CS6640 A4

Jake Bergquist

10/24/2018

1) To begin with I used the video1.avi in the A4 directory. To test different potential segmentation techniques for the cars I first looked into using background subtracted images. I ran my CS6640_bakcground function to extract the video background which is shown in **Figure 1.1**. I also used a grayscale transform un my testing shown in **Figure 1.2**. I then extracted Frame 1 from the video (shown in **Figure 2.1** and **Figure 2.2** as rgb and grayscale respectively). When comparing the background subtract (background – frame) images for rgb (**Figure 3**) and grayscale (**Figure 4**) I saw no difference. In fact **Figure 3** – **Figure 4** results in all zeros. The next step was to threshold the difference image (**Figure 5**) in which we can see there are two areas corresponding to the two cars moving in this image. Unfortunatly this difference images do not give a complete picture of movement as there are parts of the moving things that are similar enough to the background to pass as background. To assist with this the final algorithm also incorporates a frame to frame difference where the previous frame is subtracted from the current frame. This difference was then also thresholded and passed through the following filtering steps.

Figure 1.1: Background Image

Figure 1.2: Grayscale Background Image

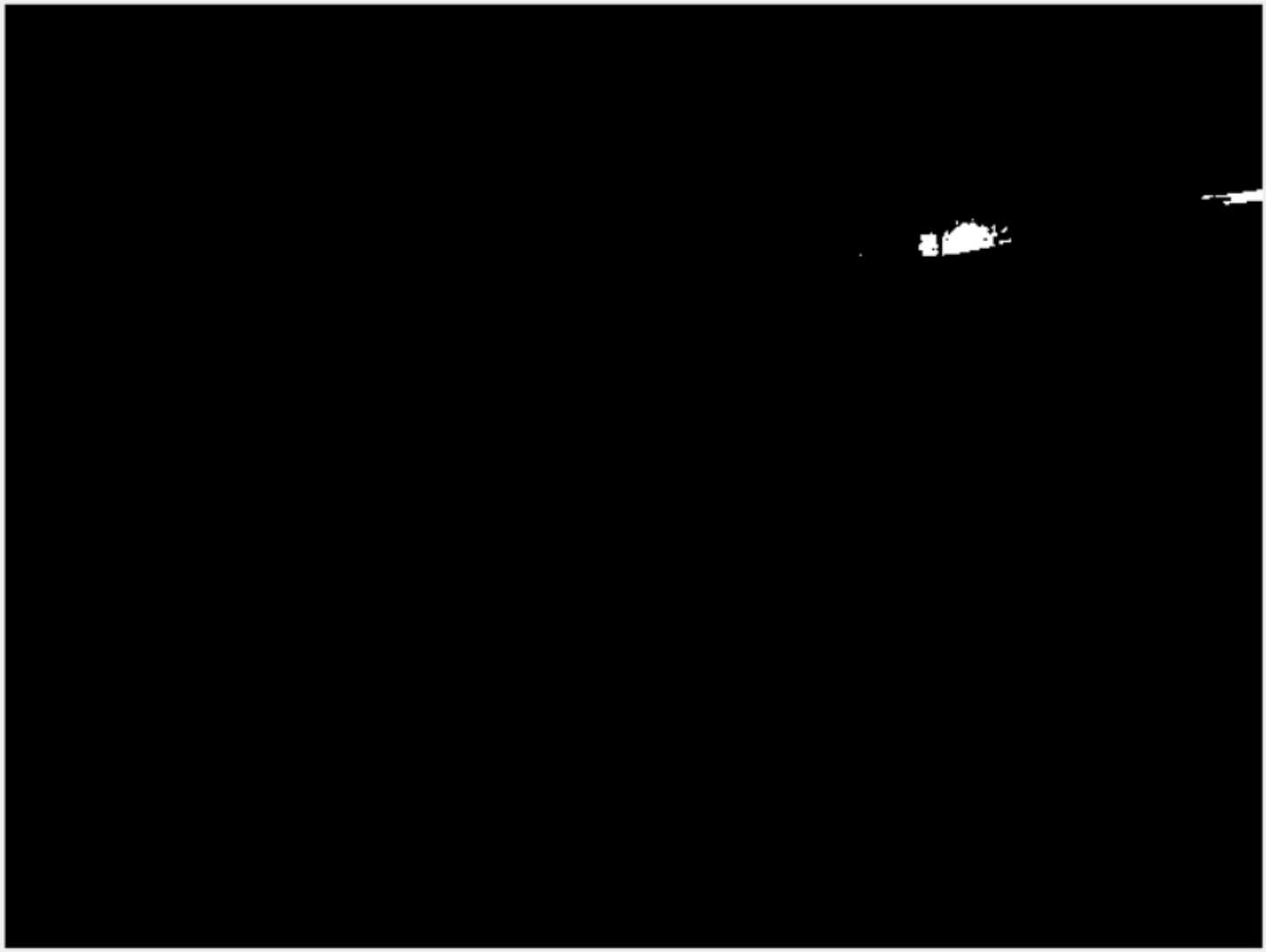
Figure 2.1: Frame 1

Figure 2.2: Grayscale Frame 1

Figure 3: Difference Image

Figure 4: Gray Difference Image

Figure 5: Threshold of Diff Image



Next to filter the the difference images, before thresholding and binaraizing, the difference image was median filtered, then 'filled out'. To fill out the image a 5x5 window was passed over a zero padded copy of the image and pixel was set to the sum of the 5 x 5 window centered over that pixel. This resulted in an increase in signal in areas where