B490/659 Project 5: Applying vision to a real problem

Spring 2015

Due: Friday May 8, 12:15PM **(15 minutes after noon)**

We've covered a lot of ground in this class, discussing many fundamental techniques in image processing and recognition. This assignment will ask you to put some of these concepts together to help solve a real vision problem. Unlike past projects, this one is open-ended. Your goal is to take a problem and design a vision system that tries to solve it. Because of this structure, this assignment has several differences from past ones:

— You may choose your own teams for this assignment. You may work in groups of 1 to 4 people, and undergraduate and graduate students may be mixed on the same team if you choose. We expect that larger teams will be able to produce more complete and higher-quality work, and will grade accordingly.

— A key part of this project is your report, which should describe your approach in detail, explain why you chose it, and give experimental results and discussion. To reflect this expectation, the report for this project should be in a formal format (submitted as a PDF) and will be worth approximately half of the project grade.

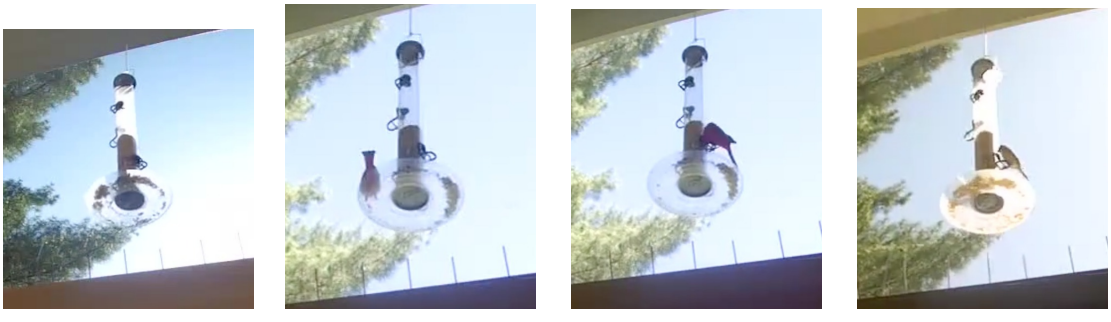
— You *may* use existing code libraries for computer vision and image processing, or languages other than C/C++. We expect that you will still do significant effort yourself, so your project report must explicitly describe any code or libraries that you used, and explicitly state your contributions on top of them.

— We have not created github repositories for you; the code for this assignment will be submitted via OnCourse. We encourage you to use GitHub for collaborating with your team, but this is not a requirement. One person should upload your team's submission via OnCourse.

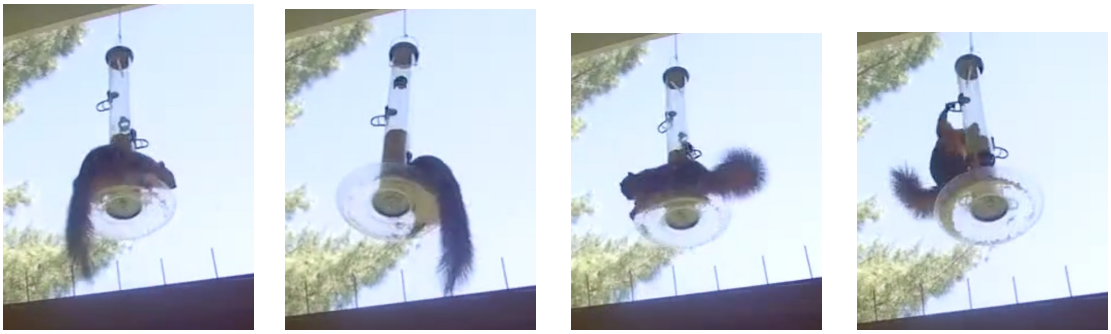
The problem

A certain professor of Computer Science at a certain major research university enjoys living in a certain quiet midwestern town. He likes watching birds, and in fact recently bought a large bird feeder that he hung outside his front window. The large feeder has attracted a variety of interesting birds, but also an unwanted guest: a squirrel which jumps on the feeder and eats most of the seed. To monitor the situation, the professor has installed a camera to collect video of the feeder, and would like a computer vision system to tell him when the feeder is empty, when the feeder has bird(s) visiting (so that he can go to the window to watch), and when the squirrel has arrived (so that he can run yell at it to scare it away).

