Assignment 5

Taylor Foxhall
tfoxhal1@binghamton.edu

December 1, 2015

1 Part A

Here we apply a couple of different nearest neighbor classifier techniques, using the top half of the image as training data and the bottom half as testing data. In the first example to produce M1, we just compute the average of each training vector and classify it based on the buckets $[0, 124]$, $[125, 174]$, $[175, 255]$

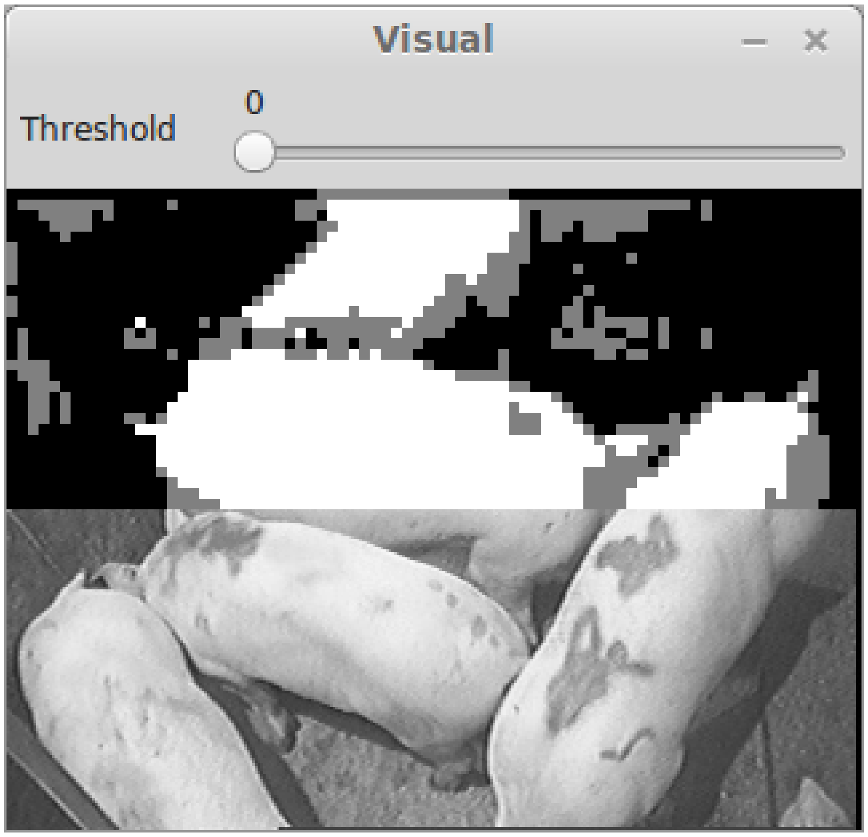


Figure 1: M1, training data manually clustered based on specifications

Using those divisions, we can go through all our testing vectors and find their nearest neighbor using the Euclidian distance formula, and assign the corresponding class.

