of clouds in images that had fairly distinct features (this can be observed in figure 3.1.2; the things that it detected as being clouds are either actual parts of the cloud [primarily the edges] or the shadow of the cloud on the roof of the building), I believe that the failure of this can be attributed to the limited size and diversity of the training data.

The second iteration, I believe, failed outright for the same reasons that the initial iteration had difficulties. Though the number of positive and negative images used to train both was identical, it is my belief that the positive training set was not diverse enough.

Another issue that I believe to have contributed to these poor results is that some of the training images contained multiple distinct clouds. This could potentially lead to training the classifier to view multiple clouds as a single entity (figure 4.2.1)

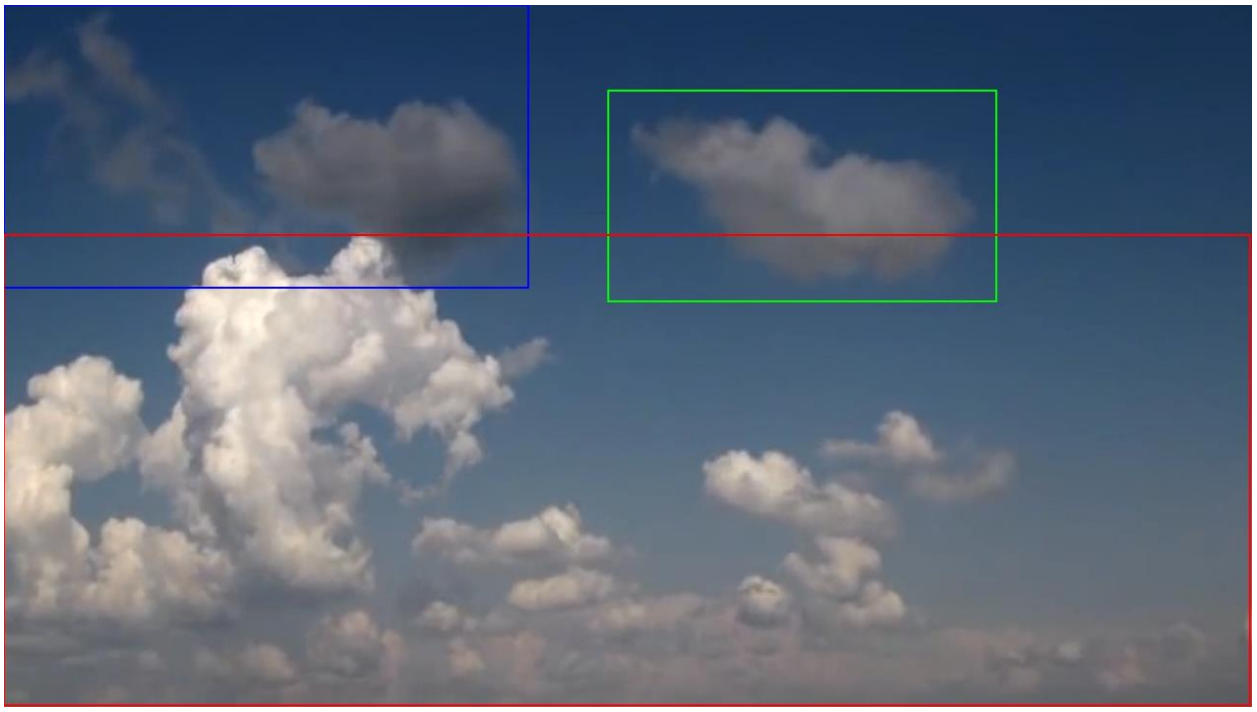


Figure 4.2.1 – a single image which was used in training depicting multiple clouds which could be treated as a single cloud during the training process

4.2 Conclusions

This research has explored an approach to automated cloud classification which employed a well-known object detection technique. Third party libraries (OpenCV and PIL) were used as a way to efficiently prepare the training data.

The project concluded with very discouraging results, but the implications are not so bleak. The first iteration yielded incredibly inaccurate and, in some cases, completely incorrect detection. This does not necessarily mean that Haar like classification is not a feasible approach to cloud identification. As I have mentioned previously, though the results are inaccurate they tended to be consistent in the misidentification and often identifying certain features as opposed to the entire cloud.

4.3 Future Work

The first and probably most obvious direction that could be followed from this research is to use a much larger data set containing a much wider diversity of the training images. This could potentially include testing whether or not including images of other cloud types as negatives for a certain cloud (for example: if training was being done to identify cumulonimbus clouds images of different types of clouds [such as cumulus or stratus] could be used as negative samples for that training) could have an impact on the reliability of the training.

As discussed, I do strongly believe that the issue of “bad” images impacted results. This could provide an area of research into whether or not this truly is an issue, and if so, to what extent.

References:

[1]J. Calbó and J. Sabburg, 'Feature Extraction from Whole-Sky Ground-Based Images for Cloud-Type Recognition', *J. Atmos. Oceanic Technol.*, vol. 25, no. 1, pp. 3-14, 2008.

[2]"How accurate are our public forecasts?", *Met Office*, 2016. [Online]. Available: http://www.metoffice.gov.uk/about-us/who/accuracy/forecasts.

[3]J.J Jans and C. Keen, 'Enriching the Modern Day Storm Spotter Through Technology & Education Enhancements' in Seventh Symposium on Policy and Socio-Economic Research, AMS Annual Meeting, New Orleans

[4]"Cloud Types and Precipitation", *Bom.gov.au*, 2016. [Online]. Available: http://www.bom.gov.au/weather-services/about/cloud/cloud-types.shtml. [Accessed: 29-Mar- 2016].

[5]P. Bauer, A. Thorpe and G. Brunet, 'The Quiet Revolution of Numerical Weather Prediction', *Nature*, vol. 525, pp 47-55, 2015

[6] S. Messenger, 'Computers predicting the storm - Canadian Geographic', Canadiangeographic.ca, 2015. [Online]. Available: http://www.canadiangeographic.ca/magazine/jf12/computers_predict_weather.asp

[7] B. Golding, *'Nimrod: A system for generating automated very short range forecasts'*, Meteorol. App., vol. 5, no. 1, pp. 1-16, 1998.

[8]C. Papageorgiou, M. Oren, T. Poggio, "A General Framework for Object Detection", in International Conference on Computer Vision, Bombay, India, 1998, pp. 555 – 562.

