CMSC426 Project 3

Rotobrush

Group Members: Marc Haller, Anthony Mahshigian, Matthew Wong

Project Overview:

Parameters

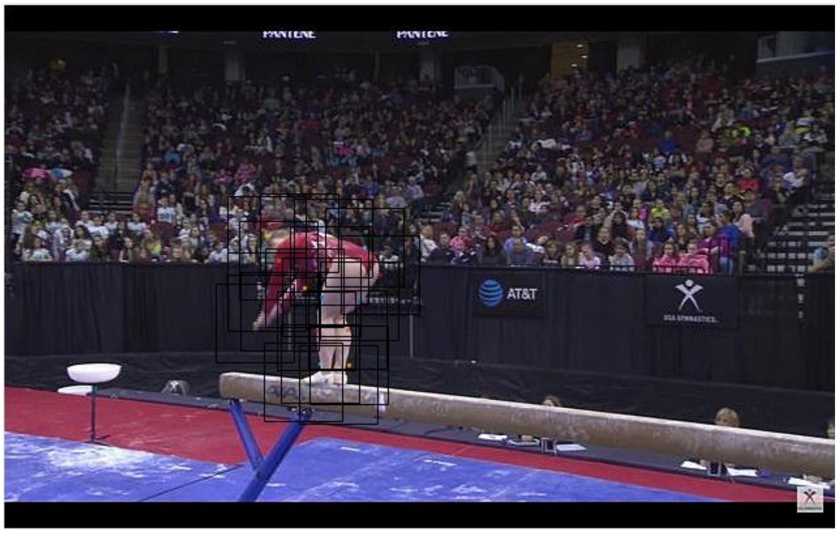
Input Files:
The program looks through the input folder with the given naming scheme of frames, e.g. 1.jpg, 2.jpg, etc. The code also uses a mask file to have the initial mask formatted as a .bmp file. We also have the option in our code to use roipoly to make the mask yourself, but as we found this was tedious for testing, we saved our masks.

Other parameters:

- WindowWidth: This specifies the size of each local window that we use to segment the foreground object. E.g. if WindowWidth is 40, the local windows will be 40x40
- ProbMaskThreshold: This is the threshold for probabilities of pixels to be a part of the final mask in updateModels
- NumWindows: This specifies how many local windows will be spaced around the mask outline (contour)
- BoundaryWidth: This specifies the range of pixels to be ignored near the boundary
- Fcutoff: This is the threshold for color confidences used in the calculation of sigma_s when creating shape models. In our code, we used .85
- SigmaMin: This specifies the minimum sigma_s allowed in our code. Also used in the calculation of sigma_s even when not directly used. In our code, we used 2
- SigmaMax: This is used in the calculation of sigma_s. In our code, we set it to WindowWidth
- R: This was used in the calculation of sigma_s. It is the power to which you raise the difference between the color confidence and fcutoff. In our code, we used 2
- A: This isn't a set parameter, but it is a constant used in the calculation of sigma_s. In our code, it is (SigmaMax-SigmaMin)/((1-fcutoff)^R)

Initializing Local Windows:

While this code was given to us, it is important to understand what this function (initLocalWindows.m) does. This function is for initializing the local windows, the windows with which we will surround the contour of the mask completely and create our models to update the mask outline. It is important that the entire contour of the mask is covered as otherwise, part of the mask outline will not be used in calculations and our models will be off. Plus, we won't be able to set areas not covered by the windows as a new mask. This function will initialize NumWindows windows of size WindowWidth x WindowWidth evenly spaced around the outline.



Example of Local Windows surrounding foreground object

Initializing Color Model and Confidence:

While colors and shapes can change, we can create models that describe the color and shape of the foreground and background for every local window. To start, we initialized a color model based on the initial given mask that will later be updated by updateModels as described later. For every window, we sample every pixel and based on the logical mask, we add it to an array for foreground and background pixels if that mask coordinate is 1 or 0 respectively. This gives us a sample for foreground and background with which to build a distribution. We run fitGMM for both sets and using pdf, we find a p_c for each pixel, which describes the probability of the given pixel to be in the foreground. This is our color model. From then, we use the given formula based on p_c and d (given by bwdist), we calculate the color confidence for each window, which describe the confidence of the color model being sufficient for detecting the foreground.

