What I’ve done:

1. Ported all the mex TSP code from C++ to MATLAB.
2. Improved efficiency by reducing function calls, using structs less, and using efficient for loops.
3. Solved the problem of the long thin lines by designing and implementing an algorithm that
   1. Checks to see if trying to move a pixel that is part of a line
   2. Sees how big the line is
   3. If it’s over a threshold, splits the line at what is most likely the base of the larger portion of the TSP
   4. Moves all the pixels from the line to neighboring TSPs
      1. If it’s moved more than a threshold, it decides that the bigger part of the TSP was at the other side of the line, so it goes back and splits it at the other end
      2. If it finds that the TSP is a loop, it just gets rid of the line, because the TSP is connected on the other side
4. Introduction
   1. Tracking objects from motion can’t be done just looking frame-by-frame—not enough information. (show an example of frame-by-frame)
5. Use many frames
   1. Check several frames into the future and into the past to find large-scale movement. [1]
6. Look forward and backwards
   1. Use information from both backward flow (frame 2 to 1) and forwards flow (frame 2 to 3). [2]
   2. Use shape from frames in front and behind. [3]
   3. Smooth direction of movement by looking at several frames instead of just the current one.
7. Interpolate
   1. Interpolate missing frames of movement in an object. [4]
   2. Combine multiple objects that are really the same.
8. Future work
   1. Right now this only works with a stationary frame of reference—what do we want the results to be when we have a moving frame of reference?
   2. Combine these results with image segmentation.
   3. Combine the large-scale results (looking at every 5th frame) with short-term results to get a more accurate outline.

Code outline!

1. Find objects
   1. For each frame…
      1. Get Boolean mask of high flow areas
         1. Get forward flow
         2. Get backward flow
         3. Get intersection of those flows
      2. Find each connected area of that mask. For each of these masks…
         1. Eliminate it if it’s too small
         2. Expand the mask to see the surrounding area
         3. Find the median theta of the flow of the mask
         4. Remove any parts of the mask whose flow is not within a tolerance of the median flow
         5. Eliminate it if it’s too scattered
         6. Check each object. For each one…
            1. If it’s dead or doesn’t have any masks before the frame in question, continue
            2. Check several possible thetas to see if the current mask is a continuation of this object. Save the best sameness rating
         7. Check the best sameness ratings of the other objects
            1. If this mask is similar enough to another object, add it to that object
            2. Otherwise, create a new object
      3. Tag any objects that haven’t been updated in a while so we don’t check them for continuation anymore
   2. Return the objects found
2. Combine objects
   1. For each object…
      1. For each subsequent object…
         1. Check to see if the subsequent object is a continuation of the original object by finding the best overlap from several thetas and also checking the difference in median theta of the objects
         2. If the objects are similar enough, merge them
3. Interpolate objects
   1. For each object
      1. Eliminate it if it has fewer than the minimum number of frames
      2. Fill in missing masks between found masks
4. Find intermediate masks between rough masks to fill in the rest of the frames