Colin Eason, Haiming Tan, Matt Morel

CS510, Spring 2016

**Video Action Recognition by Comparing Subspace Manifolds**

**Definition**

Action recognition in video streams is still an active research area in computer vision. Nonetheless, we wanted our system to at a minimum distinguish between moving and still actions. Furthermore, we wanted our system to handle distinguishing actions when temporal order matters such as the difference between dropping and picking up an object. To accomplish this task, we had to completely switch data domains from traffic to pedestrian. This introduced many challenges that will be discussed in later sections, but as a consequence we decided to focus on only people actions and remove the complexity of object classification. For evaluation, we decided to create ground truth from our video by the semi-automated VATIC approach from the previous assignment and enhance it by computing the classification accuracy and confusion matrix to illustrate general and individual performance in different approaches.

**Approach**

We first considered using Laptev STIPs [1] with a bag of words, but decided against it primarily because we had exposure to that paradigm in the previous assignment and wanted to try something novel. Therefore, we decided to attempt the video cube comparison method presented in class. To create the comparison set, we used the tracking system developed in our previous assignment based on MOSSE [2] correlation and manually selected when to output the previous N frames (45 in practice) as a tracklet. To handle translation of the object and to preserve velocity information, we saved the entire previous N frames and calculate the area which will completely cover the tracked object for that time as shown in figure 1. Each training snippet is resized to 75x100 before computing the basis vector. We sorted each action snippet into five action verb categories: walk, run, wave, drop, and pickup. During the training phase, our system reads in the collection of action snippets and computes the subspace basis for each with OpenCV PCA routines.

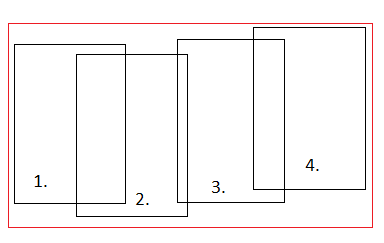


Figure 1. Collecting frames into a tracklet

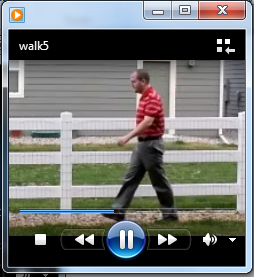


Figure 2. action snippet derived from tracklet

During runtime, our system generates a sliding window of the tracklet (using the same method described above) every five input frames and runs a comparison between every training snippet using SVD of the multiplied subspace basis vectors. In addition, we store the average velocity of each training sample and compare it to the sliding window snippet velocity. If the sliding window velocity is more than 15% above or below the average action, we don’t even compute the subspace comparison. This will be discussed in the results, but it turns out to be a significant optimization both in terms of speed and accuracy.

