In this machine problem, the goal was to create a program that would track an object, in particular a girl's face in the video. This was accomplished, with three separate results based on evaluation criteria. The overall program is broken down into eight different functions. The first is a preprocessing function which creates a 4th order tensor from a series of images that is really just a collection of the images for easier processing. Two additional auxiliary functions provide the ability to draw a box on a given image, given the top left corner's position and the width and height of the box. A similar function allows you to get a subimage given the same information. Three evaluation functions were created, one that does the calculation for cross correlation, one that does the sum of the squared difference, and one that does normalized cross correlation, all three have the invariant that their two images that are inputs must be the same size.

The overall function follows the following algorithm:

1. Get the template for the object from the first frame (the location and size of the object are inputs), draw the box around the object in the first frame.
2. In the next frame, do an exhaustive search of the surrounding region (the size of the surrounding region is also an argument of the overall function, in particular it's a box around the former location of the object plus/minus the argument in each direction). Using the given eval function (which is also an input), find the best match to the template.
3. Draw the box around the best match, and update the template to be this best match.
4. Repeat steps 2 and 3 until you run out of frames.

Thus, tracking is achieved. The final function I'd mentioned just takes the tensor and converts it into the video file.

As far as the results are concerned, normalized cross correlation seems to yield the most stable tracking, in that the box more or less stays with the object without jumping around at all, however this also means that normalized cross correlation is incredibly vulnerable to occlusion. Cross correlation and sum of squared differences yielded less stable results, the latter in particular jumped around a lot to random points in the video, however both do mostly follow the object. As a tradeoff (and hopefully earning me the extra points), both cross correlation and sum of squared differences do not follow the occluding face, staying as fixed to the original face as they normally would. That is, the sum of squared differences and cross correlation are robustly immune to occlusion in this case. Potentially fixes to address occlusion to allow normalized cross correlation to deal with it could involve restricting the search window to the direction of motion (calculated by finding the difference between current and previous locations of the object), or by using multiple templates and using a weighted summation.