

Institiúid Teicneolaíochta Cheatharlach



INSTITUTE *of*
TECHNOLOGY

CARLOW

At the heart of South Leinster

Project Report

What's it like Outside?

By

Dermot Rossiter

Supervised by

Paul Barry

Table of Contents

INTRODUCTION	2
DOCUMENT OVERVIEW	3
PROJECT DESCRIPTION	3
EXTRACTING THE SKY REGION - OPTION 1	3
<i>Confirmation of Techniques Used.....</i>	7
<i>Disadvantages of Option 1</i>	10
EXTRACTING THE SKY REGION - OPTION 2	11
<i>Confirmation of Techniques Used.....</i>	11
<i>Disadvantages of Option 2</i>	14
ANALYSING THE SKY	14
<i>RGB Colour Segmentation</i>	14
<i>HSV (Hue, Saturation, Value) Colour Segmentation</i>	16
<i>Identifying Blue Sky</i>	19
<i>Disadvantages of Using Colour Segmentation</i>	21
<i>Identifying Dark Clouds</i>	22
CONFORMANCE TO SPECIFICATION	23
LEARNING DESCRIPTION.....	23
TECHNICAL	23
PERSONAL	24
PROJECT REVIEW.....	24
ACKNOWLEDGEMENTS.....	25
REFERENCES	26

Introduction

When viewing a photograph of outside, the majority of people will be able to determine the weather condition, from even just a small amount of information. They can tell if it is sunny. Overcast. Even if it is raining in some cases. But can a computer?

The purpose of this project is to research if computer vision techniques are capable in determining if the weather is either overcast, cloudy or clear skies, by analysing an image taken of the 'outdoors'.

The steps in this report for determining the weather condition from an image is based on the following assumptions:

1. Images where the sky region is present in the upper part of the image will only be used for testing. Images with any interference with the view of the sky will not be taken into account.



2. Also, images will need to be taken during times of daylight. Therefore, no images of a morning sunrise or a night time sunset will be taken into account for this project.



Research of the computer vision techniques will be tested using OpenCV using the Python programming language on random images obtained from Google Images.

Document Overview

The following information will be explained in this document:

- Project Description: Detailing the steps taken to test the computer vision techniques to try and determine the weather condition by analysing an image
- Learning description: Discuss two different aspects of learning outcomes after finishing the project. Technical and Personal.
- Project Overview:
- Acknowledgements: This section mentions all the people who helped with the project

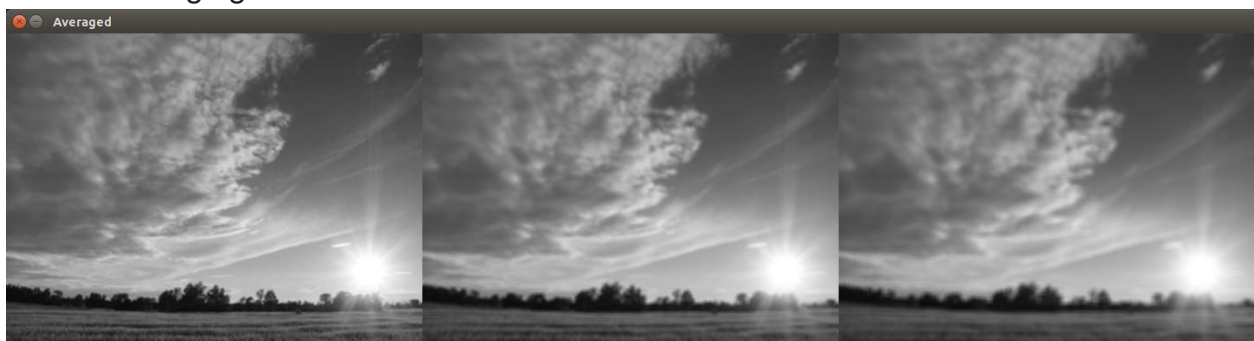
Project Description

To start off, the goal was to first try to use computer vision techniques to extract the sky region from an image. To do this, a couple of different techniques were used, such as edge detection, thresholding, masking, blurring, etc. The best solution found was to use a combination of 2 different options.

Extracting the Sky Region - Option 1

Blurring is the first technique to be researched. Image blurring is achieved by convolving the image with a low-pass filter kernel [1]. It is commonly used for removing noise from an image. In some cases, edges are blurred when the filter is applied. There are also some ways to blur an image while retaining the edges. In this report we tested 3 different blurring techniques provided by OpenCV.

1. Averaging



Averaged blurring with a 3x3 filter on the left, 5x5 in the middle and 7x7 on the right.

Implementing average blur to the image is a simple process. Using OpenCV, it can be accomplished in 1 line of code, `cv2.blur(image, (3, 3))`, where (3, 3) refers to the filter, of how much the image should be blurred.

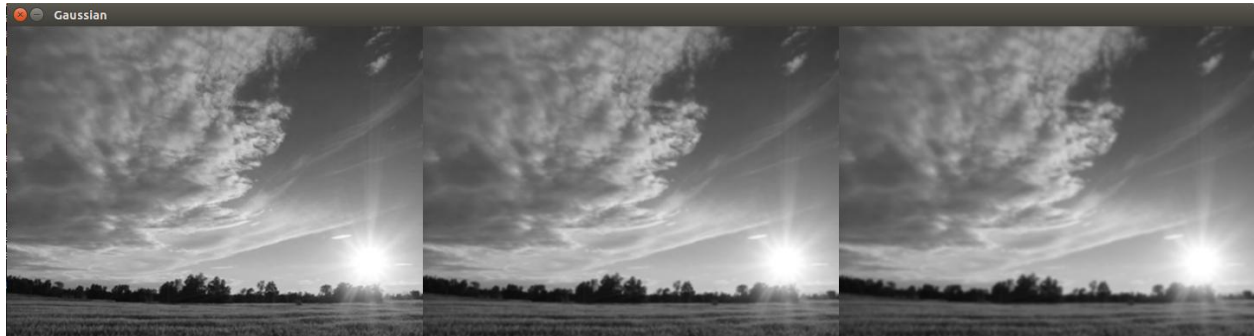
$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Figure.1: 3x3 filter

Figure 1 illustrates how the filter is applied to the image. It finds the average of all the pixels under the kernel area and replaces the central value with this average.

The results from using averaging blur with OpenCV did not produce satisfactory results to work with canny edge detection in detecting the gradients along the horizon.

2. Gaussian Blur



Gaussian blurring with a 3x3 filter on the left, 5x5 in the middle and 7x7 on the right.

