

Homework Assignment 3

CS223B, Winter 2005-06, Stanford University

Sebastian Thrun, Greg Corrado, Dan Maynes-Aminzade, Mitul Saha

cs223b@gmail.com

Due: Wednesday, February 15, 11:59 PM

Assignment Overview

This assignment is worth 10% of your total grade. It will be graded out of a total of 100 points, and consists entirely of one programming problem.

1: Motion Tracking (100 Points)

Objective

In this problem you will be asked to identify *moving* cars from video clips taken in typical driving situations. This would be trivial if your camera was stationary, but because the camera itself is in motion, identifying moving cars will be a challenge worthy of your creativity and determination. Your code will process a video stream, and output a series of monochromatic images in which each pixel is tagged as MOVING (white) or STATIONARY (black).

What We Provide

We provide 6 video clips, each exactly 10 seconds in length. For the sake of compatibility we've converted each video file to a series of JPEGs for you. The original video clips were taken at 30 frames per second, so each clip has exactly 300 JPEG frames. The frames from clip A are named `clipAxxx.jpg`, where `xxx` is the frame number. For example the files in clip 2 are named:

`clip2001.jpg`, `clip2002.jpg`, `clip2003.jpg`, ... `clip2300.jpg`

The still images are available at:

<http://cs223b.stanford.edu/homework/hw3/video/>

What Your Code Should Do

Your code should load in the data and produce an output image at the end of every second, which is every 30th frame (i.e. frames 30, 60, 90, 120...). This output images should be at the same resolution as the video frames, but with all pixels either black or white. Pixels should be white if your algorithm has classified them to be contained within a car which is moving relative to the earth. Pixels should be black if they are either not part of car or part of a stationary object. For

example, cars which are driving alongside our car or traveling on the roadway in other directions should be white. The ground, the sky, building, trees, and parked cars should be black.

(Note: In the video clips the hood of the car our camera is attached to is always visible. These pixels are both uninteresting and potentially confusing. So we are going to resolve any ambiguity and declare that these pixels should be classified as black. It is perfectly acceptable to hard-code a mask which forces this region of the output images to black at the very end processing.)

For each clip **A** these output images should be named **outXXXX.png**, where **XXX** is their original frame number. Thus for clip 2, you should have output files named:

out2030.png, out2060.png, out2090.png, ... out2300.png

As in Assignment 1, we stipulate that you produce output images in PNG format so that all of pixels are pure white or black.

What to Hand In

You will electronically submit the following items:

- A **CODE** directory which contains all of the code you wrote and any additional unusual source we would need to run your submission. Please make an effort to submit neat, commented code so that we can give partial credit.
- A **README.txt/pdf** file containing instructions for compiling and running your source code for Problem 1. If we are unable to run your code locally, we will bother you until we can.
- A **WRITEUP.txt/pdf** file containing a description of your approach. State your methods for solving the problem and explain what worked and what didn't. Even if your method didn't ultimately perform as well as others, you can still score well by arguing why your approach was a reasonable or clever one. Include the names and email addresses of all of your group members.
- An **OUTPUT** directory containing the output PNG images produced by your code as described above. You should have a total of 60 images, 10 for each of the six 10-second clips. We will use these images to evaluate the performance of your solution.
- We encourage you to also include a **SCRATCH** directory, containing movies or images of interesting intermediate stages of processing your algorithm produces. (For example you might submit a movie like the one shown in class with optical flow vectors superimposed over the raw video.) *Submitting these files is strictly optional, and is not required for your submission to be considered complete.* We encourage you to include them for two reasons: If your algorithm performs very well, these will be the sorts of things we show in class. On the other hand if your algorithm performs poorly, seeing these images will help us better understand your approach and what it was able to do well and allow us to grade you more fairly.

To submit your completed assignment, email a single ZIP, TAR, or SIT archive containing all of the above components to `cs223b+submit@gmail.com`. You must include the `CODE`, `README`, `WRITEUP`, and `OUTPUT` for your assignment to be considered complete; we will not accept partial submissions.

Grading

We will grade your solution based on the same two metrics as Assignment 1:

1. *Pixel-by-pixel comparison to hand-labeled frames.* As in Assignment 1, one of the diligent course TAs will manually classify the pixels for each of the output images you are required to submit. For each pixel in your output image that matches the hand-labeled image your score will be incremented. Thus a better match between your output and ours, the higher your score.

The one change from the scoring procedure is that not all of the output images you submit will contribute to your score. Specifically the first two output images for each clip (`clipZ_030.png` and `clipZ_060.png`), will not be used by the scoring script. You are still required to submit these images, though their pixel classifications will not be counted. We do this because several approaches you might consider using in this project might produce unstable results before a second or two of video has elapsed.

2. *Documentation and approach.* We can't stress how important the `WRITEUP` is. Very good ideas often don't work perfectly the first pass at implementing them, and uninteresting hacks sometimes perform well on small data sets using a simple metric like pixel correspondence. If you have good ideas that you can describe clearly, your grade will improve. If you do something brain-dead that happens to work well on this dataset, your grade will suffer. Use the optional `SCRATCH` directory to include any images or videos you refer to in your writeup. Don't leave this for the 30 minutes before you submit; you'll help yourself a lot by writing some of it up along the way as you try out different ideas.

Some Advice

This is a challenging and very open ended assignment, but you only have a limited amount of time to complete it. That means be creative, but think carefully about what you can realistically implement and what you cannot. No assignment will be perfect, and you can expect the percentage of pixels you are able to correctly classify to be lower than it was in Assignment 1. You are free to leverage your programming work from the first homework assignment, but you don't have to. If you choose to reuse your code for the first assignment, even if you only use it as a pre-filter/post-filter to clean up your results, you must *resubmit* all that code again with this assignment.

When you look at the video clips we've provided, you'll see that we're *not* trying to trip you up with moving objects that aren't cars. Because this assignment is primarily about motion, and nearly all the stuff that's moving in the clips *are* cars, it's possible to have one of the best submissions for this assignment even if your submission from the first project is not one you're eager to build on.

While we're not trying to confuse you with pedestrians or bicyclists, you will notice that the video clips we provide are taken over a range of different conditions and test your algorithm in different, perhaps cruel, ways. Again this is designed to get you focused on the interesting problem of extracting real motion information rather than collecting hacks like "If there's a car right in front of me it's probably moving" or "About 10% of the pixels in these videos are moving cars."

Honor Code

You are free to use MATLAB Toolboxes, OpenCV, or any other computer vision libraries. Obviously you must mention what tools you used in your WRITEUP. For example, if you used MATLAB's optical flow tools you should give them credit for it.

Because the clips are so different, it might be tempting to manually adjust parameters in your code for each clip, or worse, give the code some explicit dependence on the clip name or length. Of course we expect that your code will produce the all the output images exactly as submitted without human intervention or trickery. Anything less is unacceptable and will be considered a flagrant honor code violation.