Report

Problem: Create a small application that is able to remove the sky from the videos. Essentially this means removing clouds and anything above the horizon which is not in the foreground.

Necessary:

1. use C++
    - **Status: Done**
2. use openCV
    - **Status: Done**
3. Output should be a video stream that can be played back with black or white where the sky/clouds used to be
    - **Status: Done**
    - Implemented with a trackbar that user can toggle
4. The same solution should work on both videos (you can convert to black and white if you want though)
    - **Status: Done**

Solution

My initial approach was to find edges in the video stream then use hough lines to find the horizon. I didn't have much success with that approach because the horizon is rarely a straight line in the images used (mountainous landscapes), therefore a edge detection and Hough line transform will not work.[1] [8]Also my solution needs to work in all light conditions. Thresholding (such as the Otsu thresholding) works but does not work well in low contrast conditions such as before sunrise. Fixed value thresholding does not work as the light changes too much throughout the day.[1][8]

The steps for my current solution are as follows:

1. I read each frame one by one and apply image processing algorithms to each one of them successively.
2. I equalize the histogram of the image to improve the contrast in an image, in order to stretch out the intensity range. I tried using adaptive histogram equalization with Contrast Limited Adaptive Histogram Equalization(CLAHE) however I didn't notice any improvements in my output when using it
3. I blur the image to reduce noise in the image. I do this in two ways. I apply a Gaussian blur once or I apply it multiple times while successively increasing the kernel size. Blurring multiple times slows down the performance of the algorithm however it drastically removes false positives from the resultant output image. The user can toggle between both modes using a trackbar
4. I convert the blurred image to grayscale

5. I threshold the gray image. The default value is 150 however this can be changed with a trackbar. There's a track bar that allows the user to threshold with or without Otsu's Binarization. I observed Otsu's Binarization fills out more of the sky, however it gives more false positives on the ground and it negates the effect of changing your minimum threshold value
6. I erode and dilate the resultant thresholded image with a rectangular kernel as the structural element
7. I find contours and flood fill the area above them. Previous steps should guarantee the biggest contour is above the horizon

Conclusion

I chose this approach because the problem wanted a region of the video to be replaced by another colour. This approach gives a good result however there are other factors that can be tweaked to improve the algorithm like choice and size of kernel for morphological operations. With more time my preferred solution would have been to train a Convolutional Neural Network detect horizons and fill them in.[9]
I also tried my algorithm on a video called falcon_landing.mp4 with fairly good results. Alternatively if the solution was to only detect the horizon using an edge detection algorithm like canny edge detection or sobel would suffice or an SVM with HOG if you want to be fancy

Running Instructions

cmake .
make
./iris_automation_solution [video_name.mp4]

Issues:

1. Converting to HSV and thresholding in that colour spectrum might produce a better result, however, I couldn't use that as the dynamic_test.mp4 video was already in grayscale. Converting it to HSV gives a video that is tinted with red all over, with no distinctions between land and sky.
2. Detecting only the largest contour and filling it didn't work as well as I was expecting so I didn't implement it[3]

References

1. http://stackoverflow.com/questions/30391637/finding-the-sky-ground-separation-in-opencv

2. http://docs.opencv.org/2.4/doc/tutorials/tutorials.html
3. http://harismoonamkunnu.blogspot.com/2013/06/opencv-find-biggest-contour-using-c.html
4. http://www.robindavid.fr/opencv-tutorial/chapter5-line-edge-and-contours-detection.html
5. http://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html
6. https://www.youtube.com/watch?v=os8-3uMPtlQ
7. http://docs.opencv.org/3.2.0/db/df6/tutorial_erosion_dilatation.html
8. https://www.youtube.com/watch?v=TjcFdYzg0lw
9. http://mi.eng.cam.ac.uk/projects/segnet/