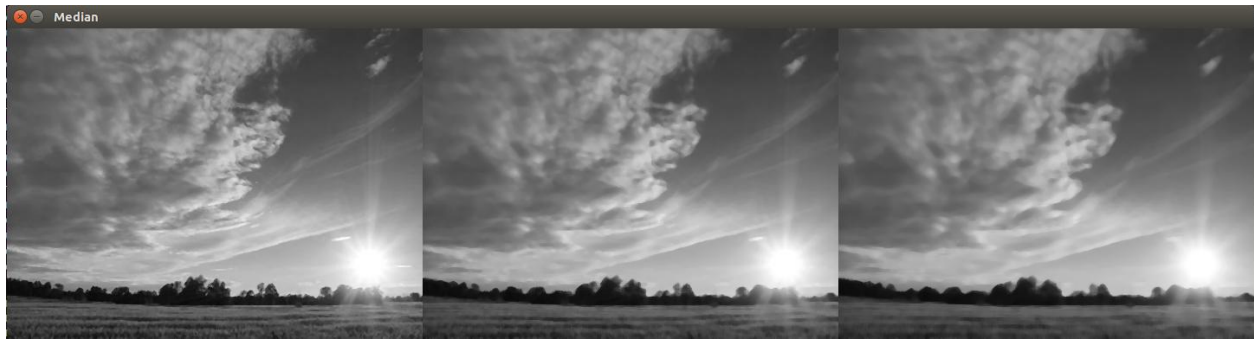
Gaussian blurring is the most commonly used blurring method [2]. It is used to reduce noise and reduce detail. It uses a convolution kernel. It is similar to average blurring except instead of using a simple mean, a weighted mean is used instead, where the central value has a greater value, and the weights decrease with distance from the neighborhood center value. Figure 2 illustrates how the kernel is applied to the image.

$$\frac{1}{16} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Figure.2: Weighted Kernel. [3]

This results in a more natural blur than average blur.

3. Median Blur



Applying the median blur method with an increase in sizes of 3 on the left, 5 in the middle and 7 on the right.

Median blur technique is very effective in removing noise from an image. The method works by finding the median of all the pixels in the kernel and then the central pixel is replaced with the median value. It is commonly used with edge detection algorithms because it reduces noise while retaining the edges.

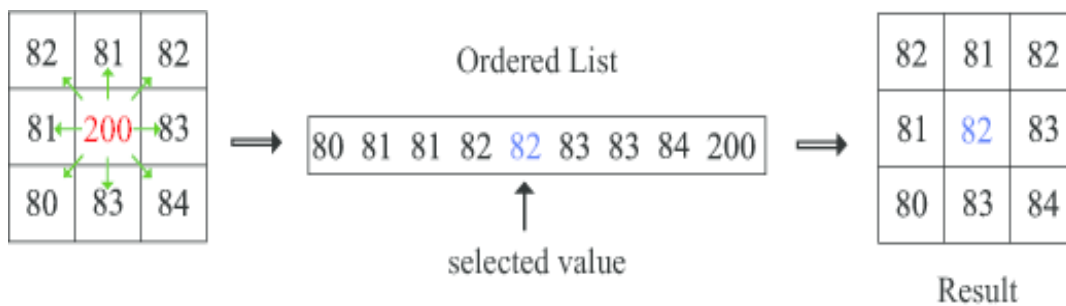
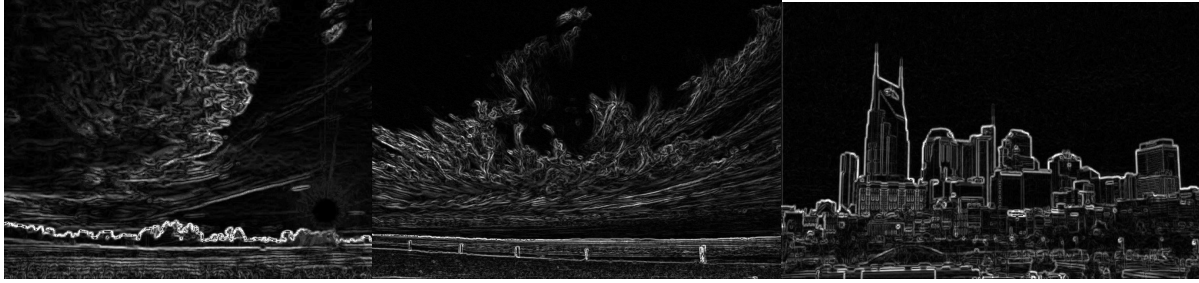


Figure.3: Median filtering [4]

As seen from Figure 3, where each element in the matrix represents the image, central element and its neighbouring elements are sorted in an ordered list. Then the central value in the ordered list is selected and it replaces the original central element with the median element from the list.

The next technique to be tested is edge detection. It is commonly used in the area of computer vision for tasks such as feature detection of feature extraction. It helps us reduce the amount of data (pixels) to process and maintains the structural aspect of the image [3]. The following are 2 commonly used edge detection algorithms used in image processing.

1. Sobel edge detection



Sobel edge detection, also called Sobel operator, finds edges by processing an image both vertically and horizontally for any difference in pixel intensities.

The operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives [5].

The sobel masks (3x3):

For x-Direction:

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

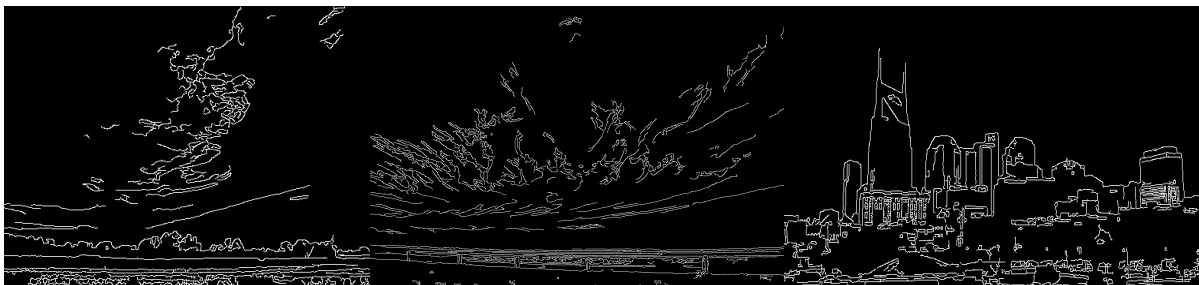
For Y-direction:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Figure.4: Sobel Kernels [6]

The results from the Sobel edge detection shown above uses a combination of the Sobel operator applied along the x-axis and the y-axis.