[9]P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 2001, pp. I-511-I-518 vol.1.

[10]R. Lienhart, A. Kuranov, V. Pisarevsky, "Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection", Lecture Notes in Computer Science, vol. 2781, pp. 297-304, 2002.

[11]http://docs.opencv.org/2.4/_images/haarfeatures.png

[12]http://www.s-schmitt.de/bilder/bilder/haar_features.jpg

[13] Pavani, Sri-Kaushik, David Delgado, and Alejandro F. Frangi. "Haar-Like Features With Optimally Weighted Rectangles For Rapid Object Detection". *Pattern Recognition* 43.1 (2010): 160-172. Web.

[14] S. McConnell, *Rapid development: taming wild software schedules*. Redmond, WA: Microsoft Press, 1996.

[15] "Cascade Classifier Training — Opencv 2.4.12.0 Documentation". *Docs.opencv.org*. N.p., 2016.

Appendix A - Source Code

```
#gen_pos_neg.py

#Generate the positive and negative lists

import os

img_dir = '../../thesis_images/'

pos_dir = img_dir + 'clouds/'

neg_dir = img_dir + 'background_images/'

pos_file = "positives.txt"

neg_file = "negatives.txt"

pos = open(pos_file, 'w')

neg = open(neg_file, 'w')

dirs = os.listdir(pos_dir)

for i in dirs:

        imgs_dir = pos_dir + i

        imgs = os.listdir(imgs_dir);

        offset = len(imgs) / 10

        for j in range(len(imgs)):

                #print(j)

                if j % offset == 0:

                        pos.write(imgs_dir + '/' + imgs[j] + '\n')

dirs = os.listdir(neg_dir)

for i in dirs:

        imgs_dir = neg_dir + i

        imgs = os.listdir(imgs_dir);

        offset = len(imgs) / 10

        for j in range(len(imgs)):

                #print(j)
                if j % offset == 0:
                        neg.write(imgs_dir + '/' + imgs[j] + '\n')
# greyscale.py
# convert images to greyscale and generate list of greyscale
# images

import os
import Image
pos = open("positives_gs_cropped.txt", "w")
sequences = "./image sequences"
gs = "./greyscale_images"
image_dirs = os.listdir(sequences)
count = 0
for i in image_dirs:          # i is a subdirctory of dir
        files = os.listdir(sequences + "/" + i)      # files is the list of files in dir/i
        if os.path.exists(gs + "/" + i) == False:
                os.mkdir(gs + "/" + i)
        for j in files:          # j is a .png file in dir/i
                img = Image.open(sequences + "/" + i + "/" + j).convert("LA")
                img.save(gs + "/" + i + "/" + j)
                #pos.write(j + "\n")
                #pos.write(dir + "/" + i + "/" + j + "\n")
        tmp = 0
dir = "./img/negatives"
negs = open("negatives_gs_cropped.txt", "w")
file = os.listdir(dir)
for i in file:
        negs.write(dir + "/" + i + "\n")

# generate_vec.py

# generate positive samples for each

# positive image

import subprocess

import string

pos = "./gs_positives_cumulo.txt"

#neg = './gs_negatives.txt'

neg = "test_negatives.txt"

file = open(pos, "r")

imgs = file.readlines()

count = 0;

dir = "../vecs/"

#w = "18"

#h = "10"

w = 16

h = 9

# Just clouds

#w = 16

#h = 12

for i in imgs:

        img = string.strip(i, "\n")

        subprocess.call(["opencv_createsamples", "-img", img, "-info", pos, "-vec", (dir +
"out" + str(count) + ".vec" ), "-bg", neg, "-num", "405", "-w", str(w), "-h", str(h)]);

        count = count + 1

import cv2

import os

def gen_results(img, casc):
        image = cv2.imread(img)
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        # Detect clouds in the image
        clouds = casc.detectMultiScale(
            gray,
            scaleFactor=2.0,
            flags = cv2.cv.CV_HAAR_SCALE_IMAGE
        )
        fname = os.path.basename(img)
        name, ext = os.path.splitext(fname)
        out_file = open("results/actual/" + name + ".txt", 'w')
        num_clouds = len(clouds)
        out_file.write(str(num_clouds) + "\n")
        # Draw a rectangle around the clouds
        for (x, y, w, h) in clouds:
                cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
                # write rectangles to .csv file
                out_file.write(str(x) + "\n")
                out_file.write(str(y) + "\n")
                out_file.write("0 0\n")
                out_file.write(str(w) + " 0\n")
                out_file.write("0 " + str(h) + "\n")
                out_file.write(str(w) + " " + str(h) + "\n")
        cv2.imshow("Clouds found", image)
        cv2.waitKey(0)
cascPath = "cloud_cascade.xml"
# Create the haar cascade
```

```
cloudCascade = cv2.CascadeClassifier(cascPath)

test_images = "test_images.txt"

test_images_file = open(test_images, 'r')

imgs = test_images_file.readlines()

gen_results("test_images/VickieIncus.jpg", cloudCascade)
```

Appendix B – Other information

B.1 opencv_createsamples parameters [15]

The below table outlines the parameters that the opencv_createsamples utility will accept:

Parameter	**Description**
-vec <vfname>	Output file for the positive samples
-img <ifname>	Source image (positive image)
-bg <bgfname>	File containing list of negative images
-num <numsamples>	Number of samples to generate
-bgcolor <bgcolor>	Background color of positive images. Background color is considered transparent
-bgthresh <bgthresh>	Threshold for background color of positive images
-inv	Invert colors of images
-randinv	Randomly invert colors of images
-maxidev <maxdeviation>	Maximum deviation of foreground pixel intensity
-maxxangle <max_x_rot>	Maximum x rotation applied to input image
-maxyangle <max_y_rot>	Maximum x rotation applied to input image
-maxxangle <max_z_rot>	Maximum x rotation applied to input image

-show	Display samples
-w	Width of output sample
-h	Height of output sample
-pngoutput	Generate collection of .png files and annotations

B.2 opencv_traincascade parameters [15]

