**Object Tracking from Sparse Optical Flow**

**Introduction:**

Tracking objects from motion cannot be done effectively when only one frame is considered at a time because there is simply not enough information. My approach is to look several frames ahead and behind each frame to see if there is consistent movement, and if there is, to see if that movement falls into a larger pattern in the clip. By using more information than just the optical flow between a frame and the frame immediately before, I am able to track objects throughout a video clip while ignoring incidental motion and noise. In addition, because I do not have to look at every frame, my method requires significantly less computation than one that reads every frame [needs citations from other work in the field].

Figure : Results from a short video, where the found objects are highlighted.

**Using many frames**

Instead of using the optical flow from consecutive frames, we use the flow from several frames apart, usually one second. Although this yields less detail, it still finds the major objects in the image, and can be done in one tenth (or whatever the frame increment is) computation time. In the figure below, the image on the left is the current frame, the image in the middle is the horizontal flow between the current frame and the previous frame, and the figure on the right is the flow between this frame and the one ten frames before. In both slides you can clearly see the man (the blue shape), and although there is slightly more noise when the frame increment is increased, this noise can be easily eliminated.

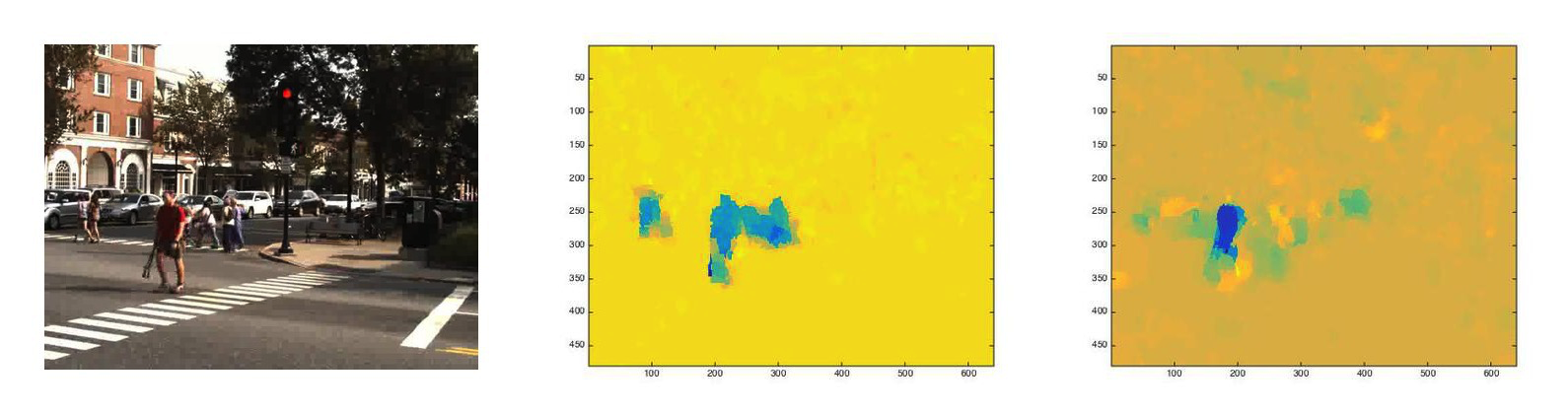


Figure : Optical flow between frames one frame and ten frames apart.