Example of Color Model of a Local Window
(Window 5 for Frame Set 3)

Initializing Shape Model and Confidence:

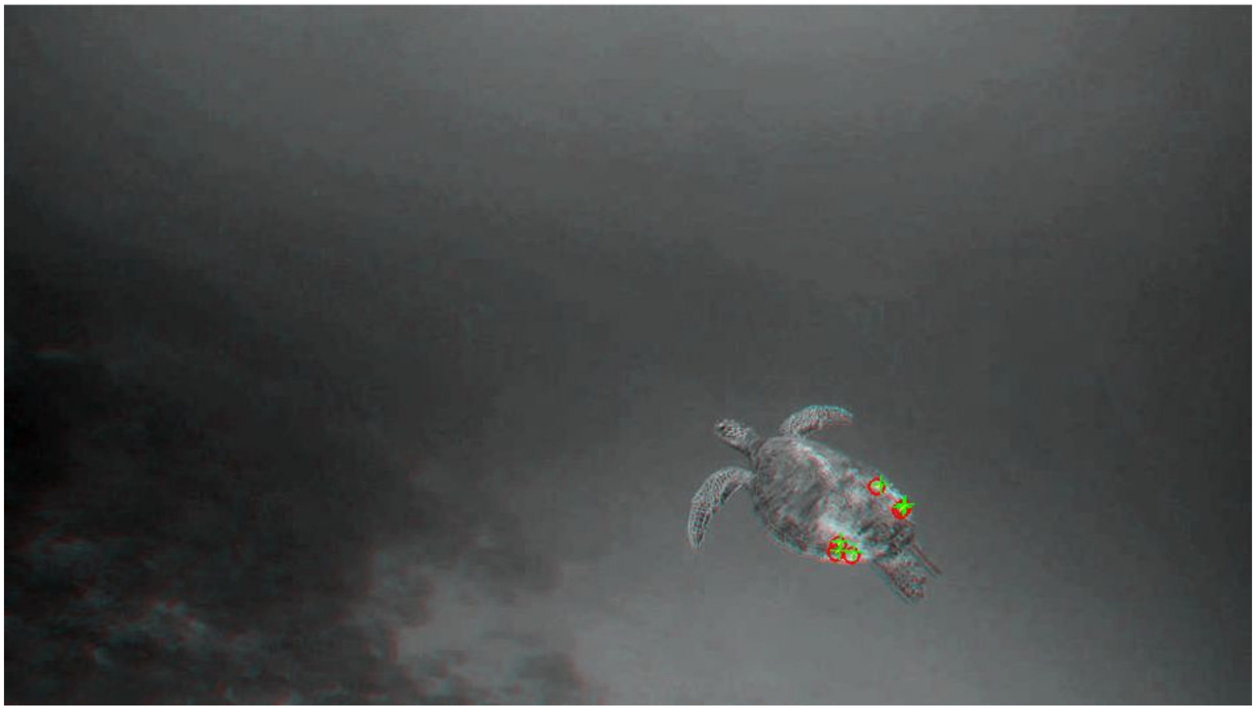
We then create our shape model to output a list of shape confidences for each window. The calculation is very simple, but depends on a lot of parameters. As mentioned often in the parameters section, this is where sigma_s is calculated to give us our shape confidence. Fcutoff will give us a threshold for accepting color confidences. If the color confidence is too low, we simply use SigmaMin. Otherwise, we add A*(fc-fcutoff)^R to SigmaMin for sigma_s. Then we calculate the shape confidence for each pixel based on its distance to the mask outline and the calculated sigma_s.

