Course Project for CS771 at IIT Kanpur

Synopsis

This Project is about classifying the moving objects detected in surveillance videos. For this project we are training and testing our models on surveillance videos taken at the gate of IIT Kanpur. Here we are trying out a bunch of different classifiers and also varying different parameters of the model and the feature array etc to obtain the best result.

Data Files

- All the following data are for the videos taken by security cameras at IIT-Kanpur gates.
- Some samples videos and their label descriptions in the required JSON format
- zip archive of training image set
- zip archive of test image set

Script description & Code Example

Extracting images from videos

- For this first of all you'll need the corresponding label descriptions in the required JSON format. Sample can be found in shared Google drive folder.
- Suppose You have two video files named - datasample1.mov and datasample7.mov which you want to use for training and testing respectively.
- Then you should first use `extract_objects.py` or `extract_filtered_objects.py` for extracting all the images out of both the videos. For this you'll first need label_data/datasample1.json file which has the description of bounding boxes of the objects in different frames and their label. eg - `python extract_objects.py datasample1.mov` and `python extract_objects.py datasample7.mov`
- After this there will be two directories created by the name of datasample1/ and datasample7/ . Both these will have subdirectories named after the labels of the objects found in the respective videos.

- Our rescaled images will be of different sizes. So for training and testing classifiers on them, we need to bring them to a common resolution. For this we have the `scaleAndPad.py` program. It takes two arguments - directory-name and resolution. eg -
`python scaleAndPad.py datasample1 32`

- Above command will result in creation of two CSV files named CSVs/datasample132_ftr.csv and CSVs/datasample164_lbl.csv in CSVs/ directory.

- Finally you use any of the classifiers available for training and testing. For example our implementation of Random-forest Classifier can be tested as below :-
`python randomForestClf.py datasample1 datasample7`

- For the last command (classifier) to work properly, You should have the appropriate .csv files for features and labels inside the CSVs directory. By default it is set to search for .csv data of 32X32 sized images. You can change it inside the file by changing the line `Rsol = 32`

Running classifiers using .csv files

- Open any of the classifier script like KNN.py or SVM.py and you'll need appropriately set the first few global variables at the top before running them.

- `Rsol = 50` is for setting resolution of training images to 50X50. And `extnsn = "_grey_ftr.csv"` is using the grey images, or for color use `extnsn = "_clr_ftr.csv"`

- You also need to set `train_dir = "CSVs/train_dir"` and `test_dir = "CSVs/test_dir"` appropriately.

- You'll need appropriate .csv files for depending on the above set variables

Running foreground extraction and classifier

- `motion_detector.py` file extracts the foreground of videos and it imports `classify_foreground.py` script which classifies the extracted foreground objects.

- Again set the Rsol and extnsn etc variable at the top of the files before running.

- `classify_foreground.py` will need the appropriate .csv files and upon first time training of any model, it'll save the trained model as pickle file so that for subsequent runs it can be easily loaded. Saving the training time.

Some Implementation Highlights

Below are some the different techniques tried by us to improve the results :-

Cutting out the entry and the exit of the objects into the video

- It was observed that objects were not very clear while entering into the video or while exiting from the video. So we cut some fraction of frames from the beginning and end.

- Moderated using `edge_cut_factor` variable in `extract_filtered_objects.py` script. Look at the use of this variable in the script to understand the implementation.

For every object skipping variable number of frames once a image is taken

- If we take object images from each and every frame of the video then there would be too much of repetition of similar images which is not useful. It simply increases too computational requirement without improving much of accuracy.

- We number of frames to be skipped depends of the speed of the object. So we use three different factor values for this skipping.

- Moderated using `cut_factor_slow`, `cut_factor_medium` and `cut_factor_fast` variables in `extract_filtered_objects.py` script. Look at the use of these variable in the script to understand the implementation.

To Do

Explore following for feature extraction :

- PCA

- HOG

- SIFT
- AlexNet
- vggnet

Observations about the obtained dataset

- Bicycle and Motorcycle images were of both types - with and without the rider cropped. So it may be helpful in the sense that sometimes foreground detector miss the the rider.
- However for labeling task instructions were given to not include the riders of the two-wheelers. Which is wrong I guess. We should have labeled by taking the rider in the box.