Your goal is to design and prototype a computer vision system that can classify frames of the bird feeder video into one of these three categories (see Figure 1). Ideally the system should be fully-automatic, requiring no human intervention, but the professor is willing to do an occasional small amount of work if needed (like marking the location of the feeder within the first frame of the video or confirming that there is or is not a squirrel in an occasional frame) as long as most of the work is done automatically.



Empty feeder | A bird | Another bird | How cute!



Squirrel | The squirrel again | More stupid squirrel | Seriously?!

Figure 1: Sample (cropped) images from a bird feeder video.

What to do

Your goal in this project is to design an approach to try to solve the professor's task, to implement a prototype of the approach, and to characterize how well the approach will work through some experiments. Unlike past projects, our goal here is less to produce a complete implementation, and more about designing an approach, explaining why you chose this approach, and quickly prototyping the key parts to see how its accuracy might be in practice. You *do not* need to necessarily deliver a complete end-to-end, polished system. For instance, maybe you'll write a program to do some preprocessing of images, then use or modify an existing implementation of a compiuter vision algorithm, then have another program that will do some post-processing.

Think of your report as sort of a proposal to get funding from a client, to show a well thought-out approach and enough initial results to convince them that your approach will work and that you deserve money to develop the system into a working product. It should include:

1. A detailed description of your approach, including a block diagram to describe the various steps your approach takes, a discussion of how it works, and an explanation of why you chose this particular approach and other approaches you may have tried or considered but discarded. (By "detailed," we mean that someone who has taken this class would be able to read your description and then know how to implement your approach. It does not have to explain how to compute convolutions or to explain how SIFT works, for example.)

2. An overview of your prototype implementation, including which parts you implemented yourself and which parts you used from others. It is fine if most of the code was implemented by others, and your contribution is to modify it and test it in this application. It is also fine if your code was largely written by you. Please tell us the steps to run or reproduce your work, including which libraries or external programs we need to make it work.

3. Sample results, both quantitative (sample images) and qualitative (accuracy numbers, measured in some reasonable way). How does your approach compare to simple baseline techniques, like simply randomly guessing or classifying every photo as an empty feeder?

4. A discussion about your results, including predicting how well your technique (if fully implemented) would work in practice, and speculation on ways of making it better. Explain the assumptions of your approach and any other limitations.

The approach you propose is up to you. You can use techniques we saw in class, explore other techniques, or develop something new. You can implement the prototype using your own code, maybe including code you wrote for an earlier project or using some libraries like CImg or OpenCV, or you can explore using research code available on the web (e.g. OpenCV detectors, Felzenszwalb part-based detector, Dalal-Triggs detector, GIST or HOG features, Theano deep learning toolkit, etc.), or any combination of these. Although it's obviously best if your approach produces good results, you can still get a good grade even if it doesn't, as long as you've proposed and rationalized an interesting and reasonable approach, implemented some key portions of it, present some experimental results, and submit a high-quality report.

To help get you started, we will provide some sample video files on OnCourse. Instead of trying to deal with video file formats, we suggest simply converting the video to a sequence of individual .jpg files and operating on those instead. The following command on the CS Linux machines will do this for you:

```
ffmpeg -i input.mp4 'frame%04d.jpg'
```

You don't necessarily need to use all of the images in the videos, although it depends on your approach; some approaches may need a lot of images (e.g. if your approach looks at differences between adjacent frames to find motion, or if it requires a lot of training images), while others may need relatively few.

You'll likely encounter some decisions as you write your programs that are not specified in this assignment. Feel free to make some reasonable assumptions — just state them in your report.

What to turn in

Submit **to OnCourse** two files: (1) a PDF containing your report, and (2) a ZIP file containing the source code.