Example of Shape Model of a Local Window
(Window 5 for Frame Set 3)

Updating Window Locations:

There were two parts to the issue of updating the windows. We calculated a global homography matrix (calculateGlobalAffine.m) as well as an optical flow (localFlowWarp.m).

For the global affine, we tried using vl_sift as the algorithm the project is based on uses, but we had challenges. There were errors in the libraries that we had difficulty debugging, so we ended up using extractFeatures to get features from each image. We then used matchFeatues and estimateGeometricTransform with those points. We finally used imwarp to get the new warped frame/mask for the next iteration, as well as the new windows. This is repeated between each pair of frames. The end result is decent for a lot of images, but it struggles with some of the images like the weightlifter that had many unrelated features that looked the same. Optical Flow was done by finding the optical flow for each window. This is then used to find the average total movement. Below is an example of feature matching between images:



Updating Local Classifiers:

The local classifiers and models are updated in the updateModels.m file.

Updating the Color Model:

For updating our Gaussian Mixture Models we include the pixels that are within the foreground if they have a shape confidence of at least 0.75 and are in the background if they have a shape confidence of at most 0.25. If the foreground or background array of pixels has less than 3 pixels, we will use the previous Gaussian Mixture Model. We then train a new Foreground Gaussian Mixture Model and Background Gaussian Mixture Model based on the foreground and background pixels in the local window. After using a history-based classifier, we check if the new foreground classifier classifies less foreground pixels than the previous foreground classifier. If the old classifier finds less foreground pixels than the history-based classifier, we keep the old color confidence value and the local window probability mask from the old classifier. If the old classifier determines that there are more foreground pixels than the history-based classifier, we update the color confidence value according to the new weight from the new classifier.

Updating the Shape Model:

The shape confidence function is updated for each local window based on the formula below:

$$f_s(x) = 1 - exp(-d^2(x)/\sigma_s^2)$$

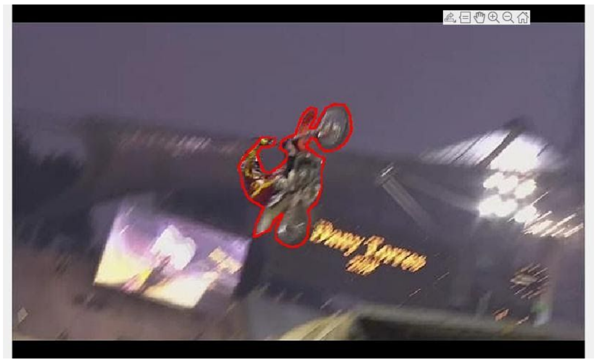
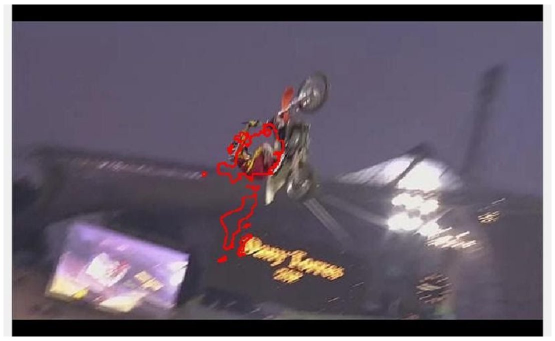
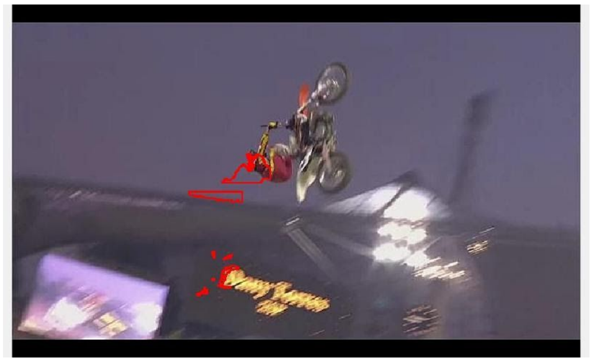
Combining the Shape and Color Models:

To combine both the shape and color models, we followed equation 5 from the paper for each local window. From the equation, it seems that we either weight heavily on the transformed mask or the classifier depending on if the shape model or color model is more confident respectively. The equation that was implemented in our code is on line 121 of updateModels.m.