Parameter	**Description**
-data <casc_dir>	Output directory for trained classifier
-vec <vec_file>	Positive samples (.vec file)
-bg <bg_filename>	List of background files (negative samples)
-numPos <num_positive>	Number of positive samples
-numNeg <num_negative>	Number of negative samples
-numStages <num_of_stages>	Number of stages to be trained.
-precalcValBufSize <buffer_size>	Feature value buffer size
-precalcIdxBufSize <idxs_size>	Feature index buffer size
-baseFormatSave	
-acceptanceRatioBreakValue	Precision of model

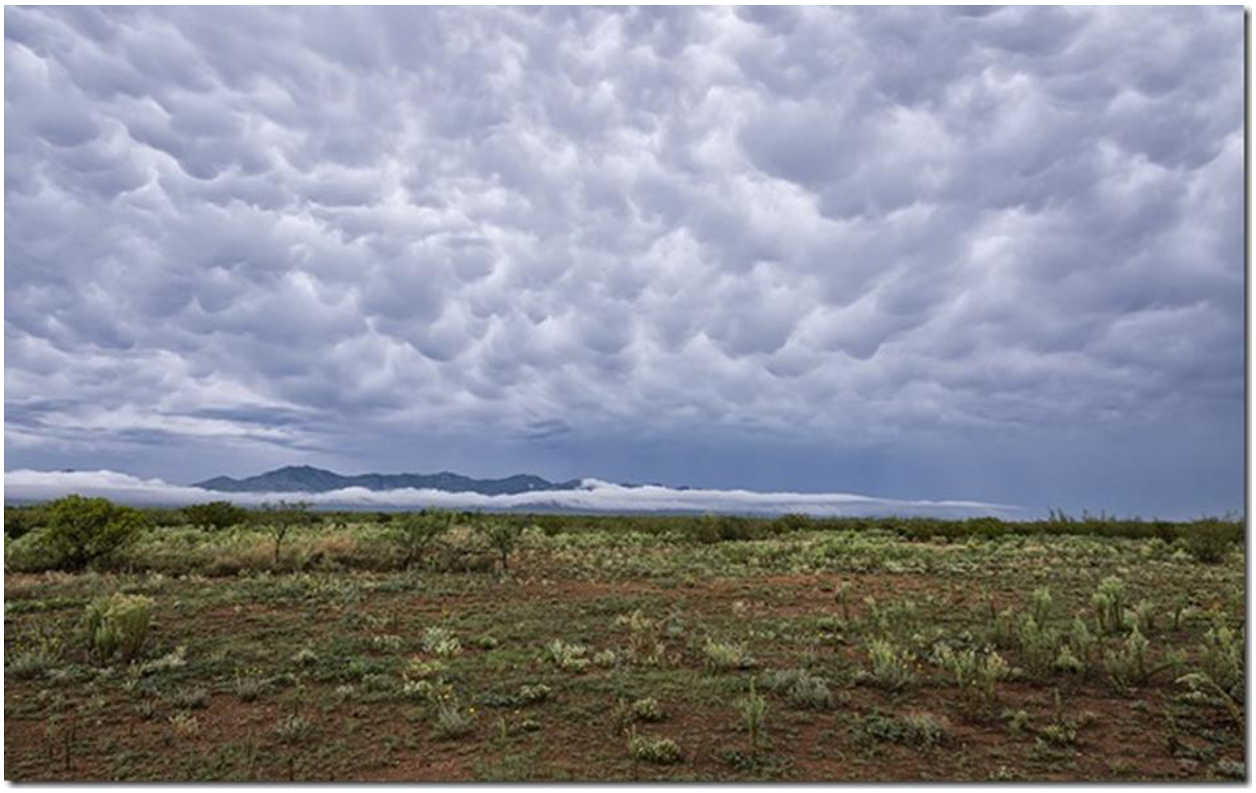
-stageType <BOOST>	Stage type (only BOOST is supported)
-featureType <{HAAR, LBP}>	Feature type (Haar like, Local Binary Patterns)
-w <sample_width>	Width of training samples
-h <sample_height>	Height of training samples
-bt <{DAB, RAB, LB, GAB}>	Boosted classifier type (Discrete AdaBoost, Real AdaBoost, LogitBoost, Gentle AdaBoost [default])
-minHitRate <hit_rate>	Minimum hit rate for each stage of classification
-maxFalseAlarmRate <alarm_rate>	Maximum false alarm rate for each stage of classification
-weightTrimRate <trim_rate>	Trim weight
-maxDepth <tree_depth>	Maximum depth of weak tree
-maxWeakCount <tree_count>	Maximum count of weak trees for each stage of classification
-mode <{BASIC, CORE, ALL}>	Type of Haar features to use in training. Default: BASIC

Appendix C: Test images

Below are the images used in testing the software. Posn.jpg are “positive” images (that is to say they have clouds), negn.jpg are negative images (which is to say that they do not have clouds in them)