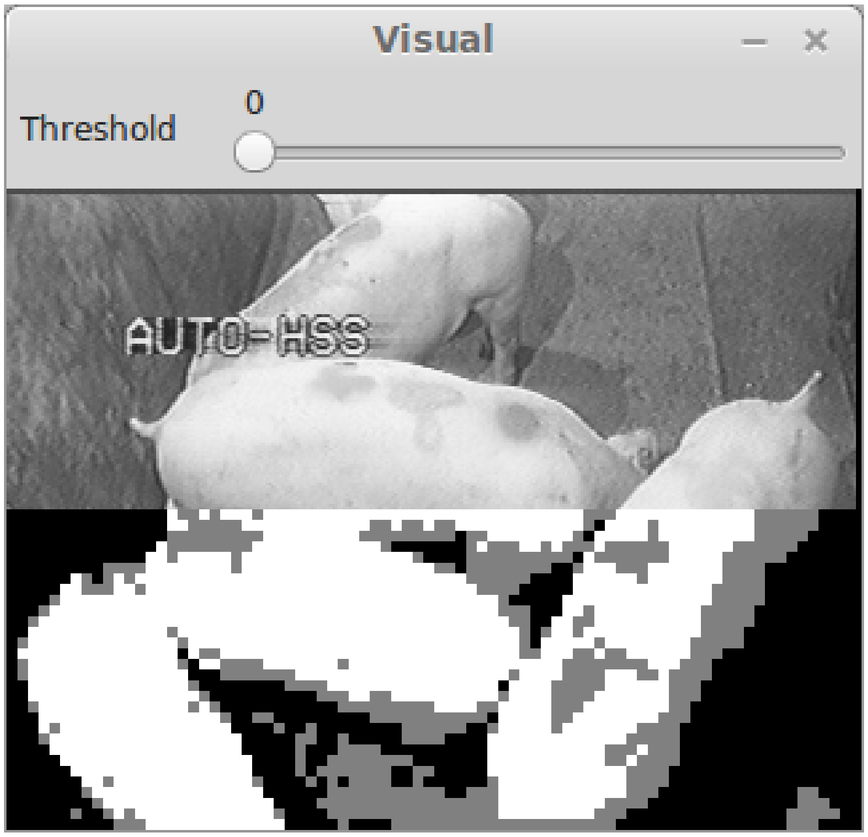


Figure 2: N1, testing data classified by nearest neighbor vector

This give us something a little ugly, and also this result takes a while to produce. Another way we could estimate this image is to replace the testing data with the nearest neighbor in the training data.

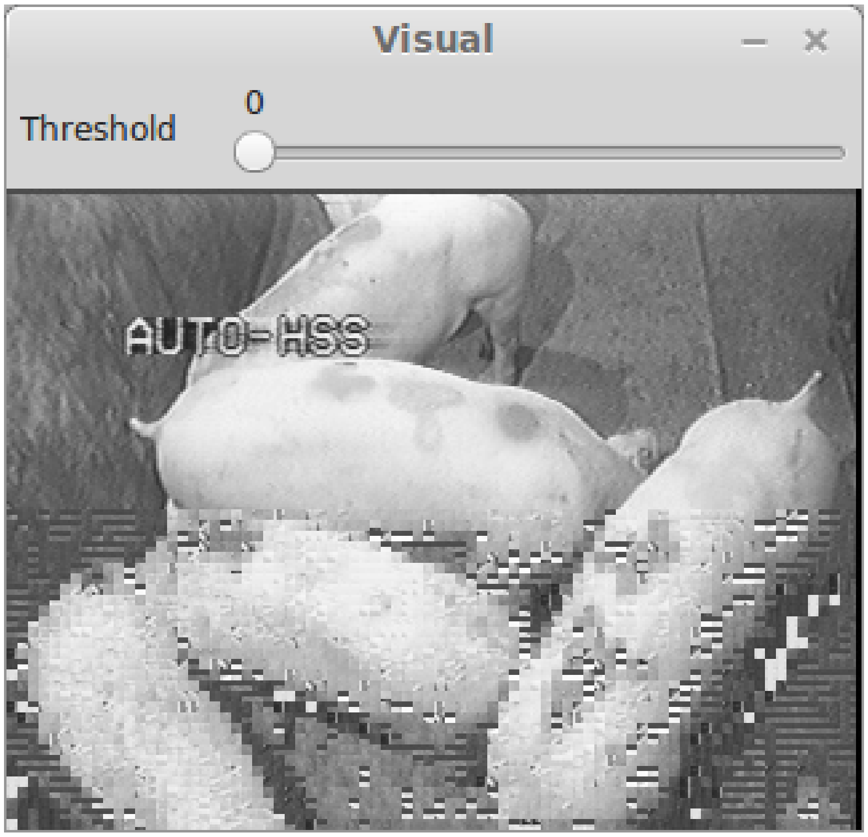


Figure 3: N2, testing data replaced with nearest neighbor training vector

However this gives us results that while quite accurate, are also visually jarring because of the sharp differences from neighboring vectors. We can smooth this up a little by replacing the vector with its average value, getting a smoother, blurred look.