2. Canny edge detection



Canny edge detection uses the following multi-step process:

- Blurring of the image to remove noise.
- Computing gradients in the x and y direction
- Suppressing edges
- Hysteresis thresholding to determine edges

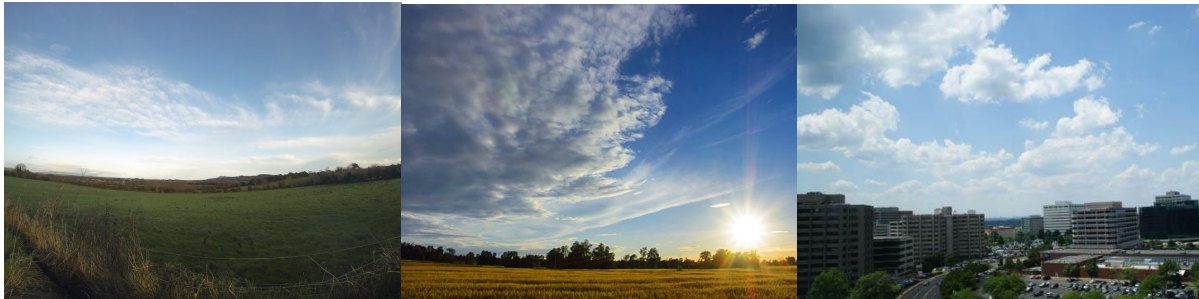
An example of applying canny edge detection using OpenCV looks like the following: `canny = cv2.Canny(image, 30, 150)`, where image refers to the blurred, grayscale image, 30 and 150 refer to threshold values. Any value below 30 is not an edge, above 150 is considered to be an edge and any values in between needs to be classified to determine if it is an edge or not.

Confirmation of Techniques Used

Through testing the above blurring and edge detection techniques concurrently with each other over various different images, it was found that the best results for detection of the horizon was with the Gaussian blur method alongside canny edge detection.

Step 1.

First approach involved reading in an image.



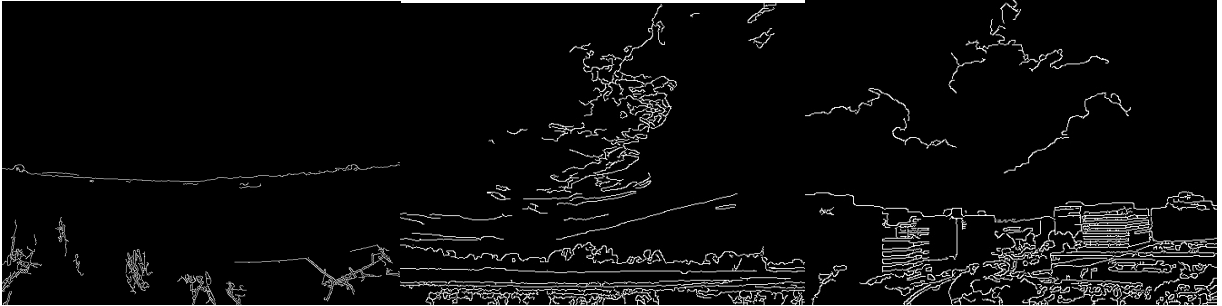
Step 2.

This image then gets converted to grayscale and is slightly blurred.

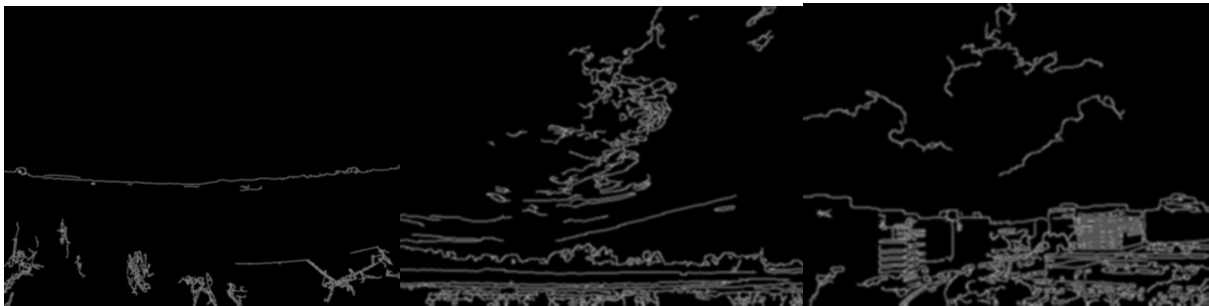


Step 3.

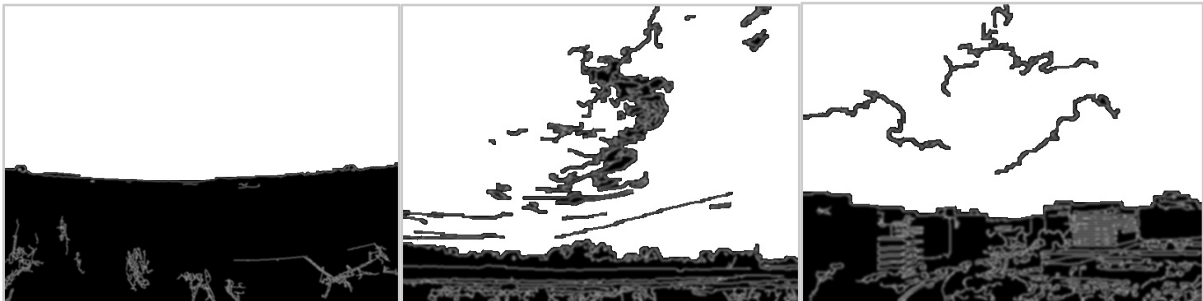
Canny edge detection is applied

**Step 4.**

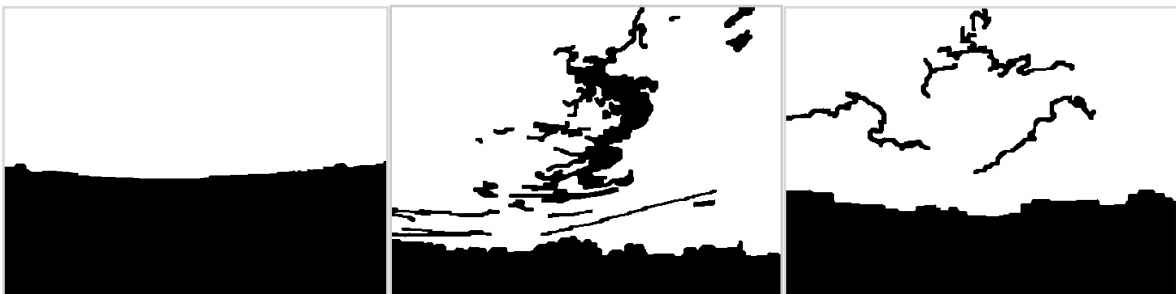
Blur the image again to merge any gaps that might be missing along the horizon.