C.1 Cloud detection

pos1.jpg



pos2.jpg

pos3.jpg

pos4.jpg

pos5.jpg

pos6.jpg

pos7.jpg

pos8.jpg

pos9.jpg

pos10.jpg

pos11.jpg

pos12.jpg

neg1.jpg

neg2.jpg

neg3.jpg

neg4.jpg

neg5.jpg

C.2 Cloud Identification

pos1.png

pos2.png

pos3.png

pos4.jpg

pos5.jpg

pos6.jpg

pos7.jpg

pos8.jpg

pos9.jpg

pos10.jpg

neg1.jpg

neg2.jpg

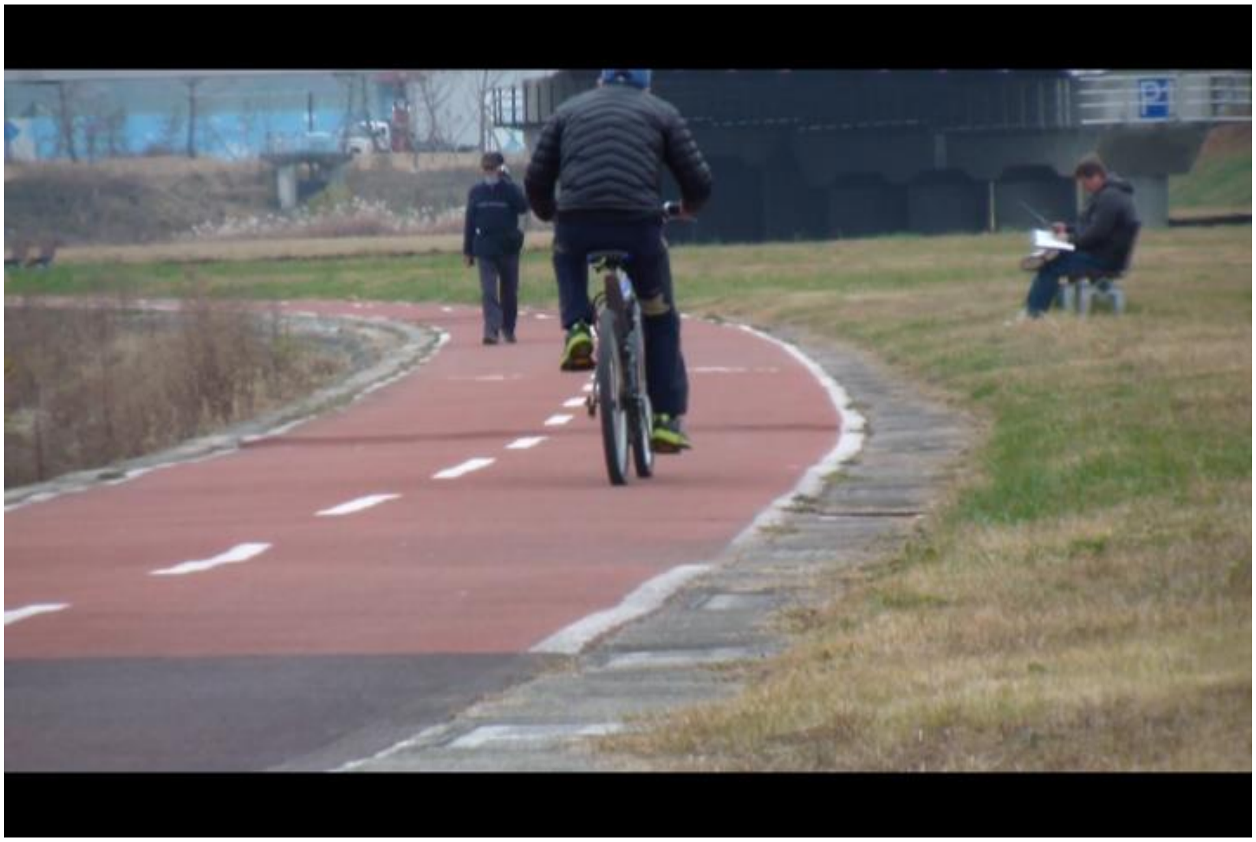
neg3.jpg

neg4.jpg

neg5.jpg

neg6.jpg

neg7.jpg



neg8.jpg

Appendix D: Raw data

The following is the output from the test results. Each entry begins with a file name (files containing “neg” correspond to images that do not contain clouds; files with “pos” correspond to images that contained clouds). The file name is followed by a line indication how many clouds were detected in that image. The lines following that correspond to the coordinates of the detected clouds in the following format:

xoff – x offset from the top left of the image to the top left of the detected cloud

yoff – y offset from the top left of the image to the top left of the detected cloud

then four (x,y) pairs (on a single line, delimited by commas) which correspond to the coordinates of the top left, top right, bottom left, bottom right bound of the detected cloud (these coordinates are 0 based therefore the actual coordinate would be x + xoff, y + yoff).

Cloud Detection – Expected results

```
i_detect/pos/pos1.jpg
1
0
0
(0,0),(951,0),(0,396),(951,396)
i_detect/pos/pos2.jpg
2
0
0
(0,0),(957,0),(0,437),(957,437)
0
436
(0,0),(957,0),(0,100),(957,100)
i_detect/pos/pos3.jpg
6
273
0
(0,0),(162,0),(0,242),(162,242)
315
241
(0,0),(176,0),(0,233),(176,233)
631
246
```

```
(0,0),(295,0),(0,216),(295,216)
168
549
(0,0),(51,0),(0,53),(51,53)
780
561
(0,0),(177,0),(0,75),(177,75)
390
537
(0,0),(116,0),(0,89),(116,89)
i_detect/pos/pos4.jpg
3
267
25
(0,0),(690,0),(0,401),(690,401)
180
207
(0,0),(96,0),(0,92),(96,92)
156
324
(0,0),(102,0),(0,124),(102,124)
i_detect/pos/pos5.jpg
1
96
47
(0,0),(782,0),(0,632),(782,632)
i_detect/pos/pos6.jpg
1
0
0
(0,0),(957,0),(0,522),(957,522)
i_detect/pos/pos7.jpg
1
0
0
(0,0),(957,0),(0,537),(957,537)
i_detect/pos/pos8.jpg
1
0
140
(0,0),(957,0),(0,419),(957,419)
i_detect/pos/pos9.jpg
1
0
0
(0,0),(957,0),(0,413),(957,413)
i_detect/pos/pos10.jpg
1
288
22
(0,0),(669,,0),(0,554),(669,,554)
i_detect/pos/pos11.jpg
1
0
```

```
0
(0,0),(957,0),(0,466),(957,466)
i_detect/pos/pos12.jpg
1
0
22
(0,0),(822,0),(0,473),(822,473)
```