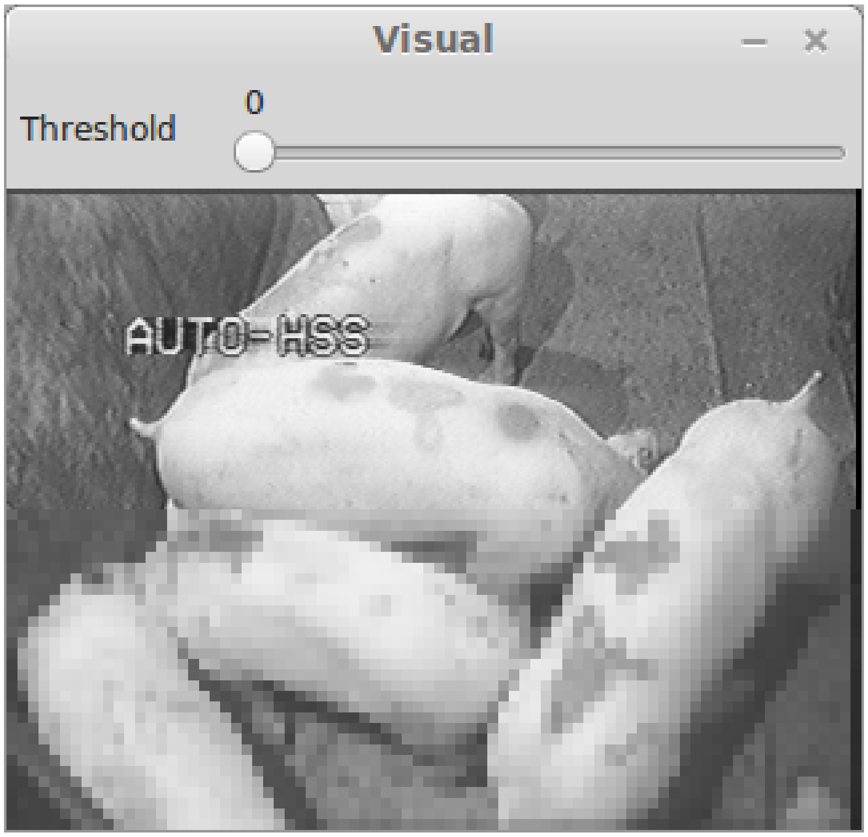


Figure 4: N3, testing data replaced with nearest neighbor average

Even yet still we can try to improve on N1's color accuracy by changing the colors to the average color of all the vectors in each corresponding class.