**Step 5.**

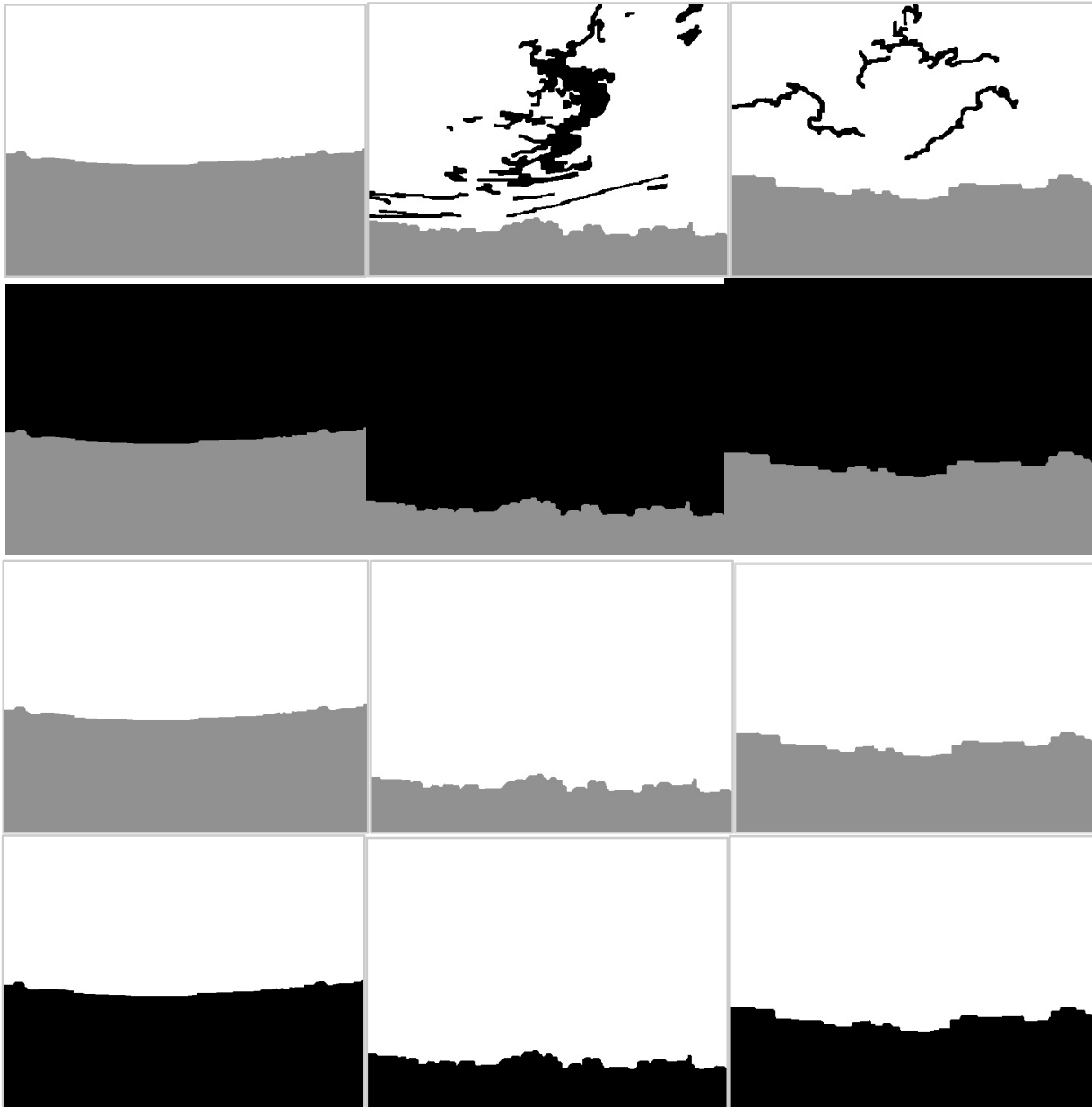
Floodfill the top left corner of the image white.

**Step 6.**

Applying thresholding to the image to remove the grey colour in the image, using simple thresholding to convert it to a binary image.

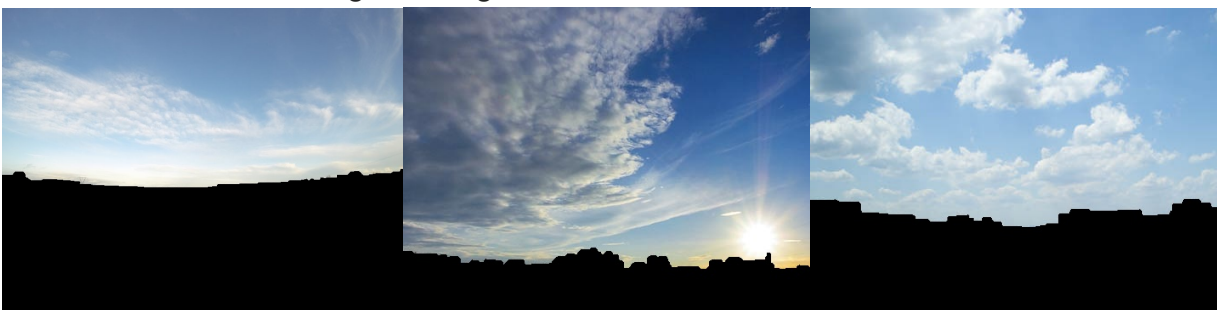
**Step 7.**

To remove the unwanted lines in the sky region, a series of the floodfill algorithm is applied both at the top-left and bottom-left corners of the image.



Step 8.

Now that the sky region is identified by the white region in the image, we can now use this to mask it with the original image.



Even though the steps used to extract the sky region from an image work for certain images, it is far from perfect. After running these steps on over 50 random images of varying different sky conditions, it was not successful in all of them. The majority of failures usually tend to be on images where the sky region is very cloudy or with dark overcast clouds. Therefore, a new approach needed to be implemented along side of the previous steps shown above.

Disadvantages of Option 1

There are a few disadvantages to using edge detection for extracting the sky region from an image. The first problem is in images where the sky is extremely cloudy or overcast, the gradient along the horizon can be difficult to establish effectively.



As we can see from the image above, the difficulty in finding the horizon is clear and therefore will not be capable in extracting the sky region from the image, because after the floodfill algorithm is applied, the entire image results in a complete black image.