Cloud detection – Actual results

```
i_detect/neg/neg1.jpg
35
270
167
(0,0),(36,0),(0,20),(36,20)
87
173
(0,0),(72,0),(0,40),(72,40)
150
168
(0,0),(72,0),(0,40),(72,40)
189
174
(0,0),(72,0),(0,40),(72,40)
328
56
(0,0),(72,0),(0,40),(72,40)
260
29
(0,0),(72,0),(0,40),(72,40)
15
265
(0,0),(72,0),(0,40),(72,40)
629
55
(0,0),(72,0),(0,40),(72,40)
326
26
(0,0),(72,0),(0,40),(72,40)
506
173
(0,0),(72,0),(0,40),(72,40)
213
318
(0,0),(72,0),(0,40),(72,40)
507
135
(0,0),(72,0),(0,40),(72,40)
536
107
(0,0),(72,0),(0,40),(72,40)
67
```

```
129
(0,0),(72,0),(0,40),(72,40)
259
180
(0,0),(72,0),(0,40),(72,40)
147
128
(0,0),(72,0),(0,40),(72,40)
105
133
(0,0),(72,0),(0,40),(72,40)
337
394
(0,0),(72,0),(0,40),(72,40)
482
384
(0,0),(72,0),(0,40),(72,40)
190
82
(0,0),(72,0),(0,40),(72,40)
452
134
(0,0),(72,0),(0,40),(72,40)
350
342
(0,0),(72,0),(0,40),(72,40)
352
186
(0,0),(72,0),(0,40),(72,40)
334
142
(0,0),(72,0),(0,40),(72,40)
177
137
(0,0),(72,0),(0,40),(72,40)
391
145
(0,0),(72,0),(0,40),(72,40)
337
102
(0,0),(72,0),(0,40),(72,40)
308
103
(0,0),(72,0),(0,40),(72,40)
551
218
(0,0),(72,0),(0,40),(72,40)
51
236
(0,0),(72,0),(0,40),(72,40)
415
215
(0,0),(144,0),(0,80),(144,80)
250
```

```
18
(0,0),(144,0),(0,80),(144,80)
61
36
(0,0),(144,0),(0,80),(144,80)
433
116
(0,0),(144,0),(0,80),(144,80)
484
146
(0,0),(144,0),(0,80),(144,80)

i_detect/neg/neg2.jpg
15
137
312
(0,0),(72,0),(0,40),(72,40)
101
317
(0,0),(72,0),(0,40),(72,40)
206
28
(0,0),(72,0),(0,40),(72,40)
578
327
(0,0),(72,0),(0,40),(72,40)
235
136
(0,0),(72,0),(0,40),(72,40)
437
59
(0,0),(72,0),(0,40),(72,40)
65
142
(0,0),(72,0),(0,40),(72,40)
583
173
(0,0),(72,0),(0,40),(72,40)
288
95
(0,0),(72,0),(0,40),(72,40)
473
103
(0,0),(72,0),(0,40),(72,40)
554
217
(0,0),(144,0),(0,80),(144,80)
8
320
(0,0),(144,0),(0,80),(144,80)
486
46
(0,0),(144,0),(0,80),(144,80)
365
```

```
136
(0,0),(144,0),(0,80),(144,80)
180
40
(0,0),(288,0),(0,160),(288,160)

i_detect/neg/neg3.jpg
7
686
429
(0,0),(18,0),(0,10),(18,10)
157
317
(0,0),(72,0),(0,40),(72,40)
437
295
(0,0),(72,0),(0,40),(72,40)
284
351
(0,0),(72,0),(0,40),(72,40)
489
341
(0,0),(144,0),(0,80),(144,80)
198
80
(0,0),(144,0),(0,80),(144,80)
184
280
(0,0),(288,0),(0,160),(288,160)

i_detect/neg/neg4.jpg
16
261
245
(0,0),(72,0),(0,40),(72,40)
596
62
(0,0),(72,0),(0,40),(72,40)
293
33
(0,0),(72,0),(0,40),(72,40)
401
186
(0,0),(72,0),(0,40),(72,40)
25
311
(0,0),(144,0),(0,80),(144,80)
171
185
(0,0),(144,0),(0,80),(144,80)
445
135
(0,0),(144,0),(0,80),(144,80)
351
```

```
275
(0,0),(144,0),(0,80),(144,80)
317
348
(0,0),(144,0),(0,80),(144,80)
450
253
(0,0),(144,0),(0,80),(144,80)
38
8
(0,0),(288,0),(0,160),(288,160)
259
6
(0,0),(288,0),(0,160),(288,160)
362
108
(0,0),(288,0),(0,160),(288,160)
83
172
(0,0),(288,0),(0,160),(288,160)
296
231
(0,0),(288,0),(0,160),(288,160)
53
269
(0,0),(288,0),(0,160),(288,160)

i_detect/neg/neg5.jpg
8
186
368
(0,0),(72,0),(0,40),(72,40)
441
280
(0,0),(72,0),(0,40),(72,40)
3
256
(0,0),(72,0),(0,40),(72,40)
500
354
(0,0),(72,0),(0,40),(72,40)
292
395
(0,0),(72,0),(0,40),(72,40)
220
247
(0,0),(144,0),(0,80),(144,80)
352
144
(0,0),(288,0),(0,160),(288,160)
90
176
(0,0),(288,0),(0,160),(288,160)
```

```
i_detect/pos/pos1.jpg
68
137
496
(0,0),(36,0),(0,20),(36,20)
130
364
(0,0),(36,0),(0,20),(36,20)
16
472
(0,0),(72,0),(0,40),(72,40)
80
543
(0,0),(72,0),(0,40),(72,40)
77
505
(0,0),(72,0),(0,40),(72,40)
229
385
(0,0),(72,0),(0,40),(72,40)
193
290
(0,0),(72,0),(0,40),(72,40)
174
478
(0,0),(72,0),(0,40),(72,40)
190
150
(0,0),(72,0),(0,40),(72,40)
138
532
(0,0),(72,0),(0,40),(72,40)
128
506
(0,0),(72,0),(0,40),(72,40)
261
145
(0,0),(72,0),(0,40),(72,40)
220
472
(0,0),(72,0),(0,40),(72,40)
329
488
(0,0),(72,0),(0,40),(72,40)
486
502
(0,0),(72,0),(0,40),(72,40)
789
383
(0,0),(72,0),(0,40),(72,40)
614
477
(0,0),(72,0),(0,40),(72,40)
698
```

```
535
(0,0),(72,0),(0,40),(72,40)
651
494
(0,0),(72,0),(0,40),(72,40)
716
504
(0,0),(72,0),(0,40),(72,40)
868
478
(0,0),(72,0),(0,40),(72,40)