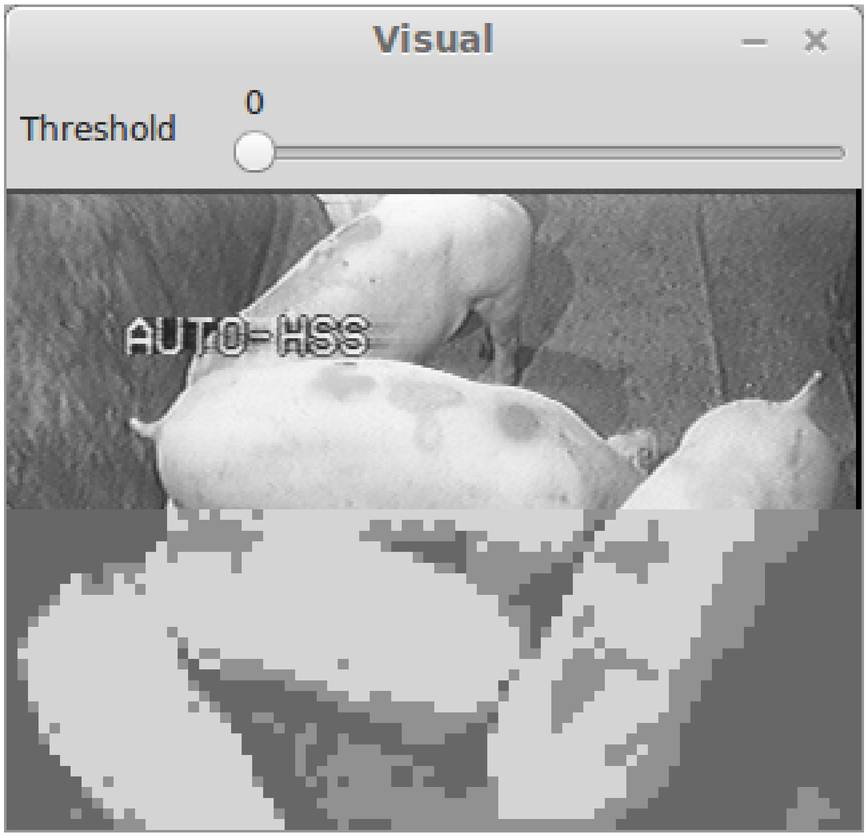


Figure 5: N4, testing data replaced with nearest neighbor class average

Assessing N1 a bit more, we can manually replace the testing vectors like we did with M1 to prodcue T1 and compute the error value between our nearest neighbor predicition N1 and the acutal classification T1.

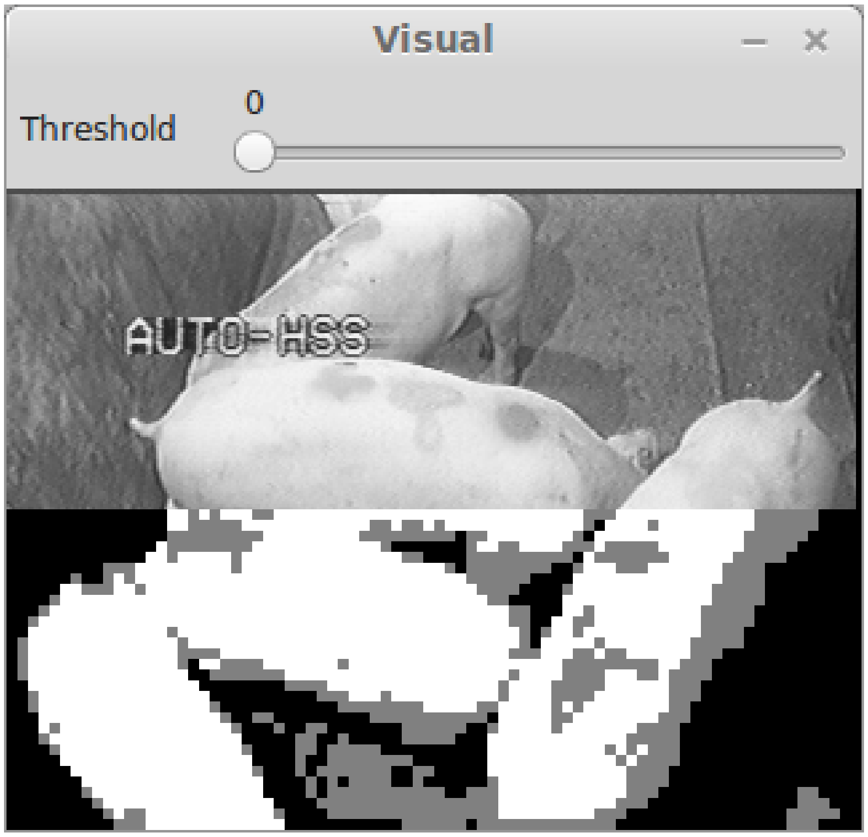
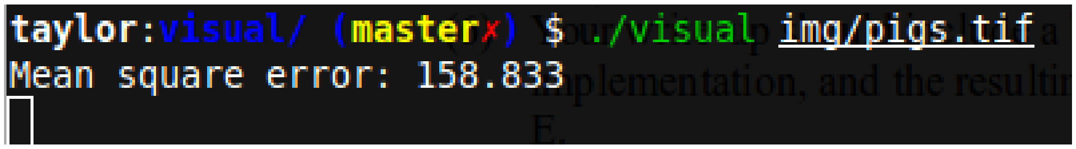


Figure 6: T1, testing data manually clustered based on specifications



```
taylor:visual/ (master✗) $ ./visual img/pigs.tif
Mean square error: 158.833
```

Figure 7: Mean squared error between N1 and T1

We could also try a k-means clustering algorithm to alternatively group the classes. My result is slightly miscolored, probably because of the ordering of the initially sampled centers, but the differences between the three classes are still apparent.