Another disadvantage is when the horizon is a good distance away from the source of where the image was taken.



Take this image of a city skyline for example. Because of the long distance between the horizon and the source of where the image was taken, the horizon is not identified properly, as seen in the third image.

Extracting the Sky Region - Option 2

Therefore, another option in segmenting the sky region from an image was needed to handle whatever images were unsuccessful in option 1. This next technique to extract the sky region in unsuccessful images from the first series of steps, is to apply a thresholding method.

Thresholding works by converting a grayscale image to a binary image. A constant value is then needed to set, where any pixel intensities below this value are set to 0, and all the pixels' intensities higher than the value gets set to 255.

A few thresholding techniques were test such as the following.

1. Simple thresholding
2. Adaptive thresholding
3. Otsu's method

All 3 techniques are similar in that they convert a grayscale to a binary image. The major difference is finding a threshold value. Simple Thresholding requires the user to input the threshold value manually. Adaptive thresholding works by calculating different thresholding values based on small sections of an image. This way the threshold values changes depending on the region of the image. It's a better approach than simple thresholding and provides better results on images where there might be varying illumination. And lastly, Otsu's method. This approach produced the best results overall. It calculates the threshold automatically by assuming that there are 2 peaks in the histogram of the image, and it finds the optimal thresholding value based on these 2 peak values.

Confirmation of Techniques Used

There are a few different thresholding options to choose from. The best option found for this project was to use Otsu's method.

Step 1.

Here, we use images that were unsuccessful in extracting the sky region from the steps undertaken in option 1.



Step 2.

Again, similar to step 2 in option 1, we apply Gaussian blur to the image.



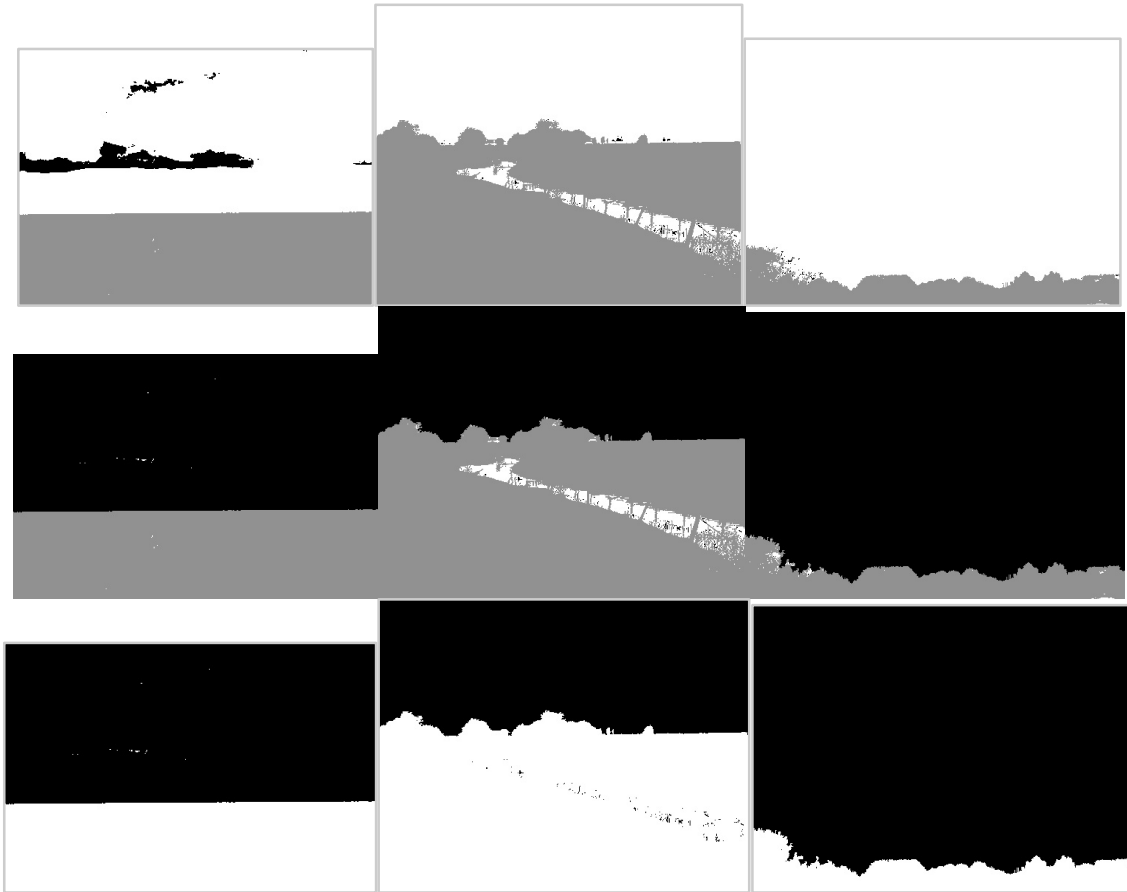
Step 3.

Applying Otsu's method to threshold the grayscale image



Step 4.

Now after thresholding the the image, and it is converted to a binary image, we need to apply the floodfill algorithm once more, to remove any unwanted features, such as the the clouds and the stream from the images above.



Step 5.

We need to swap the black and white colour from the image to prepare the image to be masked. To do this we apply binary inverse on the image.



Step 6.

Lastly, we mask the image from the output of step 4 against the original image to extract the sky region.



Disadvantages of Option 2

The main disadvantage of using a thresholding method to extract the sky region from an image is when the sky has a significant difference in pixel intensity such as when very dark clouds and brighter skies are present together, as illustrated in the images below.



Analysing the Sky

Now that the the sky region is extracted from the image, we will use colour segmentation to find and count the percentage of the colour blue range present in the sky region which will in turn allow us to make assumptions on if the weather condition in an image is either clear blue, cloudy, overcast, etc.

The following 2 approaches to colour segmentation were tested.

RGB Colour Segmentation

The RGB (Red, Green, Blue) model produces a colour which is defined by the value associated with each colour. RGB values range from 0 to 255. So, for example, the RGB value (255,0,0) produces the colour red, the RGB value (0,255,0) produces the colour green, and so on.

OpenCV handles RGB values differently, instead using it in reverse order, in the form of a BGR format.