**Looking forward and backward**

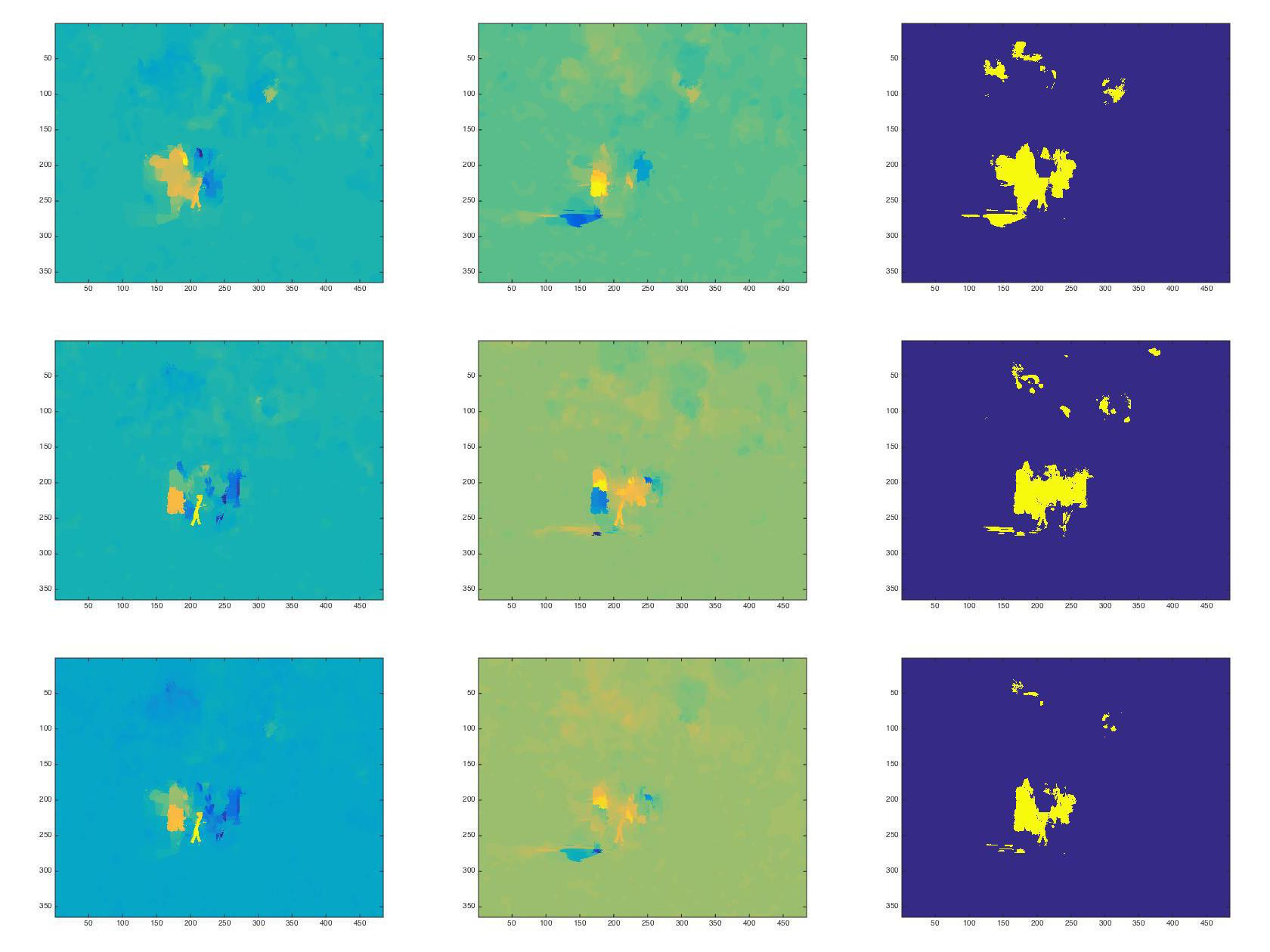
Because there is not enough information to find and track objects in an individual frame, we must use additional information. One way to do this is by looking both forwards and backwards when finding optical flow. My code uses both information from the flow of a given frame to the next frame and the given frame to the previous frame. It creates a Boolean mask from both of these, where high flow values are true and low flow values are false. It then takes the intersection of these masks to get a more accurate prediction of where the object is. In Figure 2, the left column is horizontal optical flow, the middle column is vertical optical flow, and the right column is the mask created from the optical flow. The top row is flow from the current frame to the previous frame, the middle row is flow from the current frame to the next frame, and the bottom row is a combination of the first two using the sum of the flows and the intersection of the masks.

Figure : Forward and backward optical flow

**Filtering masks using direction of movement**

Once a mask is found for an image, it is split up into individual masks based on connectivity. Each of these masks is expanded so that nothing is missed, and then the median direction of motion is found based on the summed optical flow corresponding to this mask. Then, all areas of the mask that are not within a certain percentage of the median direction of motion are eliminated.

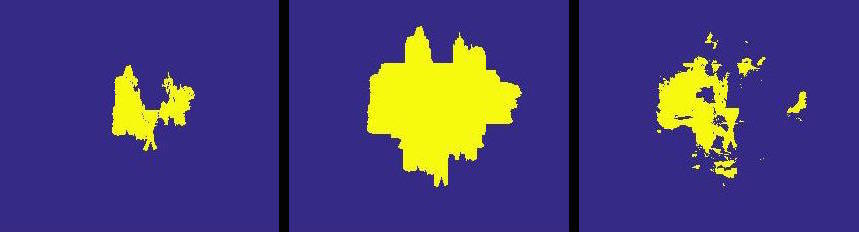


Figure : On the left is a connected area of the full mask, in the middle is the expanded version of this mask, and on the right is the mask with differing directions eliminated.