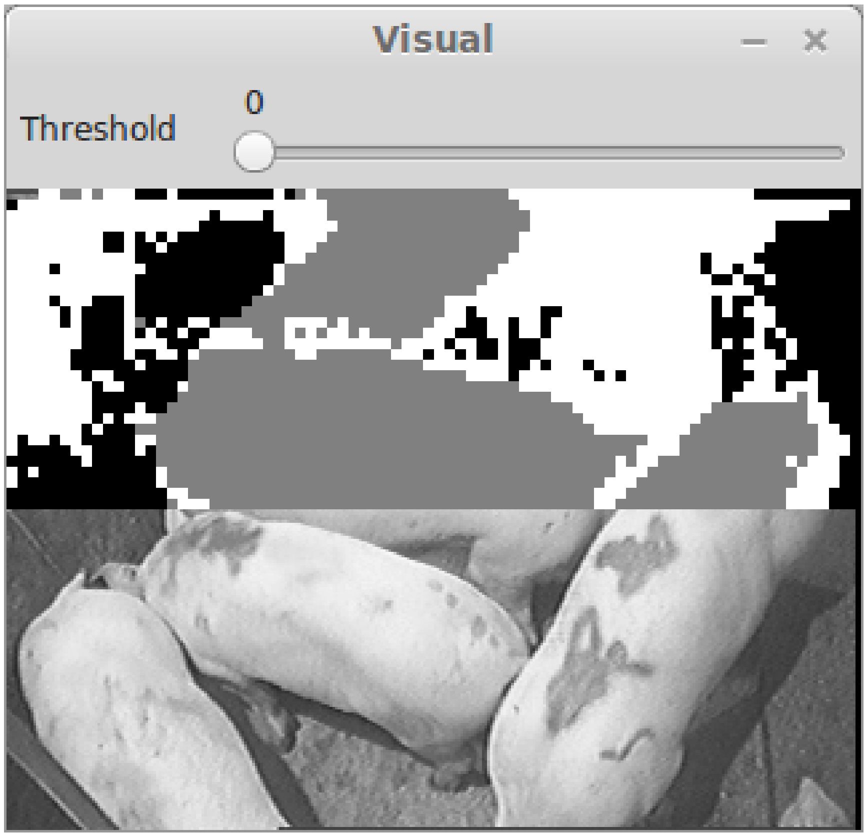


Figure 8: K1, training data separated by k-means classfication

2 Part B

Here we examine some motion detection techniques. If we simply take the difference between two frames, we get the differences overlayed but we can't really tell what's moving where. We can compensate the difference in motion between the two frames by using an idea similar to nearest neighbor classifiers to find the shortest Euclidian distance between 8×8 segments of the frames. The nearest neigbhor is then subtracted from the original image and we get a better idea of the motion vectors.