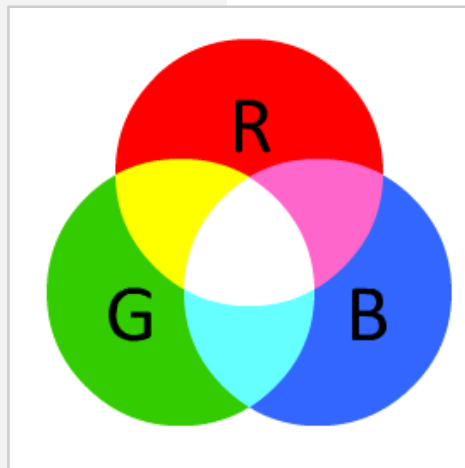
There are many RGB colour charts available to find a certain range of colours, but for this project it provided a difficult task to find the best value for a light blue and the value for a dark blue which are used for detecting the range of the blue sky in images.

An example of finding the BGR values for the blue sky is as follows:

```
BLUE_MIN = np.array([99,46,0],np.uint8)
BLUE_MAX = np.array([255,200,142],np.uint8)
```

From the lines of code shown above, the [99,46,0] corresponds to a dark blue colour and [255,200,142] corresponds to a light blue colour. These values are used to locate all the pixels within the ranges of these two values, which results in the blue sky being identified.

This approach is not very user friendly. The BGR values were extremely difficult to obtain to sufficiently work over a series of images. So another approach was needed.



RGB colour model. [7]

Blue matrix

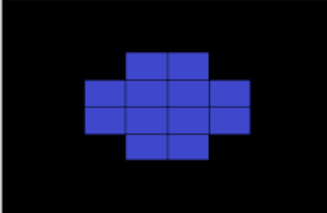
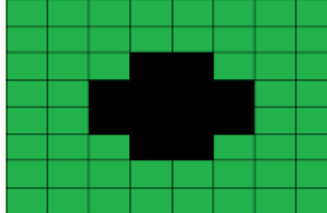

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	255	255	0	0	0
0	0	255	255	255	255	0	0
0	0	255	255	255	255	0	0
0	0	0	255	255	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Green matrix

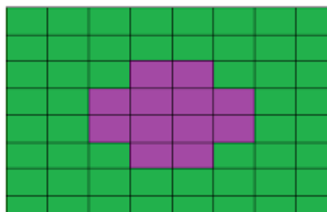
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	0	0	255	255	255
255	255	0	0	0	0	255	255
255	255	0	0	0	0	255	255
255	255	255	0	0	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255

Red matrix

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	255	255	0	0	0
0	0	255	255	255	255	0	0
0	0	255	255	255	255	0	0
0	0	0	255	255	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Resultant Image

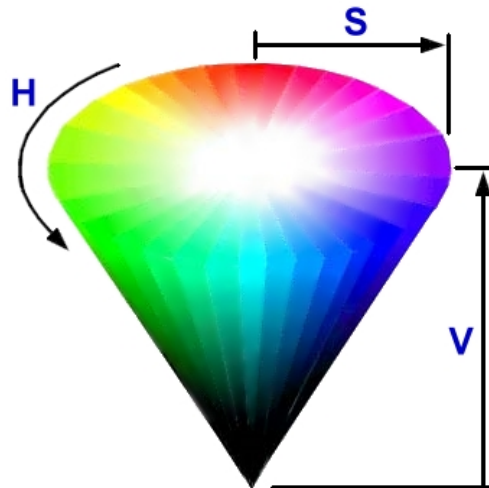


How BGR image is formed

[8]

HSV (Hue, Saturation, Value) Colour Segmentation

“Usually, one can think that BGR color space is more suitable for color based segmentation. But HSV color space is the most suitable color space for color based image segmentation”[8].

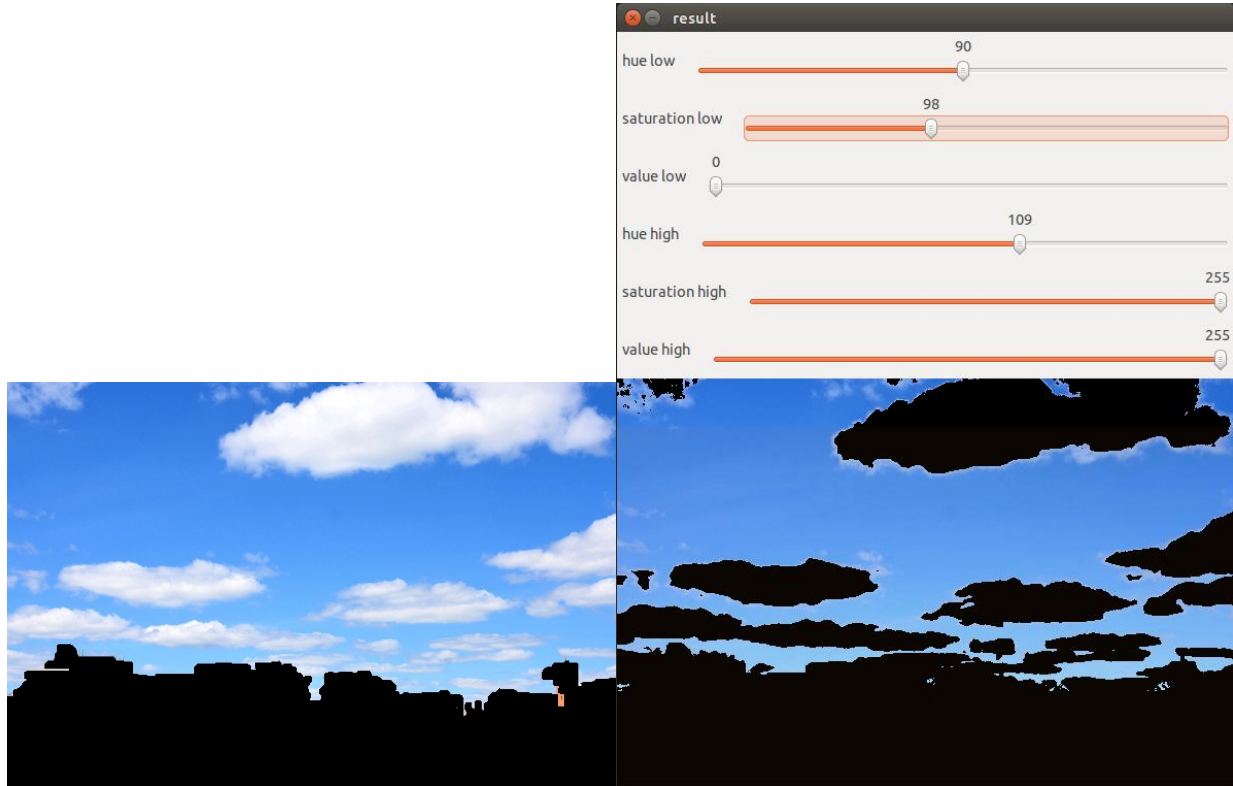


HSV color model. [7]

In HSV, hue represents colour, so for example, to find the pure colour of blue, the only value that needed to change was the hue value. The saturation value indicates the range of grey in the colour space [9]. And lastly, Value is the brightness of the colour which ranges from 0 to 100% which is used to find the different shades of blue, from light blue to dark blue.