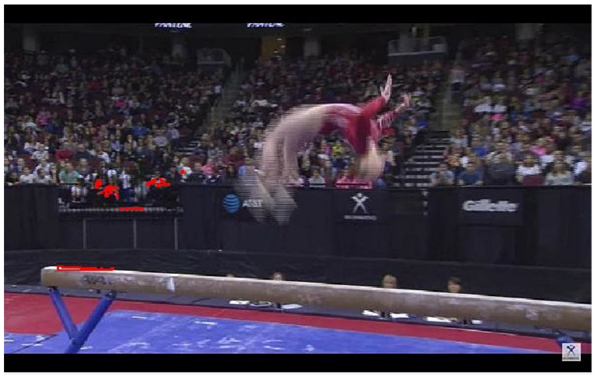
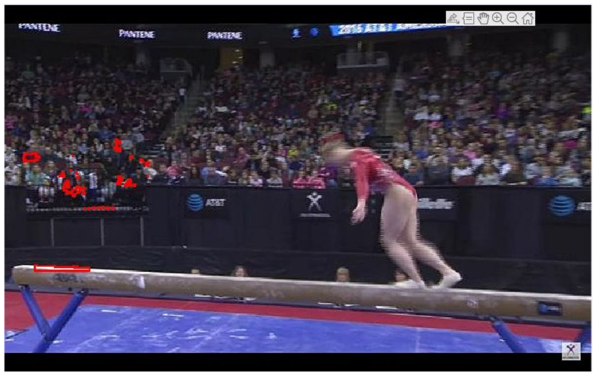
Updating the Foreground Mask:

The last thing we do in updateModels is to merge the local windows together and use their foreground probabilities to make a final foreground mask for each frame in the program. We use equation 6 in the Snapcut paper and use a threshold of ProbMaskThreshold.

Problems:

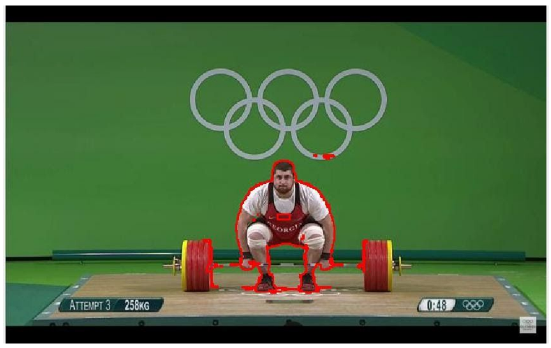




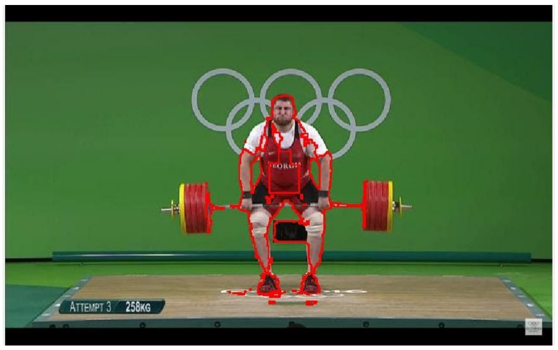
On frame set 2, the mask outline holds shape well on the bike but loses it once the outline starts moving below the bike as the camera moves up.

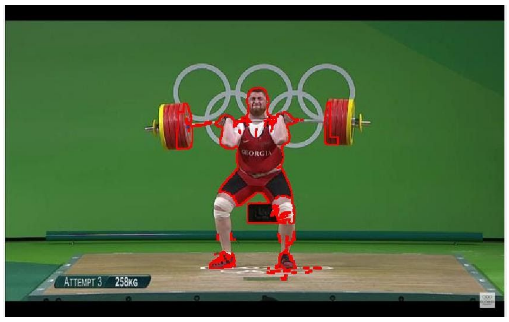
An example late into frame set 3 shows what's left of the mask outline moving left as the camera moves right. Notice how it stays in the relatively same place.