825
550
(0,0),(72,0),(0,40),(72,40)
121
398
(0,0),(72,0),(0,40),(72,40)
414
550
(0,0),(72,0),(0,40),(72,40)
641
150
(0,0),(72,0),(0,40),(72,40)
741
547
(0,0),(72,0),(0,40),(72,40)
275
399
(0,0),(72,0),(0,40),(72,40)
751
493
(0,0),(72,0),(0,40),(72,40)
848
508
(0,0),(72,0),(0,40),(72,40)
206
320
(0,0),(72,0),(0,40),(72,40)
370
550
(0,0),(72,0),(0,40),(72,40)
578
549
(0,0),(72,0),(0,40),(72,40)
150
324
(0,0),(72,0),(0,40),(72,40)
493
407
(0,0),(72,0),(0,40),(72,40)
216
173
(0,0),(72,0),(0,40),(72,40)
572
```

```
173
(0,0),(72,0),(0,40),(72,40)
389
519
(0,0),(72,0),(0,40),(72,40)
49
347
(0,0),(72,0),(0,40),(72,40)
93
328
(0,0),(72,0),(0,40),(72,40)
85
71
(0,0),(72,0),(0,40),(72,40)
239
428
(0,0),(72,0),(0,40),(72,40)
552
425
(0,0),(72,0),(0,40),(72,40)
842
422
(0,0),(72,0),(0,40),(72,40)
174
352
(0,0),(72,0),(0,40),(72,40)
167
437
(0,0),(72,0),(0,40),(72,40)
799
522
(0,0),(72,0),(0,40),(72,40)
272
427
(0,0),(72,0),(0,40),(72,40)
369
356
(0,0),(72,0),(0,40),(72,40)
764
111
(0,0),(72,0),(0,40),(72,40)
3
531
(0,0),(72,0),(0,40),(72,40)
453
432
(0,0),(72,0),(0,40),(72,40)
27
273
(0,0),(72,0),(0,40),(72,40)
9
250
(0,0),(72,0),(0,40),(72,40)
240
```

```
237
(0,0),(144,0),(0,80),(144,80)
318
396
(0,0),(144,0),(0,80),(144,80)
361
164
(0,0),(144,0),(0,80),(144,80)
612
482
(0,0),(144,0),(0,80),(144,80)
700
315
(0,0),(144,0),(0,80),(144,80)
572
409
(0,0),(144,0),(0,80),(144,80)
461
499
(0,0),(144,0),(0,80),(144,80)
91
122
(0,0),(144,0),(0,80),(144,80)
704
433
(0,0),(144,0),(0,80),(144,80)
568
144
(0,0),(144,0),(0,80),(144,80)
227
286
(0,0),(144,0),(0,80),(144,80)
108
304
(0,0),(144,0),(0,80),(144,80)
232
236
(0,0),(288,0),(0,160),(288,160)
452
254
(0,0),(288,0),(0,160),(288,160)
244
384
(0,0),(288,0),(0,160),(288,160)

i_detect/pos/pos2.jpg
31
586
616
(0,0),(18,0),(0,10),(18,10)
345
692
(0,0),(18,0),(0,10),(18,10)
544
```

```
684
(0,0),(18,0),(0,10),(18,10)
37
663
(0,0),(72,0),(0,40),(72,40)
132
591
(0,0),(72,0),(0,40),(72,40)
300
550
(0,0),(72,0),(0,40),(72,40)
203
663
(0,0),(72,0),(0,40),(72,40)
378
659
(0,0),(72,0),(0,40),(72,40)
81
612
(0,0),(72,0),(0,40),(72,40)
344
598
(0,0),(72,0),(0,40),(72,40)
617
569
(0,0),(72,0),(0,40),(72,40)
727
669
(0,0),(72,0),(0,40),(72,40)
871
573
(0,0),(72,0),(0,40),(72,40)
222
578
(0,0),(72,0),(0,40),(72,40)
646
616
(0,0),(72,0),(0,40),(72,40)
781
594
(0,0),(72,0),(0,40),(72,40)
782
482
(0,0),(72,0),(0,40),(72,40)
307
560
(0,0),(144,0),(0,80),(144,80)
780
506
(0,0),(144,0),(0,80),(144,80)
97
353
(0,0),(144,0),(0,80),(144,80)
17
```

```
427
(0,0),(144,0),(0,80),(144,80)
181
450
(0,0),(144,0),(0,80),(144,80)
255
528
(0,0),(144,0),(0,80),(144,80)
700
592
(0,0),(144,0),(0,80),(144,80)
552
540
(0,0),(144,0),(0,80),(144,80)
6
622
(0,0),(144,0),(0,80),(144,80)
118
388
(0,0),(288,0),(0,160),(288,160)
590
392
(0,0),(288,0),(0,160),(288,160)
302
394
(0,0),(288,0),(0,160),(288,160)
208
532
(0,0),(288,0),(0,160),(288,160)
43
78
(0,0),(576,0),(0,320),(576,320)

i_detect/pos/pos3.jpg
2
640
385
(0,0),(144,0),(0,80),(144,80)
621
326
(0,0),(288,0),(0,160),(288,160)

i_detect/pos/pos4.jpg
9
237
407
(0,0),(72,0),(0,40),(72,40)
192
364
(0,0),(72,0),(0,40),(72,40)
713
186
(0,0),(72,0),(0,40),(72,40)
47
```

```
312
(0,0),(72,0),(0,40),(72,40)
127
318
(0,0),(144,0),(0,80),(144,80)
760
176
(0,0),(144,0),(0,80),(144,80)
32
362
(0,0),(144,0),(0,80),(144,80)
285
33
(0,0),(288,0),(0,160),(288,160)
279
58
(0,0),(576,0),(0,320),(576,320)

i_detect/pos/pos5.jpg
12
300
631
(0,0),(72,0),(0,40),(72,40)
483
653
(0,0),(72,0),(0,40),(72,40)
102
320
(0,0),(144,0),(0,80),(144,80)
156
577
(0,0),(144,0),(0,80),(144,80)
476
608
(0,0),(144,0),(0,80),(144,80)
430
43
(0,0),(288,0),(0,160),(288,160)
119
154
(0,0),(288,0),(0,160),(288,160)
307
464
(0,0),(288,0),(0,160),(288,160)
83
515
(0,0),(288,0),(0,160),(288,160)
600
523
(0,0),(288,0),(0,160),(288,160)
69
69
(0,0),(576,0),(0,320),(576,320)
292
```

```
20
(0,0),(576,0),(0,320),(576,320)

i_detect/pos/pos6.jpg
25
748
292