Again, OpenCV handles HSV values differently than standard HSV values. For example, some applications use Hue = 0-360, Saturation = 0-100 and Value = 0-100 for the HSV range. But OpenCV uses Hue = 0 - 180, Saturation = 0 - 255, Value = 0 - 255.

Therefore, the quickest and best solution to find the most accurate HSV values was to create a simple program, that allows each value to be tweaked through a trackbar.



Extracted Sky Region

Program to find hsv values for the colour blue

The above image illustrates how the HSV values are achieved. As we can see the range for the colour for the blue sky falls between the Hue range of 90 - 110.

Colours	Hue Ranges
Orange	0 - 22
Yellow	22 - 38
Green	38 - 75
Blue	75 - 130
Violet	130 -160
Red	160 - 179

Identifying Blue Sky

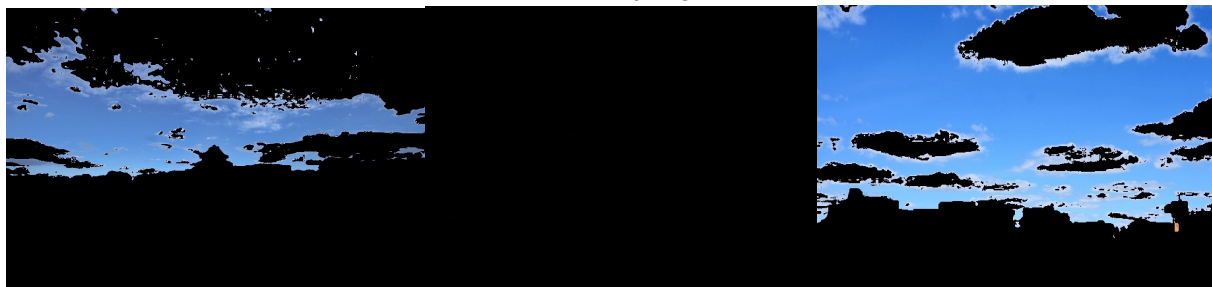
Extracting the clouds from the images using HSV colour values.



Original Images



Extracted sky region



Identifying blue sky

Results from the first column of images

```
25
The total number of pixels is: 539100.0
The number of blue pixels is: 135074
The number of black pixels is: 221038
The number of pixels of the sky region is : 318062.0
The percentage of blue in the sky region is : 42.46782073935271
Cloudy sky with some clear blue sky.
```

Results from the second column of images

```
21
The total number of pixels is: 230400.0
The number of blue pixels is: 3829
The number of black pixels is: 101170
The number of pixels of the sky region is : 129230.0
The percentage of blue in the sky region is : 2.962934303180376
The sky is overcast.
```

Results from the third column of images

```
30
The total number of pixels is: 238200.0
The number of blue pixels is: 138407
The number of black pixels is: 59287
The number of pixels of the sky region is : 178913.0
The percentage of blue in the sky region is : 77.3599458954911
Clear sky with some clouds.
```

Step 1.

Find the total number of pixels in the image.

Step 2.

Count the number of black pixels (non-sky region) and subtract the value from the total number of pixels.

Step 3.

Now that we have the number of pixels in the sky region, we now count the number of blue pixels in the sky region.

Step 4.

From this, we can establish the percentage of the colour blue in the sky region.

Step 5.

Below is a table showing how the results are displayed by using the percentage of the colour blue present in an image.

BLUE PERCENTAGE RANGE	OUTPUT DISPLAYED
Greater than 90	Clear Blue Sky
Greater than 70 and less than or equal to 90	Clear skies with some clouds
Greater than 50 and less than or equal to 70	Clear and Cloudy sky
Greater than 30 and less than or equal to 50	Cloudy sky with some clear blue sky
Greater than 10 and less than or equal to 30	Cloudy Sky
Less than or equal to 10	Overcast

As shown from the table, it is assumed that if the percentage of blue pixels present in the image is below or equal to 10 that the weather condition in the image is overcast. Because some cases of where the sky is overcast, some clouds are much darker than others, so from this an assumption can be made that if darker the clouds, the more likely the case of there being stormy skies present in the image.

Disadvantages of Using Colour Segmentation

There were some difficulties in trying to extract the colour blue from the sky region. The first issue was the glare of the sun in the image. Because of it's bright colour, the glare is mistaken as a cloud. The range value was tweaked as much possible to retain as much blue sky from such images without the same values affecting lighter cloud pixel values in other images. The problem is shown in the images below.



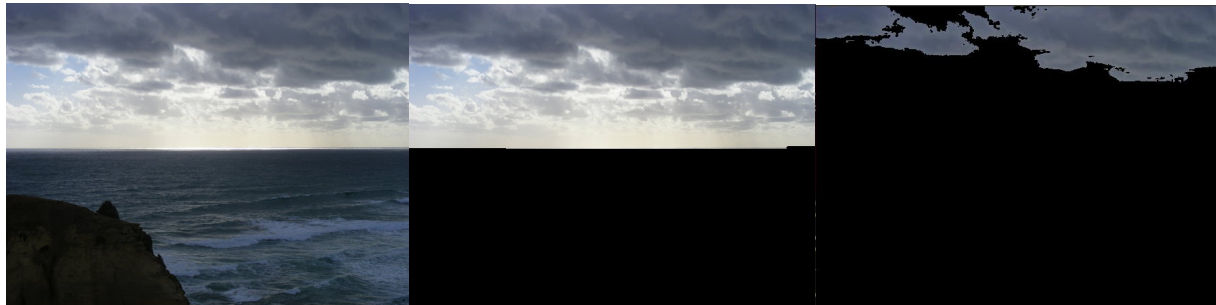
The next issue was with the presence of some dark clouds. Some pixel colours within specific parts of the clouds, fall between the ranges used for finding the blue sky. This problem cannot be solved using the techniques used in this project. If the bluish colour in the clouds are removed, then it affects the darker blue pixels in other images.



Identifying Dark Clouds

To find stormier skies, images assumed to be overcast will be analysed more to try to identify significantly darker clouds from the sky region. The HSV trackbar program previously used in identifying the blue sky will be used again to find the best HSV values for identifying darker clouds.

The images below illustrate the darker clouds being identified and the resulted output.



```
The percentage of grey in the sky region is : 32.5780470967247
Some stormy skies with potential for rain.
```

The following table shows how the results are displayed. The output is not an important factor in this project and the results are just for testing purposes only.

