Comments on Lab7 - Robert Jauquet

General section discussion:

There are a few areas that need to be improved to make this algorithm 100% true to the description. The two biggest problems I ran into were as follows.

1. Trying to use the vx and vy vectors given by the optical flow to determine where the user points correspoded was most difficult. At first I didn't realize I was using two different coordinate systems. The optical flow returned a coordinate system in the first quadrant, while the user inputs placed the origin at the upper left which made all the gathered points in the fourth quadrant. After resolving just that portion, the output data improved substantially, but there were still a few issues.
   The problem is that the optical flow doesn't give a translation of which pixel goes to which other pixel, but instead gives values of overall movement within the picture, so
   that one cannot just take the magnitude of the x and y vectors at a single point and expect to use this distance in a useful translation to the movement of that pixel. It worked in a general sense in this way, but I could not come up with a consistant change that would allow the computation of the distance to be comparable to that which is returned by 'graphShortestPath'. The best solution I can think of would be to write a new optical flow computation that, for every pixel, returns the new pixel location.

2. Along with the last issue, the next problem was trying to figure out what path the selected pixel actually traveled. I wanted to use the vx and vy vectors to compute the actual points at which the pixel was located, but for the reasons described above this was not a viable solution given the current method of computing optical flow. Instead, I took each frame that was going to be added to the output, and computed the change from the nearest point selected by the user using vx and vy for that point. This gives a rough idea of how off each point is, but does not fully give the path that the pixel follows.

Advaced section discussion:

For the advanced sections, I chose to do the following:

1. Synch the video with audio. At first, I thought this was going to be a simple matter of finding a usable Matlab fuction to add audio to .avi files. It seemed very logical that after already being able to create an avi format file, that it would be simple to just add the audio and be done. This was obviously not the case, as it seems there is suprisingly no easy way to have both audio and video inserted to the same avi file in MATLAB. I read that the current version does have functionality, but there was no documentation available and the produced code would not work on all machines.

   Because of this, I decided to try something a bit more simple. I played around with MATLABs audio functions, and thought it would be quite easy to display sections of the video and have a bell chime at every user point before continuing with the video. Most of the needed information was already implemented in my code, so it was just a matter of figuring out how to time the audio with the frames.
   MATLABs movie function has a useful optional variable to select which frames of a movie you want to play. Along with that, the play function has a useful feature that halts other computation until the audio stops playing. In this way, it was just a matter of using a simple for loop to get a gong to play at every user input in the video. The only downside to this is that the collection is not exportable, so you have

to be at the computer when computation finishes to view the output frames with the audio.
After the inital computation, it is just a matter of copying over the for loop to the command line where the workspace is saved to get the video to play again.

MATLAB also includes some timing variables, so to improve on this audio implementation,
it would be useful to have the audio playing while the video is running but change the frame rate of each line segment so that the section of frames is displayed exactly over
the length of the audio each time.

2. New data. The last thing I atempted to do was to collect new data and see if I could improve on the output video. I knew that the runtime of the algorithm was determined mostly by the number of pictures being used, so I decided I would try to get a working computation using fewer images. To do this, I used a lego figure to take the pictures of its head moving left and right, as well as his body moving forwards and backwards. This created images that are very similar compared to pictures of a moving human head, so the threshold of the similarity function could be reduced substatially. The problem
with using Legos however, is the lack of defined facial features. For the most part the
direction was easily computed, but smooth movements were considerably more difficult to
achieve. Because of this, I've concluded that to use this algorithm Legos, it would be
better served to track well defined features, like the arms and hair. Possible other uses would be to have an interactive, buildable lego structure where pieces could be connected in the same way the faces are moved around.