**Calculating probability that a mask is a perpetuation of a previously found object**

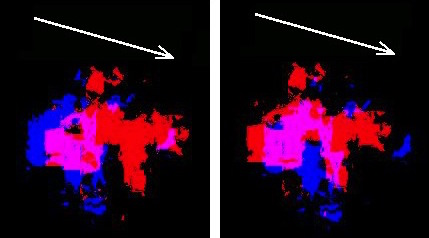
Once a mask has been found, it is checked against all previously found objects to see if it is likely to be part of any of them. This calculation is done by finding the maximum overlap possible between two frames when one is moved along its direction of movement. Because the direction is often inaccurate, several possible directions are used to mitigate error. In the following figure, the white arrow is the direction of motion, the blue is one object, the red is another object, and the purple is the overlap of these objects. On the right we see the objects before any movement takes place, and on the left we see the maximum overlap possible to attain while moving the blue object in the direction of motion.

Figure : Finding overlap between objects.

**Use masks from after and before to get a more accurate shape**

Instead of only using the mask from the current frame as the shape, we combine it with the masks from before and after. We find the maximum overlap of these masks, and set the final mask to be wherever two of these three masks (before, current, and after) are. In the figure below, the masks are red, green, and blue, and the other colors correspond to overlap (for example, purple is the overlap of red and blue). The leftmost figure shows the masks before the maximum overlap is found, the middle figure shows after the maximum overlap is found, and the rightmost image is the final mask resulting from the overlap.



Figure : Using overlap from before and after for a more accurate shape.

**Interpolating missing masks**

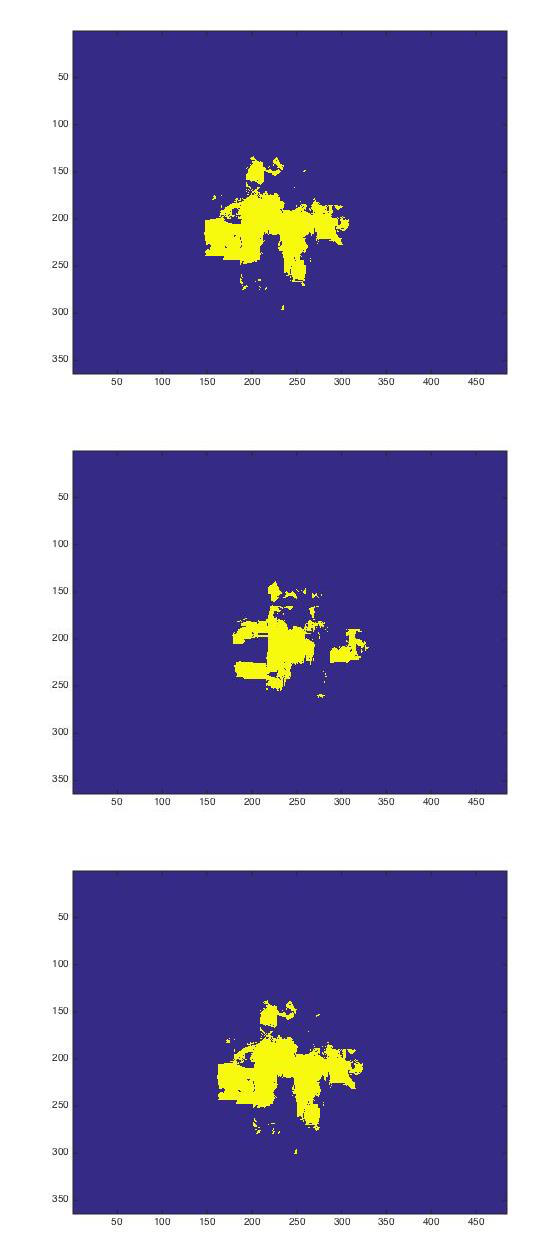
Sometimes there are frames that are missing masks for an object, but have masks for that object before and after them. In these situations, the missing mask is interpolated. In the below image, the mask at the top and the middle had been found, but there was a frame between them with no mask. The mask at the bottom was interpolated for the middle frame by moving the top mask along its direction of motion.

Figure : Interpolating missing masks

**Future Work**

-Figure out how to handle a moving camera.

-Integrate TSP segmentation in masked areas.

-Use color in addition to optical flow to track objects better.

-Differentiate between two moving objects that are next to each other.

-In addition to the larger flow increments, use smaller flow increments (one or two frames instead of 10) to find faster moving objects, objects that move more erratically (like a person’s arms and legs), and smaller objects.