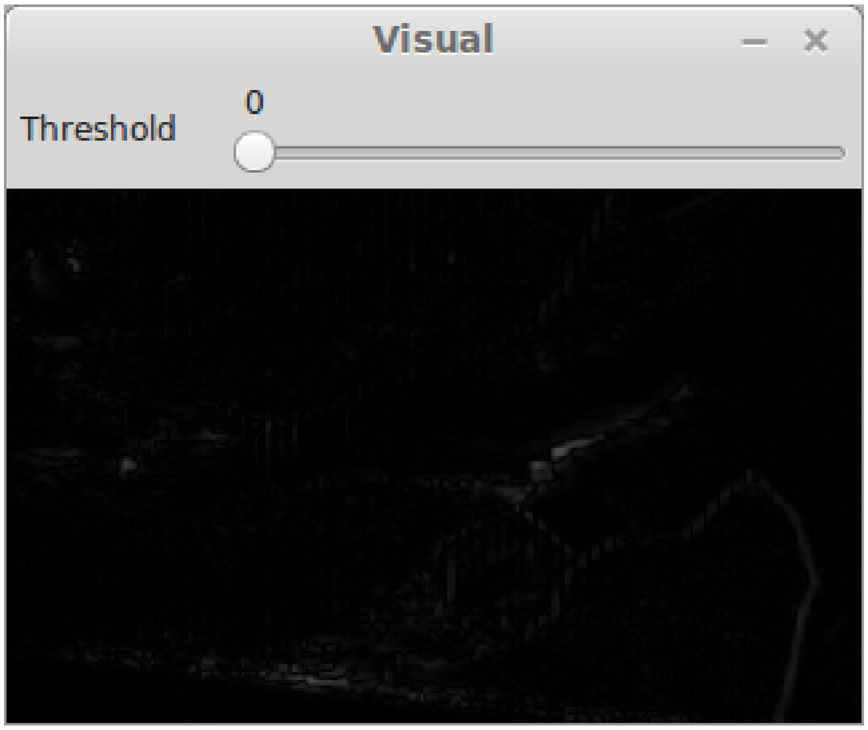


Figure 9: The direct difference between the first two frame sets

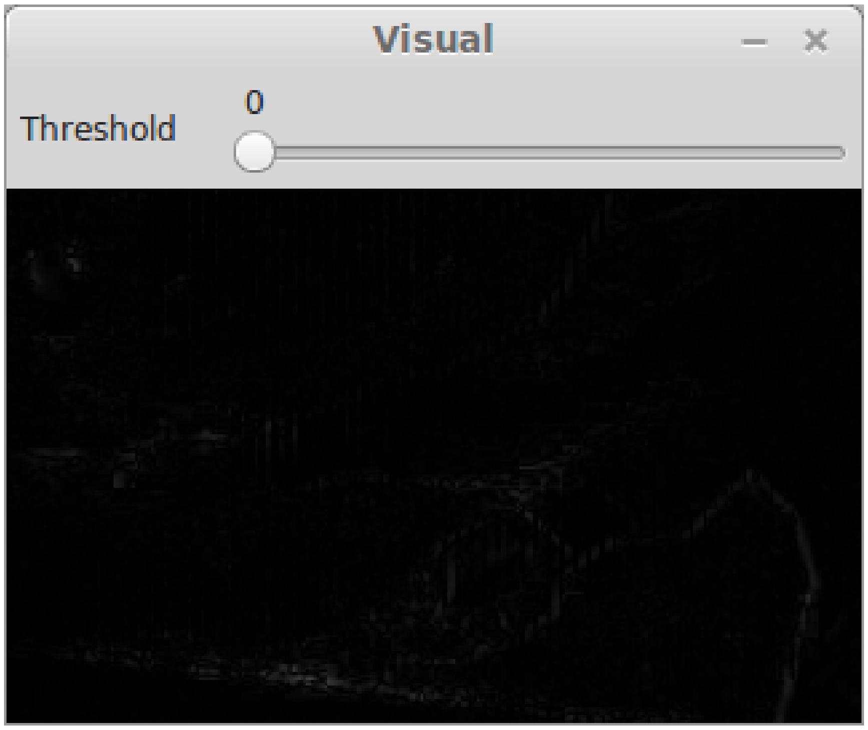


Figure 10: Compensated difference between the first two frame sets

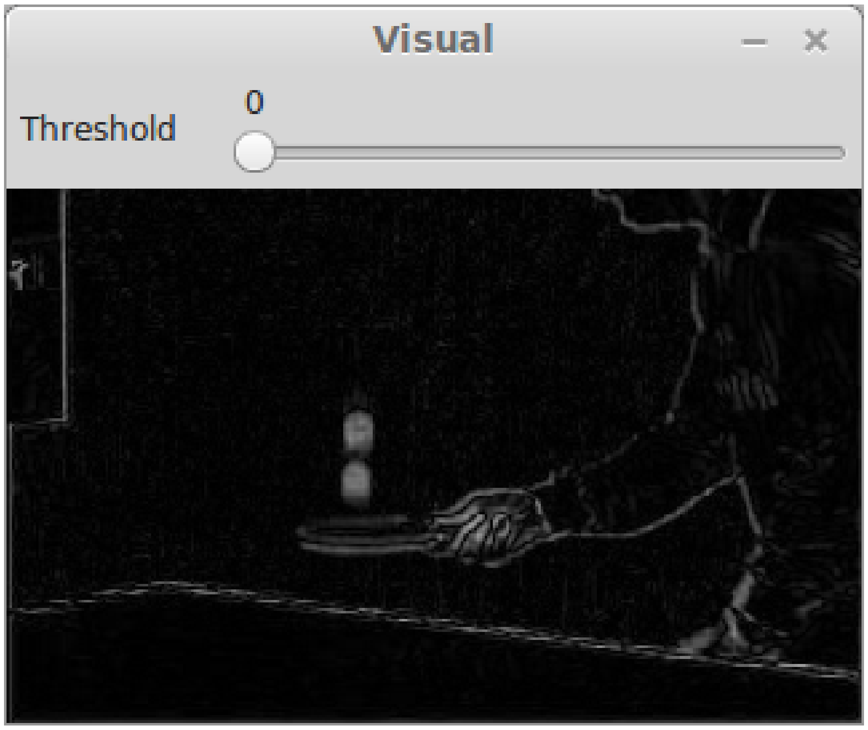


Figure 11: Direct difference between the second two frame sets

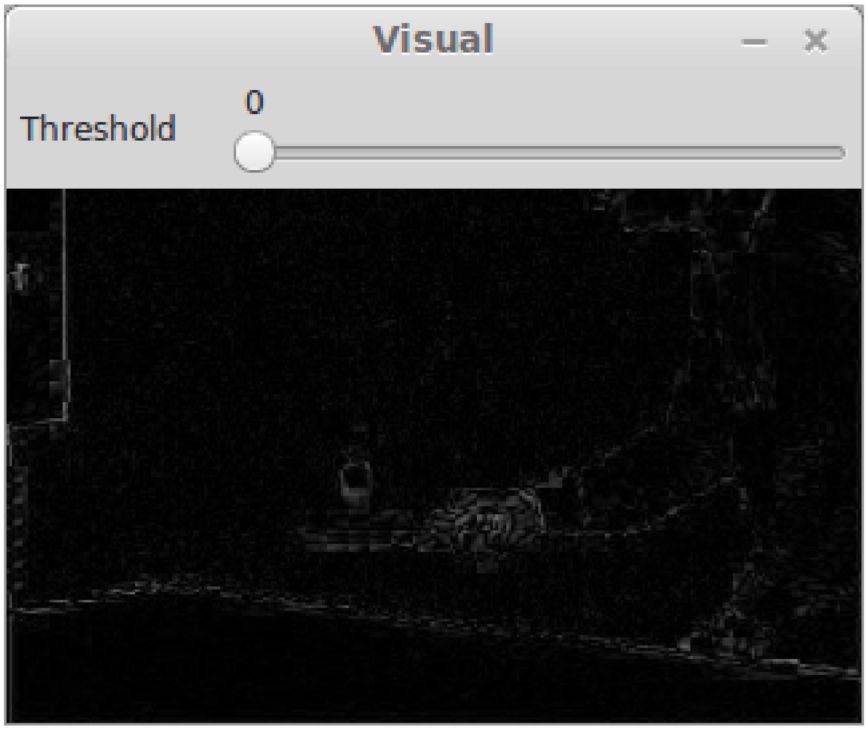


Figure 12: Compensated difference between the second two frame sets

The compensated difference in Figure 12 is especially distinct, as in Figure 11 you can clearly make out 2 distinct balls (their “shadows”), but the difference disappears in figure 12 and you can see the motion vectors better.

3 References

1. OpenCV 3.0 Documentation — http://docs.opencv.org/3.0.0/