there was difference, filling in holes and amplifying the signal there. (**Figure 6**) It also had the side effect of aplifying low level differences but these were omitted thanks to a threshold of max intensity – 4*standard deviation. The then binirized image was eroded to get rid of small noise from tiny movements, then dialiated, then dialated again to connect close regions, the eroded back to the normal level. Then holes were filled using bwmorph and lone small groups were deleted using bwmorph. This binary mask (**Figure 7**) was used to identify moving vehicles. Such a mask was made for both the background subtract and the frame to frame differences. For each frame the background subtract and frame difference masks were combined to get the final oving objects mask. In order to hilight the moving objects in the video the mask was inverted, dialated by 2, then intersected with the original mask. This resulted in an outline around the moving objects which was colored blue using the custom mask() function. mask() was used to generate the frames for the movie using the original frames and the object outline mask. **Figures 8, 9, 10** show the outlines for the moving objects in Frames 1,2 and 12 respectivly. Of interest in frame 12 we can see that even the person walking on the back sidewalk was detected and outlined. In each case the algorithm does a decent job identifying the moving objects, however it occasionally catches things like the shadows of the cars/clouds or sometimes regions just behind where the cars were. Additionally when two moving objects intersect they become one, and when a moving object passes behind a large stationary object (like the trees in frame 12) the moving object can get cut up a little. However the overall centoids of

the moving objects look by eye to be mostly correct and thus useful for identifying and tracking the moving objects across the movies.