(0,0),(72,0),(0,40),(72,40)
536
241
(0,0),(72,0),(0,40),(72,40)
866
297
(0,0),(72,0),(0,40),(72,40)
709
202
(0,0),(72,0),(0,40),(72,40)
379
369
(0,0),(72,0),(0,40),(72,40)
792
317
(0,0),(72,0),(0,40),(72,40)
520
159
(0,0),(72,0),(0,40),(72,40)
825
140
(0,0),(72,0),(0,40),(72,40)
831
596
(0,0),(72,0),(0,40),(72,40)
173
704
(0,0),(72,0),(0,40),(72,40)
267
696
(0,0),(72,0),(0,40),(72,40)
689
126
(0,0),(72,0),(0,40),(72,40)
459
702
(0,0),(72,0),(0,40),(72,40)
386
697
(0,0),(72,0),(0,40),(72,40)
50
66
(0,0),(72,0),(0,40),(72,40)
568
82
(0,0),(72,0),(0,40),(72,40)
320
```

```
690
(0,0),(72,0),(0,40),(72,40)
62
24
(0,0),(144,0),(0,80),(144,80)
106
21
(0,0),(144,0),(0,80),(144,80)
457
584
(0,0),(144,0),(0,80),(144,80)
378
640
(0,0),(144,0),(0,80),(144,80)
704
650
(0,0),(144,0),(0,80),(144,80)
111
629
(0,0),(144,0),(0,80),(144,80)
251
665
(0,0),(144,0),(0,80),(144,80)
523
536
(0,0),(288,0),(0,160),(288,160)

i_detect/pos/pos7.jpg
20
849
645
(0,0),(36,0),(0,20),(36,20)
743
665
(0,0),(72,0),(0,40),(72,40)
301
623
(0,0),(72,0),(0,40),(72,40)
493
337
(0,0),(72,0),(0,40),(72,40)
775
325
(0,0),(72,0),(0,40),(72,40)
40
144
(0,0),(72,0),(0,40),(72,40)
393
345
(0,0),(72,0),(0,40),(72,40)
472
388
(0,0),(144,0),(0,80),(144,80)
725
```

```
267
(0,0),(144,0),(0,80),(144,80)
589
259
(0,0),(144,0),(0,80),(144,80)
699
602
(0,0),(144,0),(0,80),(144,80)
731
194
(0,0),(144,0),(0,80),(144,80)
531
515
(0,0),(144,0),(0,80),(144,80)
692
213
(0,0),(144,0),(0,80),(144,80)
38
610
(0,0),(144,0),(0,80),(144,80)
632
620
(0,0),(144,0),(0,80),(144,80)
455
101
(0,0),(288,0),(0,160),(288,160)
288
189
(0,0),(288,0),(0,160),(288,160)
479
305
(0,0),(288,0),(0,160),(288,160)
250
224
(0,0),(576,0),(0,320),(576,320)

i_detect/pos/pos8.jpg
16
417
402
(0,0),(72,0),(0,40),(72,40)
848
141
(0,0),(72,0),(0,40),(72,40)
804
149
(0,0),(72,0),(0,40),(72,40)
712
362
(0,0),(72,0),(0,40),(72,40)
232
267
(0,0),(72,0),(0,40),(72,40)
751
```

```
136
(0,0),(144,0),(0,80),(144,80)
263
185
(0,0),(144,0),(0,80),(144,80)
393
178
(0,0),(144,0),(0,80),(144,80)
768
290
(0,0),(144,0),(0,80),(144,80)
74
193
(0,0),(144,0),(0,80),(144,80)
365
382
(0,0),(144,0),(0,80),(144,80)
144
201
(0,0),(144,0),(0,80),(144,80)
384
154
(0,0),(288,0),(0,160),(288,160)
60
148
(0,0),(288,0),(0,160),(288,160)
46
245
(0,0),(288,0),(0,160),(288,160)
224
117
(0,0),(576,0),(0,320),(576,320)

i_detect/pos/pos9.jpg
56
10
490
(0,0),(18,0),(0,10),(18,10)
74
375
(0,0),(18,0),(0,10),(18,10)
713
480
(0,0),(18,0),(0,10),(18,10)
6
466
(0,0),(36,0),(0,20),(36,20)
814
477
(0,0),(36,0),(0,20),(36,20)
558
485
(0,0),(36,0),(0,20),(36,20)
913
```

```
507
(0,0),(36,0),(0,20),(36,20)
230
183
(0,0),(72,0),(0,40),(72,40)
268
203
(0,0),(72,0),(0,40),(72,40)
684
271
(0,0),(72,0),(0,40),(72,40)
68
130
(0,0),(72,0),(0,40),(72,40)
267
216
(0,0),(72,0),(0,40),(72,40)
24
236
(0,0),(72,0),(0,40),(72,40)
99
73
(0,0),(72,0),(0,40),(72,40)
115
137
(0,0),(72,0),(0,40),(72,40)
810
269
(0,0),(72,0),(0,40),(72,40)
20
80
(0,0),(72,0),(0,40),(72,40)
492
82
(0,0),(72,0),(0,40),(72,40)
526
328
(0,0),(72,0),(0,40),(72,40)
573
328
(0,0),(72,0),(0,40),(72,40)
152
368
(0,0),(72,0),(0,40),(72,40)
253
415
(0,0),(72,0),(0,40),(72,40)
304
413
(0,0),(72,0),(0,40),(72,40)
410
374
(0,0),(72,0),(0,40),(72,40)
147
```

```
345
(0,0),(72,0),(0,40),(72,40)
60
423
(0,0),(72,0),(0,40),(72,40)
4
341
(0,0),(72,0),(0,40),(72,40)
198
346
(0,0),(72,0),(0,40),(72,40)
82
458
(0,0),(72,0),(0,40),(72,40)
63
349
(0,0),(72,0),(0,40),(72,40)
5
478
(0,0),(72,0),(0,40),(72,40)
46
469
(0,0),(72,0),(0,40),(72,40)
123
450
(0,0),(72,0),(0,40),(72,40)
154
487
(0,0),(72,0),(0,40),(72,40)
169
438
(0,0),(72,0),(0,40),(72,40)
533
455
(0,0),(72,0),(0,40),(72,40)
610
470
(0,0),(72,0),(0,40),(72,40)
92
188
(0,0),(144,0),(0,80),(144,80)
20
90
(0,0),(144,0),(0,80),(144,80)
165
310
(0,0),(144,0),(0,80),(144,80)
198
154
(0,0),(144,0),(0,80),(144,80)
430
13
(0,0),(144,0),(0,80),(144,80)
248
```

```
158
(0,0),(144,0),(0,80),(144,80)
448
229
(0,0),(144,0),(0,80),(144,80)