We had the most problems with frame sets 2 and 3. We noticed the mask would hold shape ok for the first several frames very well, but then the mask would linearly drift away from the foreground, after which it would then lose its shape as its color models become skewed. What we realized was that in these frame sets, the camera moves and the entire background moves along with the foreground object. Because of this, we believe the issue with this is our optical flow estimation. Since our code worked alright for the other sets, we believe the flow calculation was correct, but perhaps our homography estimation was off.



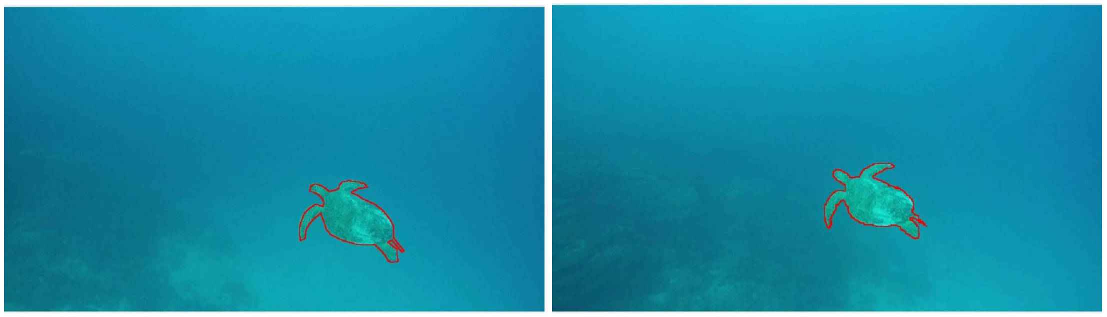


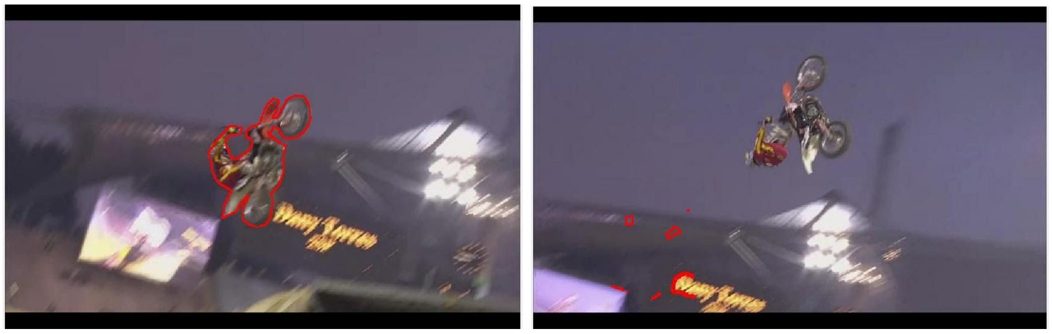
Sometimes when we would test different inputs, a recurring problem that we would have is to change the parameters to be adjusted to certain input frames to fit the entire object we are trying to mask. This can be seen by the two screenshots above where the body of the weightlifter is not being included in the mask. The screenshot below also highlights the problem of false foreground in a stationary background where the black box behind the weightlifter is classified as the foreground.