Figure 6: Filtered and filled out, Frame 1

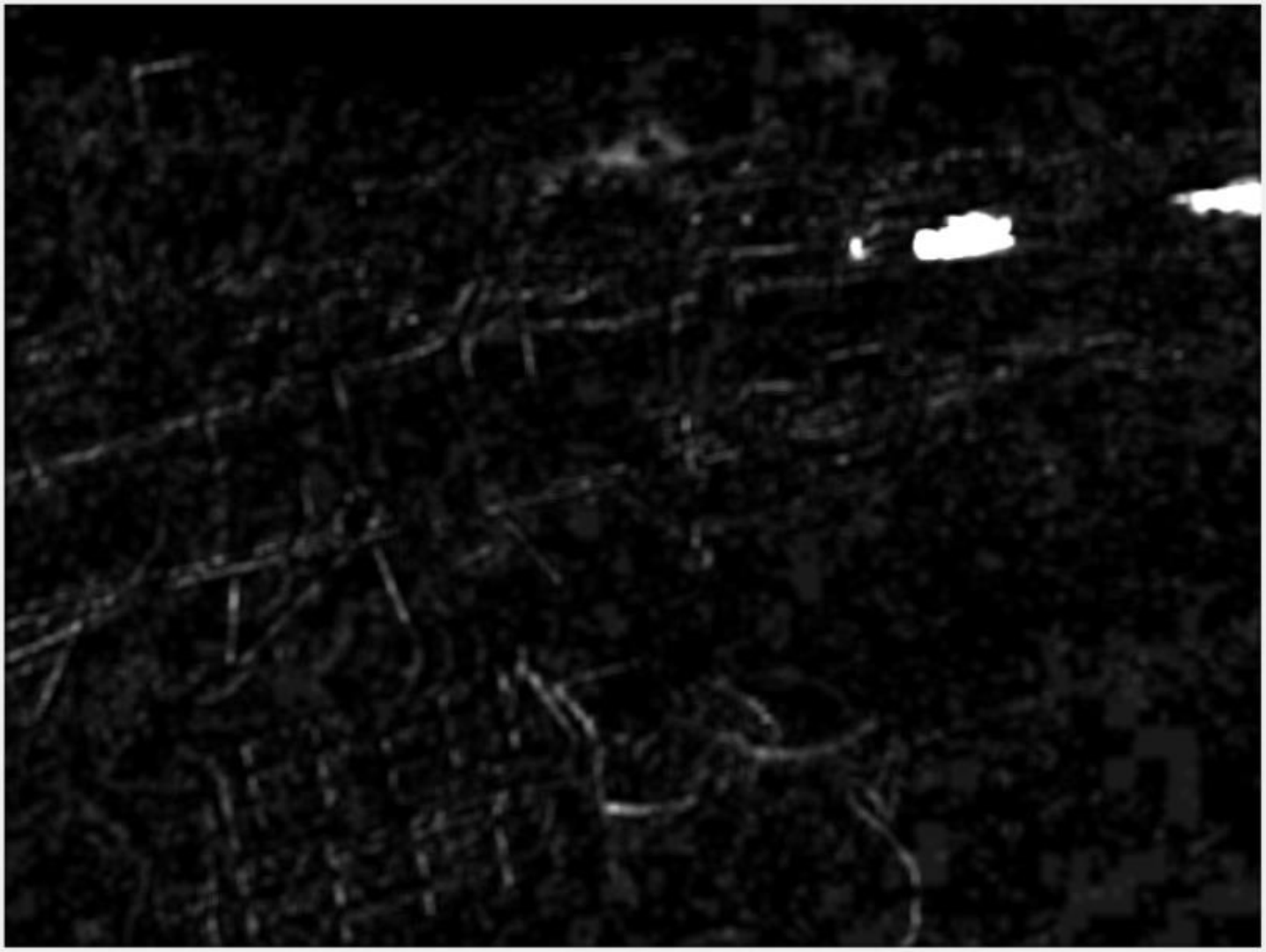


Figure 7: Processed, frame 1

Figure 8: Outlined moving things frame 1

Figure 9: Outlined moving things frame 2

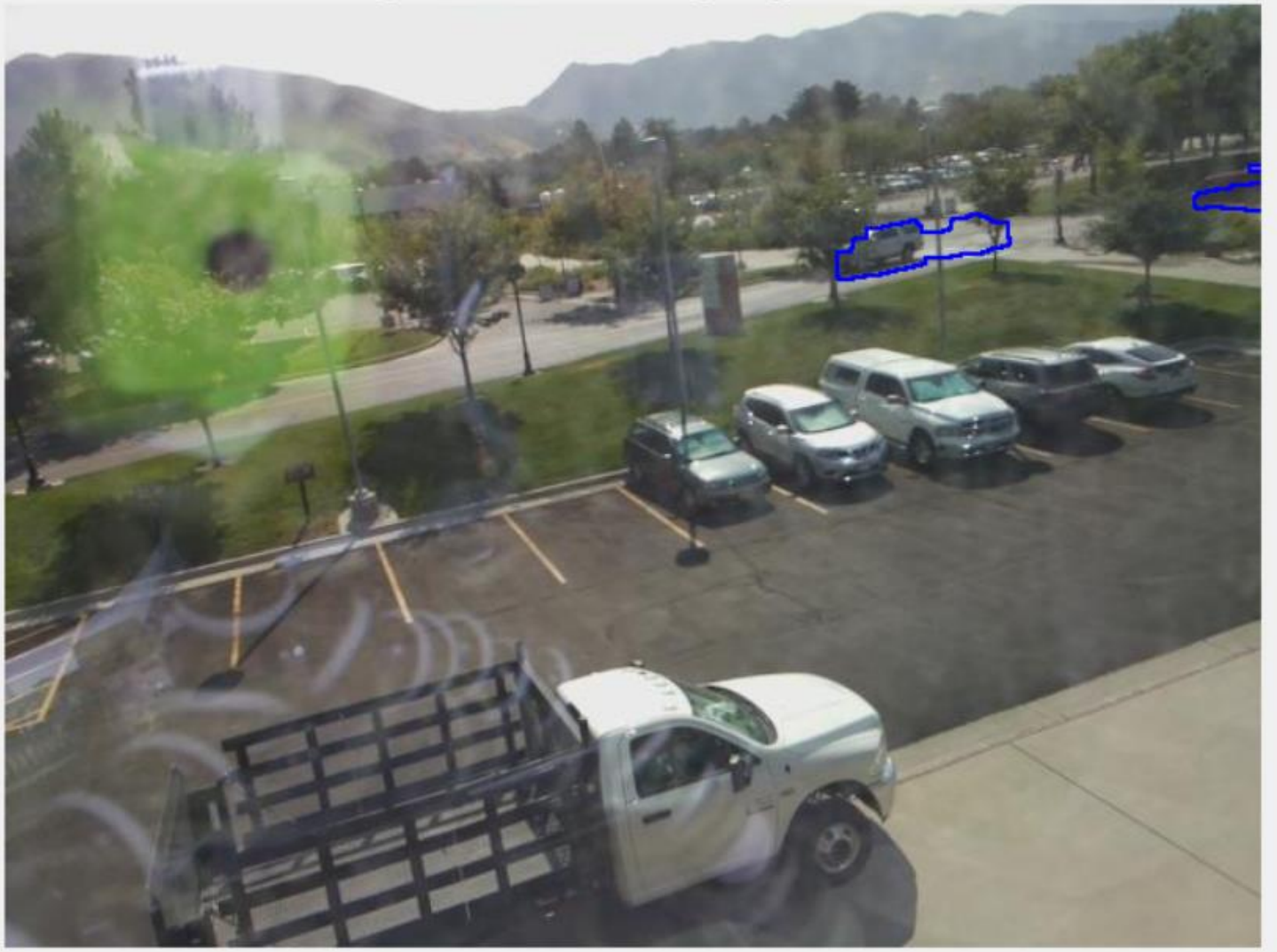
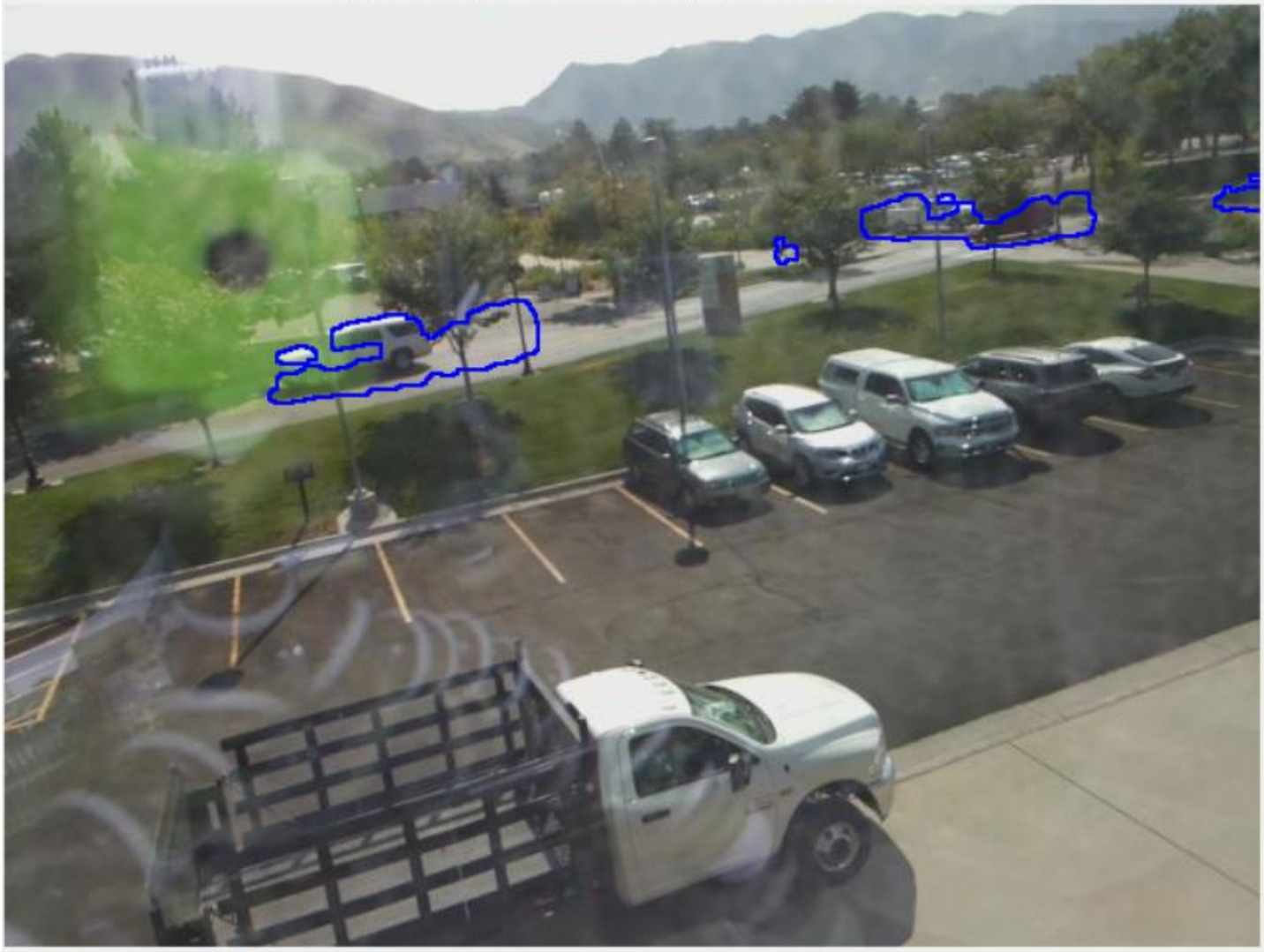


Figure 10: Outlined moving things frame 12



2) For the object_data function I used the region props function to attain most fo the statistics. I first used the same algorithm as before to identify the moving things binary mask for each frame. I then used bwconnectedcomp to identify the connected objects and reparate them. I then fed them into regionprops to get the centriod, bounding box, number of pixels. The bounding box was used to get the indicies of th etop left and bottom right corner column and rows. The centroid was used to get the mean column and row. The pixel indicies provided by connected component were used to get the R G and B values, and the median functuion was used to get the median. **Table 1** shows the object data for a few of the frames. **Table 2** shows the statistics for the objects in frame 12.

Table 1

Frame	NumObjects
Frame1	3
Frame2	3
Frame12	5

Table 2

Object	NumPixels	redMedian	greenMedian	blueMedian	columnMean	rowMean
Obj1	4128	119	143	143	177	206
Obj2	149	126	125	125	126	398
Obj3	2465	87	91	91	110	497
Obj4	294	72	82	82	99	629
Obj5	30	67	74	74	89	637