691
316
(0,0),(144,0),(0,80),(144,80)
479
172
(0,0),(144,0),(0,80),(144,80)
738
240
(0,0),(144,0),(0,80),(144,80)
786
232
(0,0),(144,0),(0,80),(144,80)
14
440
(0,0),(144,0),(0,80),(144,80)
98
119
(0,0),(144,0),(0,80),(144,80)
754
287
(0,0),(144,0),(0,80),(144,80)
196
33
(0,0),(288,0),(0,160),(288,160)
613
65
(0,0),(288,0),(0,160),(288,160)
432
144
(0,0),(288,0),(0,160),(288,160)
479
244
(0,0),(288,0),(0,160),(288,160)
604
252
(0,0),(288,0),(0,160),(288,160)

i_detect/pos/pos10.jpg
25
870
53
(0,0),(36,0),(0,20),(36,20)
881
530
(0,0),(36,0),(0,20),(36,20)
693
321
(0,0),(72,0),(0,40),(72,40)
841
```

```
18
(0,0),(72,0),(0,40),(72,40)
848
50
(0,0),(72,0),(0,40),(72,40)
552
147
(0,0),(72,0),(0,40),(72,40)
814
242
(0,0),(72,0),(0,40),(72,40)
227
110
(0,0),(72,0),(0,40),(72,40)
642
278
(0,0),(72,0),(0,40),(72,40)
476
363
(0,0),(72,0),(0,40),(72,40)
632
399
(0,0),(72,0),(0,40),(72,40)
803
507
(0,0),(72,0),(0,40),(72,40)
520
449
(0,0),(72,0),(0,40),(72,40)
823
435
(0,0),(72,0),(0,40),(72,40)
632
328
(0,0),(144,0),(0,80),(144,80)
337
249
(0,0),(144,0),(0,80),(144,80)
489
153
(0,0),(144,0),(0,80),(144,80)
541
90
(0,0),(144,0),(0,80),(144,80)
716
480
(0,0),(144,0),(0,80),(144,80)
314
424
(0,0),(144,0),(0,80),(144,80)
729
15
(0,0),(144,0),(0,80),(144,80)
579
```

```
50
(0,0),(144,0),(0,80),(144,80)
523
11
(0,0),(288,0),(0,160),(288,160)
384
36
(0,0),(288,0),(0,160),(288,160)
202
221
(0,0),(288,0),(0,160),(288,160)

i_detect/pos/pos11.jpg
45
226
541
(0,0),(18,0),(0,10),(18,10)
543
595
(0,0),(36,0),(0,20),(36,20)
149
561
(0,0),(72,0),(0,40),(72,40)
160
398
(0,0),(72,0),(0,40),(72,40)
183
313
(0,0),(72,0),(0,40),(72,40)
468
337
(0,0),(72,0),(0,40),(72,40)
532
413
(0,0),(72,0),(0,40),(72,40)
810
556
(0,0),(72,0),(0,40),(72,40)
841
328
(0,0),(72,0),(0,40),(72,40)
806
399
(0,0),(72,0),(0,40),(72,40)
654
578
(0,0),(72,0),(0,40),(72,40)
697
577
(0,0),(72,0),(0,40),(72,40)
68
341
(0,0),(72,0),(0,40),(72,40)
837
```

```
571
(0,0),(72,0),(0,40),(72,40)
160
588
(0,0),(72,0),(0,40),(72,40)
200
542
(0,0),(72,0),(0,40),(72,40)
225
348
(0,0),(72,0),(0,40),(72,40)
869
583
(0,0),(72,0),(0,40),(72,40)
19
244
(0,0),(72,0),(0,40),(72,40)
49
242
(0,0),(72,0),(0,40),(72,40)
840
547
(0,0),(72,0),(0,40),(72,40)
872
549
(0,0),(72,0),(0,40),(72,40)
104
143
(0,0),(72,0),(0,40),(72,40)
439
594
(0,0),(72,0),(0,40),(72,40)
765
593
(0,0),(72,0),(0,40),(72,40)
238
386
(0,0),(72,0),(0,40),(72,40)
26
318
(0,0),(144,0),(0,80),(144,80)
99
236
(0,0),(144,0),(0,80),(144,80)
147
328
(0,0),(144,0),(0,80),(144,80)
258
297
(0,0),(144,0),(0,80),(144,80)
683
343
(0,0),(144,0),(0,80),(144,80)
158
```

```
518
(0,0),(144,0),(0,80),(144,80)
249
430
(0,0),(144,0),(0,80),(144,80)
111
119
(0,0),(144,0),(0,80),(144,80)
783
199
(0,0),(144,0),(0,80),(144,80)
131
60
(0,0),(144,0),(0,80),(144,80)
347
289
(0,0),(144,0),(0,80),(144,80)
490
53
(0,0),(144,0),(0,80),(144,80)
520
295
(0,0),(144,0),(0,80),(144,80)
798
380
(0,0),(144,0),(0,80),(144,80)
35
29
(0,0),(288,0),(0,160),(288,160)
502
161
(0,0),(288,0),(0,160),(288,160)
363
135
(0,0),(288,0),(0,160),(288,160)
356
336
(0,0),(288,0),(0,160),(288,160)
192
288
(0,0),(576,0),(0,320),(576,320)

i_detect/pos/pos12.jpg
17
852
151
(0,0),(18,0),(0,10),(18,10)
923
22
(0,0),(18,0),(0,10),(18,10)
851
440
(0,0),(18,0),(0,10),(18,10)
804
```

```
120
(0,0),(18,0),(0,10),(18,10)
174
504
(0,0),(18,0),(0,10),(18,10)
791
460
(0,0),(72,0),(0,40),(72,40)
735
472
(0,0),(72,0),(0,40),(72,40)
695
89
(0,0),(72,0),(0,40),(72,40)
558
300
(0,0),(72,0),(0,40),(72,40)
634
490
(0,0),(72,0),(0,40),(72,40)
161
70
(0,0),(144,0),(0,80),(144,80)
257
174
(0,0),(144,0),(0,80),(144,80)
445
310
(0,0),(144,0),(0,80),(144,80)
643
154
(0,0),(144,0),(0,80),(144,80)
754
444
(0,0),(144,0),(0,80),(144,80)
246
31
(0,0),(288,0),(0,160),(288,160)
94
151
(0,0),(288,0),(0,160),(288,160)
```

Cloud identification

Omitted. The results from this experimented yielded no data (ie no clouds were detected in any of the images tested).