**I400/B490 Intro to Computer Vision - Programming assignment #9 - Final**

: This assignment heavily relies on lecture10_1_motion.pptx and lecture11_1_activity.pptx. For this assignment, we will need to use the `opencv` library for optical flow extraction and image reading. Optical flow extraction can be done by following the textbook, pages 216-217 (265-266 in pdf copies).

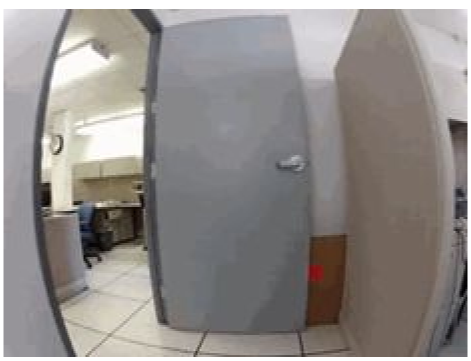
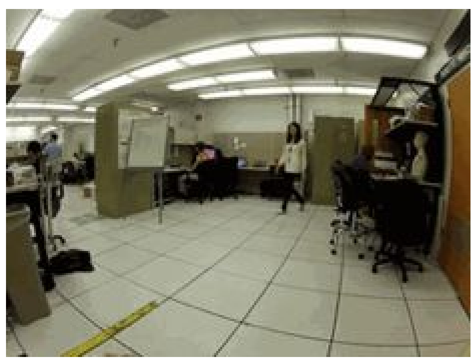
***Problem 1: Optical flows [10%]***

0. Download the two zips files containing video frames from CANVAS: jpl_thomas.zip and dog.zip

1. Use `cv2.calcOpticalFlowFarneback` dense optical flow extractor. Try to visualize the extracted optical flows. Use `draw_flow` function in the textbook. Save the results (image frame + optical flows) as flow1.zip and flow2.zip.

***Problem 2: Activity classification using HOG/HOF [60%]***

0. Download the JPL-Interaction dataset from the URL: [url_TBD]. We will use the sets 1-6 for the training and the sets 7-12 for the testing of the classifiers. The video name N_M.avi means that the video belongs to the set N and it contains activity class M.

1. Load each video as a 3-D array (XYT data) using the following code:
import os
from skimage.io import imread
import numpy as np

```
def xyt(sequence_dir):
    fnames = os.listdir(sequence_dir)
    label = int(sequence_dir.split('_')[1])
    frames = sorted(fnames, key = lambda x: int(x.split('_')[1].split('.')[0]))
    return label, np.stack([imread(os.path.join(sequence_dir,f)) for f in frames])
```

\# data now is of shape (num_frames,height,width,3), and label is the integer class label for the activity
`label, data = xyt("jpl/10_1")`

2. Prepare a function `hist = getHOGperFrame(image)`, which extracts histogram of gradients (HOG) descriptor from each image frame. As described in the slide 31 of lecture11_1_activity.pptx, divide the frame into 5-by-5 spatial regions, and then count the number of pixels (in each region) belonging to each of 9 gradient orientation bins. We will use unsigned gradient: i.e., 20 degrees and 200 degrees (180+20 degrees) are treated identically. The 9 orientation bins will be ([0,20], [20,40],…, [160,180]). [20%]

- For each spatial region, iterate through each pixel in the bin, get its gradient orientation, and assign it to one of the 9 bins based on the orientation value. This will generate 9 bins (i.e., an array with 9 values) per spatial region.
- Concatenate all of them, obtaining a total of 225 bins (i.e., an array with 225 values).

3. For each video, compute the average of all per-frame HOG vectors in the video. Create a 2-D array with size 225*<num_videos>, by stacking averaged HOG vectors of all videos. You don't need L1 normalization. [20%]

- Try to visualize the resulting HOG vectors of 1_1 and 2_2. Try to create a grayscale image similar to HOG images of slide 21: draw one line per bin (I.e., 9 lines per spatial bin). The intensity of the line will be proportional to the value of the bin: the value 0 means intensity of 0, and the value 3456 means intensity 255). Save the results as 1_1_hog.jpg and 2_2_hog.jpg

Also maintain the activity IDs of the videos, by generating an array `labels` with size 225*1 (or 1*255). Each value of `labels` should be the activity ID of the corresponding video.

4. Using the videos belonging to sets 1-6 (i.e., training videos), build the k-nearest neighbor (k-NN) classifier (k=3). Do the classification with the testing videos using the k-nearest neighbor classifier. Compare the classification result with the ground truth (i.e., `labels`). Measure classification accuracy: <num_correct>/<num_total_testing_vieo>. Report your classification accuracy using Canvas. [10%]

5. Repeat the steps 1-4, while using histogram of 'optical flow orientations' instead of gradients. [10%]

Use 8 optical orientation bins, instead of 9 bins. That is, you will have an array with 200 values based on the function `hist = getHOFperFrame(frame, next_frame)`. We will use a 'signed' orientation values: ([0,45],…,[315-360]).

- Try to visualize the resulting HOF vectors of 1_1 and 2_2. Try to create a grayscale HOF image similar to HOG images (I.e., 8 lines per spatial bin). The intensity of the line will be proportional to the value of the bin: 0 means intensity 0, and 3456 means intensity 255). Save the results as 1_1_hof.jpg and 2_2_hof.jpg

Similar to HOG, do averaging of HOF per-frame descriptors, and then do k-NN classification. Report your classification accuracy using Canvas.

***Problem 2: Activity classification using HOG/HOF + bag-of-words [30%]***

1. For HOG, repeat the steps 1-4 of the above problem. However, this time, instead of averaging per-frame HOG descriptors, we will use bag-of-words representation to summarize them. [20%]

- Sample 10 frames per video, and do k-means clustering (k=400) to find cluster centers based on those 10*<num_total_videos> samples. You will probably want to use the function `random.sample`.
- For each video, create a histogram. Assign every per-frame HOG to one of the cluster centers. Recall the function `np.bincount` is helpful for constructing histograms.
- Create a histogram (size 400) per video by counting the number of HOGs assigned to each cluster.
- Do k-NN classification similar to the problem #1 and report the classification accuracy using Canvas.

2. Do the above step 1 with HOF instead of HOG. Report the classification accuracy using Canvas. [10%]