Take Home Challenge: Horizon Line Detection

Overview

Our hypothetical drone is carrying a front-facing camera. We want to use it to detect the horizon line. Your challenge is to draw a single straight line across each of a series of video frames we've provided such that the straight line best represents the horizon in the image.

Expected Time: ~2 hours
Maximum Time: 3 hours (We know you could spend a long time on this project. Out of respect for your time, we ask that you not spend too long on this. We'll give you a chance to tell us how you'd improve it if you had more time.)
Please send us:

- Your code
- Instructions for running your code
- A brief explanation of how your code works
- The output frames from running your code on the provided images

After you've sent us your response, we'll setup a half hour Zoom call with you and a few Zipliners. On that call:

- We'll ask you to present your solution to us very briefly (~5-10 minutes)
- Tell us what you might do if you had more time and could work with the whole Zipline team (~5 minutes)
- Chat with us and answer any questions we have about your solution

Input Data

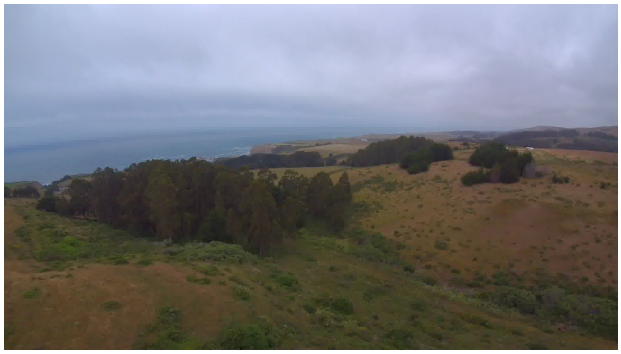
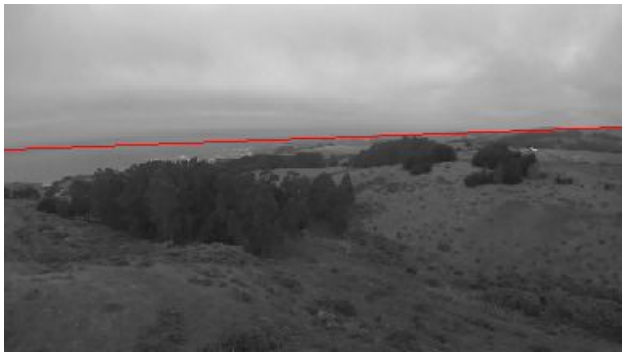
We've given you 150 frames extracted at 5 frames per second out of a video we took at our beautiful coastal engineering facility. Each frame is a JPEG image at `1920x1080` pixels sequentially labelled as `frame0001.jpg` through `frame0150.jpg`.

The Task

Your job is to draw a straight line over each frame that most closely represents the horizon in the image. When we say horizon, what we mean is where the land or sea meets the sky. Of course this won't always be a straight line in the image, but what we're looking for is the *closest straight line to following the horizon*. No curves or complex boundaries are expected or required - we're trying to keep this simple enough to be completed in a few hours.

Below on the left is an example input frame. On the right is what we would consider a correct horizon line detection for this frame. Don't stress the formatting - we're not running any auto-graders here. Make reasonable assumptions where needed and note your assumptions in the documentation. Once again, we know you could spend ages perfecting this, but, out of

respect for your time, we want to see your first pass and look forward to discussing with you all of the other awesome ideas you have.

Output Data

Your script's output should be 150 independent frames in some common image format. The output should include a copy of the image with the line shown in a contrasting color so it's reasonably easy for us to see.

In our example above, we made the output image grayscale to make it easier to see the line. You may do this but it's not required so long as it's easy to find the horizon line. The output image does not have to be the same resolution as the input images as long as it's high enough resolution for us to easily see. Bottom line: if it's easy for a human to see where the line is, it's good enough.

Code and Instructions

The instructions for running the script that you send us should explain how we can run your script on a folder of .jpg images (same resolution as the input we've given you) and have it write output frames to another folder. For example, you could tell us to run your script like this:

> My code is written in Python 3.5. Please run it like this:
> > python3 detect_horizon.py --input /path/to/images/ --output /path/to/output

You do not need to follow this exactly. Just make it clear how we can run you script.

Presenting your work to us

When we setup the Zoom call after you send us your response, we will ask you to briefly tell us about your solution. There's no required format, but you might find it helpful to show us some example frames from your output or a video/GIF of your output. Just in case it's helpful, this is an easy way to combine frames into an animated gif:

```
convert -delay 20 -loop 0 *.jpg output.gif
```

There's also plenty of websites you can use: https://ezgif.com/maker

Again, this is *not* a requirement. Just a tip in case it's handy.

Ground Rules

- You may use any language you like. We may ask you to explain your choice.
- You can use libraries as long as the libraries you use don't fully solve this problem. So, for example, a library that implements common image filters is fine, but please don't go find a horizon detection library and just call that.
- We still expect you to understand what's happening inside any library functions you call, at least at a high level.
- Feel free to look things up online, but you must write all of the code you submit. If you're using some helpful reference, please cite it in your documentation.
- You are free to test out your solution on other images, but please don't introduce outside data for any purpose other than testing edge cases. If you find yourself tempted to do this, remember that, if we invite you onsite, we'll give you a chance to tell us what you'd do with more resources and time.
- A simplified-but-working solution is better than something that tries to do everything but doesn't actually work.
- Please don't spend more than the maximum time (3 hours) on this take-home.

If you have any questions, just get in touch with us and we'll be happy to help answer them.