DARK GREY PERCENTAGE RANGE	OUTPUT DISPLAYED
Greater than 70	Severe stormy skies
Greater than 50 and less than or equal to 70	Very Stormy skies
Greater than 30 and less than or equal to 50	Some stormy skies
Greater than 9 and less than or equal to 30	Scattered rain clouds
Less than or equal to 9	Overcast

Based on these results, it might be safe to assume that the darker the clouds, the greater the threat of the possibility of rain being present in the image.

Conformance to Specification

The original specification states the following:

“Web camera captures an image of “outside” hourly, then processes the image to determine what the weather is like, for instance: cloudy, overcast, raining, sunny, etc. Description is then converted to text, then spoken - designed to be used by people with vision impairment.”

Given this specification, I am very satisfied that the finished product conforms to the original specification. Though far from perfect, I believe that the techniques used produced accurate results the majority of the time. If the sky region can be extracted from the image, the colour segmentation method provide enough information to assume the weather conditions mentioned in the original specification.

Learning Description

Technical

This was my first time being introduced to the area of computer vision. There is a vast range of techniques to learn and through this project I gained valuable knowledge in understanding and learning how implement basics of computer vision techniques. The experience of now being presented with any image, and being tasked with trying to extract and analyse any information, seems like a fun and interesting task. With a comfortable knowledge of basics of computer vision techniques, it now gives me confidence to expand my internet into the field of computer vision even more.

The choice of using OpenCV as the choice for implementing these vision techniques was a no-brainer. There is a huge community, with a huge source of material available for all aspects of image analysis.

The start of the project was a very stressful period. I had absolutely no idea how to approach the provided specification. As I mentioned, I had no knowledge of the techniques that needed were required to complete such a task.

Finally, it was also a nice project to build upon my knowledge of using the Python programming language. This is my first year studying the topic, and I find far more enjoyable, and easier to code with than any other programming language used so far.

Personal

Though this project, I feel that my researching skills have greatly improved. As mentioned, I found the beginning of this project pretty stressful. While I was watching fellow students get up and running with their projects, I was still researching different computer vision techniques without any knowledge of how these techniques even worked. But from this experience, looking back, I found this period hugely beneficial. And by researching various computer vision topics, that somewhat relate to specification of my project, was extremely valuable.

Also, another area where I need a huge improvement in, is my presentation skills. This project requires that students perform 2 presentations in front of their fellow students and lecturers. It can be a very difficult thing to do for most people, and the experience of doing it twice can only be beneficial.

Project Review

The majority of this project involved a huge amount of trial and error. Each technique applied had to be tested, then tweaked, then tested again, to finally finish with a satisfactory product.

Overall, I am reasonably satisfied with the finished outcome of the project. Before the project, I was completely new to the area of computer vision techniques. As the project progressed I gained a solid grasp on the basics of how the techniques worked, and how best to implement them into my project.

The results from the testing on over 50 random images with different weather conditions, suggests that the correct result is given more times than not. Even though the solution is far from perfect, the amount of the sky region extracted provided enough information to make assumptions on the weather condition. I found it really difficult to find a specific set of techniques that best works in extracting the sky region for numerous images. I believe it is possible that using these computer vision techniques, that the sky region could be extracted from the majority of images analysed separately, where the techniques could be tweaked to best suit whatever information the image provides.

Something I wish that I was able to get working, was to try and identify if the sun was present in an image. Although I believe that to accomplish this would have been very difficult, I would have liked to try other computer vision techniques to see if it was possible to do so. In some cases, where the sun was present, I tried testing if the sun could be extracted through colour segmentation. In most cases, the colour values for

the sun was almost a complete white colour. So when I tested extracting the sun's colour value through numerous random images, a lot of white clouds were being extracted also. This would have resulted in quite a few false results.

If starting this project again, I would try to find a good source of information for learning the basics of computer vision techniques and test any of the examples provided for each of the different vision techniques, to quickly understand how the technique works.

I believe that my choice of technologies was best suited for this project. Using OpenCV over any other available computer vision library was an easy decision. It is the most popular and has the most resources available. The choice of using Python over C++ was a coin flip. The reason I choose to use Python was because of it's ease of and its simplicity in understanding the documentation provided by OpenCV

Acknowledgements

I would like to thank my supervisor, Paul Barry, for his help and constant feedback on the progress throughout duration of the project.

Also, I would like to thank Adrian Rosebrock who runs a website, www.pyimagesearch.com [10], where he provides detailed, easy to understand tutorials on OpenCV using the Python programming language. I found the tutorials so useful that I decided to purchase Adrian's starter package on the basic's of computer vision techniques, and it wasn't until this point, that I really felt comfortable with how to approach the techniques needed for this project. It was a huge help throughout this project.

References

- [1] OpenCV Python Tutorials, (2013), [online], Available at: <http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html> [Accessed 25 March 2016].
- [2] K Hong, bogotobogo, (2016), [online], Available at: <http://www.bogotobogo.com/OpenCV/opencv_3_tutorial_imgproc_gaussian_median_blur_bilateral_filter_image_smoothing.php> [Accessed 24 March 2016].
- [3] Eric Yuan, Bilateral Filtering,(Oct, 2013), Available at: <<http://eric-yuan.me/bilateral-filtering/>> [Accessed 28 March 2016].
- [4] Antoine Nectoux, Klein Project Blog,(May, 2015), Available at: <<http://blog.kleinproject.org/?p=588>> [Accessed 29 March 2016].
- [5] Sobel operator, (Jan, 2016), [online], Available at: <https://en.wikipedia.org/wiki/Sobel_operator> [Accessed 28 March 2016].
- [6] Sobel edge detection, [online], Available at: <<http://angeljohnsy.blogspot.com/2011/12/sobel-edge-detection.htm>> [Accessed 28 March 2016].
- [7] Pasquale D'Silva, A little about color: HSV vs. RGB, (July, 2008), [online], Available at: <https://www.kirupa.com/design/little_about_color_hsv_rgb.htm> [Accessed 02 April 2016].
- [8] OpenCV Tutorial C++, [online], Available at: <<http://opencv-srf.blogspot.ie/2010/09/object-detection-using-color-seperation.html>> [Accessed 01 April 2016].
- [9] HSV (Hue, Saturation and Value), (Nov, 2015), [online], Available at: <<http://www.tech-faq.com/hsv.html>> [Accessed 02 April 2016].
- [10] Adrian RoseBrock, PyImageSearch, [online], Available at: <<http://www.pyimagesearch.com/>> [Accessed 10 Oct 2015].