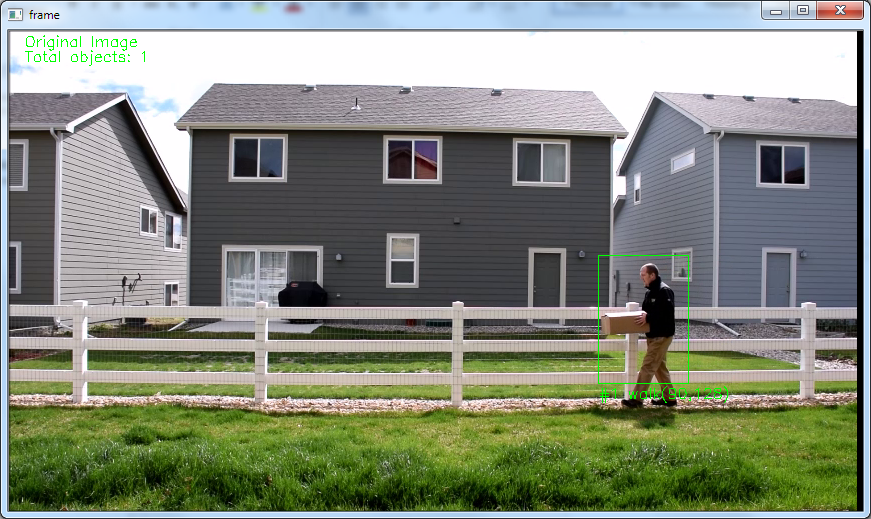


Figure 3. successful snippet comparison



Figure 4. successful transition to new action

The first and biggest challenge we ran into was that our tracker has a very difficult time seeding new tracks on people. Cars were relatively simple since the center of each movement blob closely follows the center of mass and there is not generally much motion in that area. People, on the other hand, ended up with the center of the track right below and in front of the waist where there is lots of motion from hands and legs moving. This resulted in poor tracking behavior until we added some logic to estimate where the torso is and seed each track based on that subset of the motion blob. For some tracks, we had to just seed them by hand in the interest of time.

**Evaluation**

We decided to keep the same semi-automated evaluation system from PA3, but made a few tweaks to support this assignment. The first of these were to simplify the evaluation by only considering action labels per frame instead of the tracked bounding boxes. Second, we did not consider results before capturing a full video cube (N frames) because the amount of false negatives is a large percentage of our short test videos. In addition, we thought it would be interesting to compare our system performance when some key algorithm tweaks were implemented. First off, we tried applying a Sobel X/Y edge mask to each frame for both training and testing samples. Our hypothesis was that edge images would be less sensitive to changes in background and clothing. Next, we tested our system with the consideration of tracklet velocity completely disabled, expecting to see a noticeable drop in performance distinguishing between walk versus run.

**Results**

Below are the raw and processed results of our system run on six different test videos starting with an overall picture of different algorithm performance followed by the breakout of each algorithm and action type.

