PROJECT 1
IMAGE THRESHOLDING AND BLOB TRACKING

OVERVIEW

For this project, you will build a system to:

- threshold a sequence of color or grayscale images to distinguish objects of interest from the background.
- apply morphological operators to the thresholded images in order to remove noise and imperfections.
- perform a connected components analysis to distinguish between separate objects, and to identify their locations in the image.
- track the connected components over time.

The last two items in the list above are collectively known as blob tracking.

TASKS

Thresholding. Your system will produce a thresholded binary image where the non-zero pixels correspond to objects of interest (foreground), and the zero pixels correspond to background. You will need to decide on some details of your thresholding approach:

Averaging. Instead of straightforwardly thresholding each frame, your system may need to perform some averaging in order to improve performance. We discussed two averaging strategies in class: spatial and temporal. In the spatial averaging approach, also known as adaptive thresholding, the system looks at the difference between a given pixel value and the average of the pixels around it. OpenCV implements adaptive thresholding with the adaptiveThreshold function. In the temporal averaging approach, the system looks at the difference between a given pixel value at some location and the average intensity of that location over time. You can take a temporal average by simply adding together multiple frames and dividing by the number of frames; however, be careful about overflow. In practice, this means you will want to convert frames from 8-bit integer format to a representation with more precision before summing them together. To perform a straightforward threshold on an image, you can use the OpenCV threshold function.

RGB thresholding. In class, we discussed a number of ways to threshold RGB-valued pixels. The simplest is to convert RGB to grayscale and threshold accordingly. Other methods include a planar decision boundary or distance from a reference RGB value. Although you may be tempted to write a loop over each pixel in the image, you might be able to find some OpenCV or NumPy functions to perform the equivalent functionality more quickly.
Morphological operators. If your thresholded image contains speckles or noise, you will likely want to apply some morphological operators on it to clean it up. OpenCV provides the \texttt{erode}, \texttt{dilate}, and \texttt{morphologyEx} functions to implement erosion, dilation, and opening/closing, respectively. Your goal here is to produce the best possible image to send into the next stage of the processing pipeline.

Connected components analysis. The OpenCV function \texttt{findContours} retrieves the outlines of connected components of non-zero pixels. Applied to your thresholded image, this corresponds to outlines of the objects of interest in your scene. Some additional analysis of the contours yields information such as the area, and centroid of each connected component (see the \texttt{regions.py} example from the course webpage for details).

Tracking. Your system should, at minimum, extract the position of each object’s centroid in each frame. Better yet, it should also be able to track objects’ trajectories over time by associating the connected components in the current frame with those of the previous frame (note that this is trivial in scenes containing a single object).

Scenarios. Your system should be targeted at some particular scenario. Here are some examples, ordered roughly in increasing difficulty:

- A single brightly colored object moving through the scene. Example: the video of the bright green cup that I showed in class.
- Multiple brightly colored objects moving through the scene. Example: video of someone juggling two or more differently colored balls.
- A single, arbitrarily colored object, tracked using temporal averaging. Example: the cat video I showed in class.
- Multiple objects tracked using temporal averaging. Example: the fruit flies video from class.

You should be careful to pick a scenario that is feasible given your programming ability and the time available. If you have questions about picking a particular scenario, or coming up with your own, don’t hesitate to ask.

EVALUATION CRITERIA

Your project will be evaluated by the following criteria (percentages are approximate):

\textbf{a. source code and raw data (15\%)} - Turn in the full source code for your project, including any makefiles or Visual Studio project files needed to compile it. I expect your code to be neatly indented and reasonably commented. Unless the data was provided by me, you should
also turn at least two sets of raw data (movie files and any additional input necessary) that can be supplied to your program as input.

b. **thresholding and morphological operators (25%)** - Your system should be able to output the binary image after thresholding and morphological operators have been applied, but before binary segmentation. The output should be relatively free of noise and speckles, with the objects of interest well distinguished from the background. Submit a set of several representative images from this intermediate data.

c. **binary segmentation and tracking (25%)** - Produce plots of the positions of the objects of interest over time. At minimum, you should have a single \((x, y)\) plot of disconnected points showing the positions of the centroids of the connected components in the thresholded images. Better yet, produce a plot of connected points showing the *trajectory* of each tracked object over time.

d. **performance (5%)** - Your system should run in real-time or close to it. I expect it to be able to process 640×480 video at least 10 frames per second (note that most video cameras generate output at about 30 fps).

e. **“cool factor” (10%)** - Go a little above and beyond the tasks described in the section above. Possibilities include allowing parameters of the system to be modified interactively, visualizing some aspect of the data not described above, or doing something really creative or cool.

f. **written report (20%)** - Turn in a written report in PDF format describing your overall approach and its effectiveness. In addition to addressing the criteria listed here, your report should also address some questions about generality. How broad a class of data can your system work on? What assumptions are encoded in your methods? Would your system work with objects of different colors? In changing brightness conditions?

Instructions for submitting your project online will be provided.