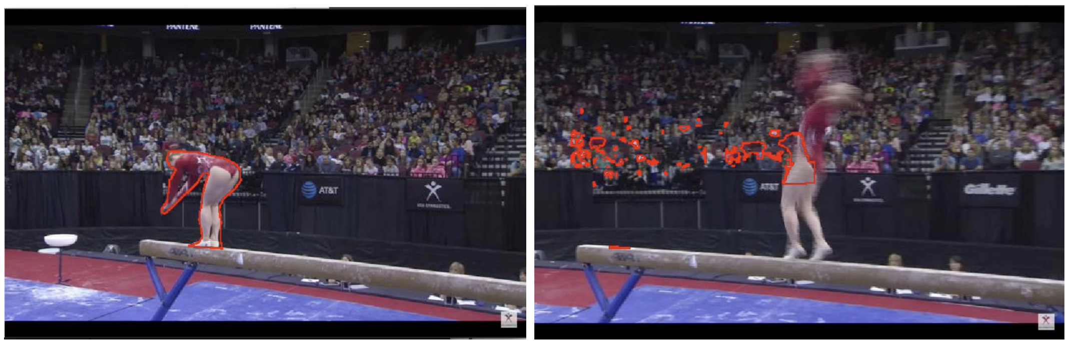


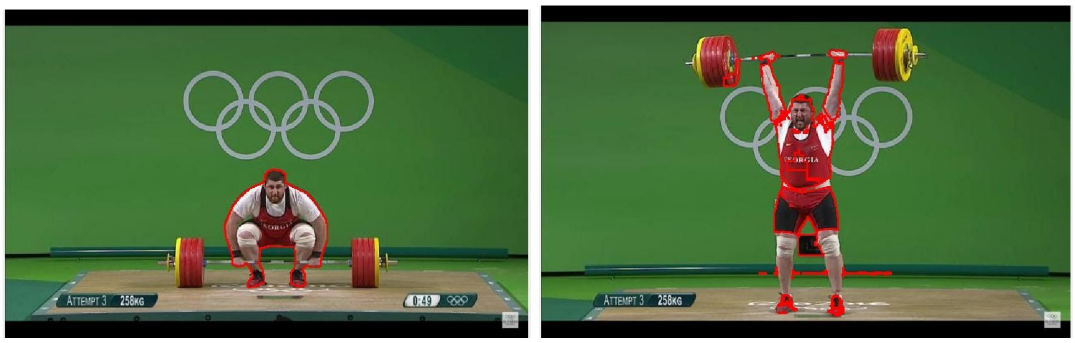
Start-To-Finish Results:

(Videos for all results are in the main directory folder)

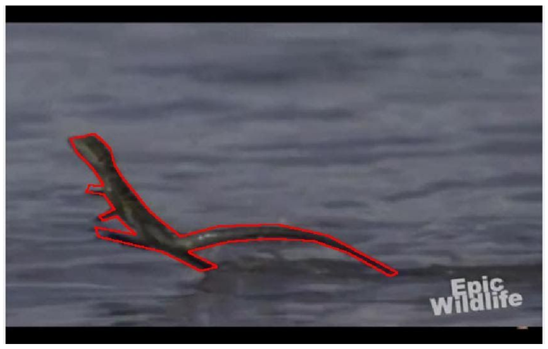

BoundaryWidth = 3, WindowWidth = 80, ProbMaskThreshold = 0.4, NumWindows = 40

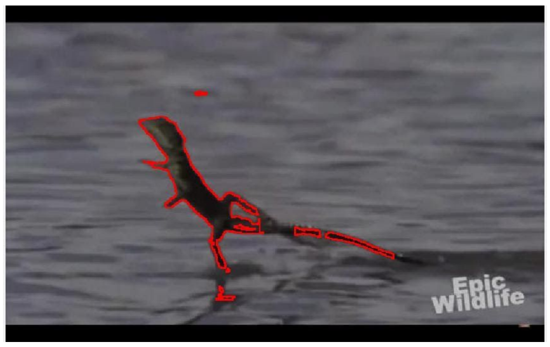

BoundaryWidth = 3, WindowWidth = 90, ProbMaskThreshold = 0.35, NumWindows = 40


BoundaryWidth = 1, WindowWidth = 80, ProbMaskThreshold = 0.35, NumWindows = 30



BoundaryWidth = 3, WindowWidth = 90, ProbMaskThreshold = 0.35, NumWindows = 40





BoundarySize = 2, WindowWidth = 70, ProbMaskThreshold = 0.4, NumWindows = 40