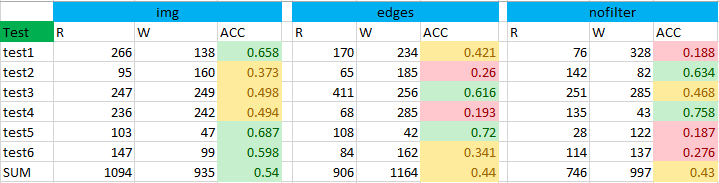


Table 1. Accuracy results between three key algorithm tweaks

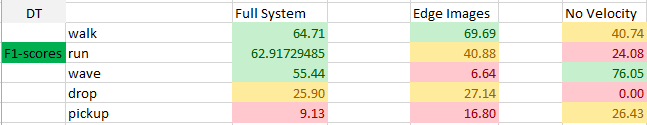


Table 2. Combined f-score comparison between algorithms

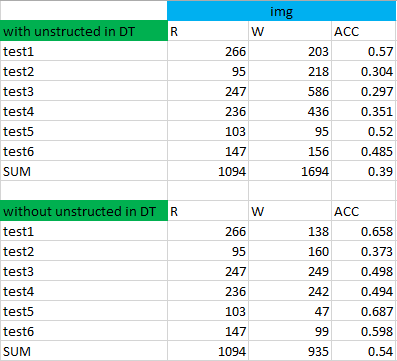


Table 3. Accuracy result for full system test

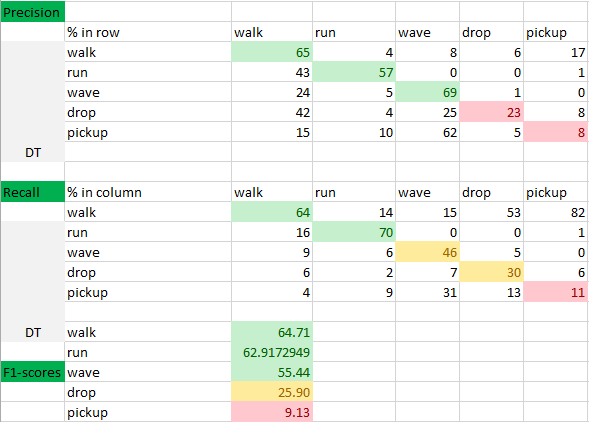


Table 4. Confusion matrix for full system test

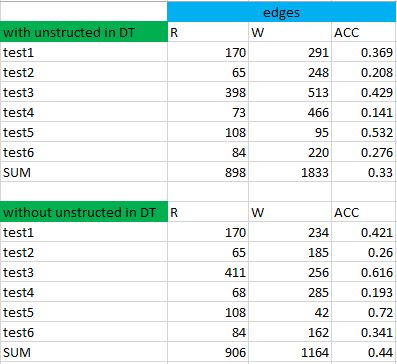


Table 5. Accuracy result for edge images test

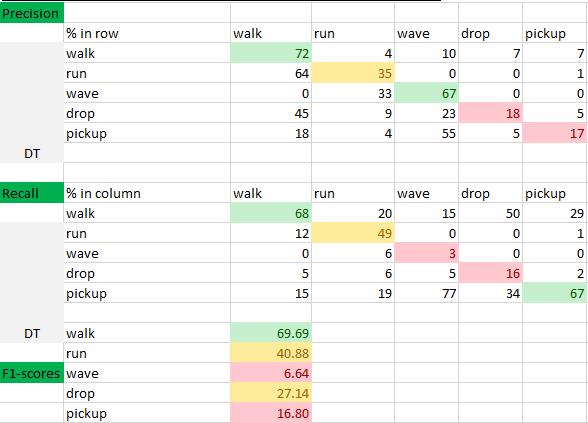


Table 6. Confusion matrix for edge images test

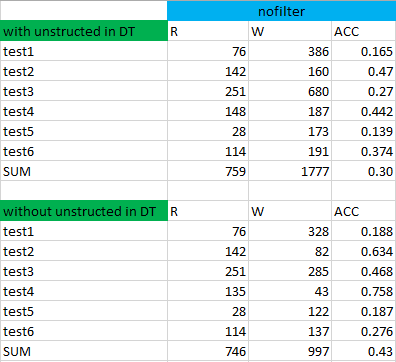


Table 7. Accuracy result for velocity filtering test



Table 8. Confusion matrix for velocity filtering test

**Analysis**

When we just look at the complete system with no tweaks, the state of affairs is actually decent. Table 1 shows that our full system did indeed have the best average accuracy compared to the modified algorithms. Additionally, table 2 shows that even though the tweaked algorithms sometimes made gains in one action, they cut other actions f-score dramatically. We didn’t set a formal f-score goal for this assignment, but were pleased that our system did perform better than random for the majority of actions. Obviously, teasing out very similar actions drop versus pickup was more difficult. While not represented in the raw results, anecdotally the majority of our mismatches came from the transition between actions, which is to be expected. Comparing our system using plain spatial images to using Sobel edge images, the results were actually a little bit disappointing. We thought for sure it would perform at least as well as using spatial images, but overall it hurt accuracy more than it helped. Perhaps the most confounding aspect is that actions were not uniformly affected, for which we don’t have enough understanding to offer an explanation. One hypothesis we have about the effect of edge images is that stripping away so much information leaves the comparison much more susceptible to noise. Lastly and least surprisingly, our system performed much worse when velocity data was ignored both in terms of accuracy and execution speed.

**Conclusion**

Overall, we were pleased that our system could actually function given the task complexity. That being said, there were several improvement areas we discussed but simply did not have time to implement. The first of these would be some sort of clustering algorithm to limit the total number of comparisons. While we did not maintain real time processing in this assignment, our system was actually close and clustering might have been enough to push it into real time. Second, we really liked the idea of temporal smoothing with a Hidden Markov Model to eliminate some of the false positive action responses our system generates. Since one of the major problem areas for our system was in the transition between actions, the HMM would likely greatly improve our ability to smoothly transition actions without getting lost. Everything considered, this assignment was challenging yet rewarding.

**References**

[1] I. Laptev and T. Lindeberg; ["Space-Time Interest Points"](http://www.irisa.fr/vista/Papers/2003_iccv_laptev.pdf) (2003), in Proc. ICCV'03, Nice, France, pp.I:432-439.

[2] David S. Bolme et al. "Visual Object Tracking using Adaptive Correlation Filters"