Object tracking, analytics and its applications

Abhinav Sathiamoorthy
Boston University
BU Id: U13420091
abhisat@bu.edu

Gur Asees Singh Chandok
Boston University
BU Id: U95489771
gurasees@bu.edu

Chunar Singh
Boston University
BU Id: U10488522
chunar@bu.edu

Abstract

Object detection and tracking have a wide variety of applications. Currently, keeping a count of people entering or exiting a crowded location like malls, events, etc., is done manually, thereby introducing considerable amount of complications due to human errors, manpower requirement and data maintenance. We use pose detection and train a CNN (convolutional neural net) to recognize everyday objects like people and vehicles present in the real environment from video footage. The system would be able to recognize objects even when they are in motion and track their movement for analysis. We use real-world footage recorded or obtained from the internet to train and test our model. In addition, we modify this system to track vehicles in videos to generate and analyze traffic patterns to plan an intuitive traffic infrastructure.

1 Introduction

1.1 Use case 1: People tracking

We keep a count of the number of people entering and exiting a venue using video footage and use this information to generate statistics about population locomotion (useful in libraries, shopping malls, etc.) that are saved as a CSV file and we use Microsoft Excel to generate plots using the statistics.

1.2 Use case 2: Vehicle tracking

We track moving vehicles in a given still video footage and use the coordinates of the vehicles in every frame to map the movement of vehicles on a particular route in the form of heatmaps.This can provide valuable insights into how traffic signals and streets can be modified for efficient movement of traffic.

2 Development flow

2.1 Identifying applications (research)

The first step involved identifying the relevant use cases of the proposed system to formulate an in-depth design idea and gather all the requirements. While researching we realized how still manual counters(tickers) are used to keep track of the number of people entering a venue. Immediately, we

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

realized that this approach has several flaws. First, this involves a lot of manual work and therefore has a margin for human error. Secondly, the statistics generated out of such an approach can at best give you the number of people entering in a day, while our approach can provide granular details about the movement of people. Finally, our approach would not require any new infrastructure as most entrances already have cameras installed thus saving money on manual labor.

While we were working on implementing the first use case we realized we can extend our approach to detect and track vehicles and use that data to map traffic as heatmaps at intersections with high traffic. The heatmaps generated could be used by cities to tweak stop lights depending on traffic flow at a particular time of day rather than using static values. This would be useful especially because of the way our cities are designed with segregated work and residential areas, traffic flows chain substantially depending on the time of the day. Finally, just like our last use case, this does not involve new infrastructure costs because most modern cities have cameras on intersections.
As a part of our future work, we would also like to be able to detect the duration of each color of stop light so that our system could recommend changes to the lights without analysis by a human. We would also like to integrate the two systems to get a better understanding of how pedestrians interact with traffic to suggest changes for pedestrian movement as well.

2.2 Model selection

The next step is where we evaluated different approaches to detect and track people. We also had to decide which neural network models to use for our particular use case. We were looking for a model that not only gave the right output but also in a way that we understood so that we can tweak it as required and therefore we choose Pose detection for person detection and Feature classification four detecting vehicles.

2.3 Pose detection (person detection)

In this step we perform the task of human pose estimation of multiple people in real world video footage. We use an approach that jointly solves the tasks of detection and pose estimation.

2.4 Feature classification (vehicle detection)

In this step, we perform a HOG based feature extractor on a labeled training set of images and trained a Linear SVM classifier. We implement a sliding-window technique and used our trained classifier to search for vehicles in images. We run our pipeline on a video stream and created a heat map of recurring detections frame by frame to reject outliers and follow detected vehicles. We estimate a bounding box for vehicles detected.

2.5 Tracking

Once the object (person/vehicle) is detected, the next step is tracking. We break videos into frames and use the models to detect objects frame by frame and track them through different frames. We keep tracking the object till it is in the frame so that we do not count it multiple times. Therefore, we keep a track of unique detections in the current frame and use it to compute the cumulative number of objects.

2.6 Object detection and tracking

Finally, once the object is detected and tracked, we construct bounding boxes around the object and overlay it on top of the original video. We also add text to each frame which contains the number of detection in this frame, cumulative detections,frame number,time required to process. Finally, we save the frames back into a video file.

3 Design and implementation

3.1 Preprocessing

During preprocessing we read the video, get information about the video like size, duration, fps. Using the information we resize the video to match our specification and decide how to iterate over the entire video frame by frame using "Moviepy", which is an opensource python library to read and tweak video files frame by frame.In Future, we would also like to use background subtraction algorithms during the preprocessing phase to improve accuracy. Resizing is done so that we do not have to worry about scaling the detections to the current size of the video.

3.2 Pose estimation and detection

We use a partitioning and labeling formulation of a set of body-part hypotheses generated with CNN-based part detectors. An instance of an integer linear program, implicitly performs non-maximum suppression on the set of part candidates and groups them to form configurations of body parts respecting geometric and appearance constraints. It infers the number of persons in a scene, identifies occluded body parts, and disambiguates body parts between people in close proximity of each other. The model generates a set of proposals for each frame of the video. Each proposal denotes the index of the video frame, the spatial location of the proposal in image coordinates, the probability of correct detection, and the type of the body joint (e.g. ankle or shoulder).[4]

3.3 Tracking heuristics

Apart from using pose detection to detect and track people, we use the strategy of analyzing the coordinates of the bounding boxes and comparing them with a threshold coordinate radius. While the coordinates of the bounding box stay within this radius range, the bounding box is associated with the same person. Once the person moves out of the frame, the model stops tracking the person as he/she is out of the tracking radius. Using this along with pose detection ensures that despite multiple people occluding each other our system performs reasonably.

3.4 Neural network model

The structure of the convolutional network used to generate person-conditioned proposals is shown in Fig. 4. The network uses the ResNet-101 from [1] that is modified to bring the stride of the network down to 8 pixels [2]. The network generates predictions for all body parts after the conv4_4 block. The cross-entropy binary classification loss is used at this stage to predict the part heatmaps. At each training iteration we forward pass an image with multiple people potentially in close proximity to each other. A single person is selected from the image and the network is conditioned on the person's neck location by zeroing out the heatmap of the neck joint outside the ground-truth region. The neck heatmap is then passed through a convolutional layer to match the dimensionality of the feature channels and add them to the main stream of the ResNet. Finally a joint prediction layer is added at the end of the network with a loss that considers predictions to be correct only if they correspond to the body joints of the selected person. In the network, the person identity signal is provided by the location of the head. In principle the receptive field size of the network is large enough to propagate this signal to all body parts. However it is observed that it is useful to introduce an additional mechanism to propagate the person identity signal. To that end, intermediate supervision layers are injected for individual body parts arranged in the order of kinematic proximity to the root joint (Fig. 4). Prediction layers for shoulders are placed at conv4_8, for elbows and hips at conv4_14 and for knees at conv4_18. It is observed that such an explicit form of spatial propagation significantly improves performance on joints such as ankles, that are typically far from the head in the image space.[3]

3.5 Training phase

Caffe's [18] ResNet implementation is used and initialized from the ImageNet-pre-trained models. Networks are trained on the MPII Human Pose dataset [1] with SGD for 1M iterations with stepwise learning rate (lr=0.002 for 400k, lr=0.0002 for 300k and lr=0.0001 for 300k).[3]

3.6 Testing phase

During the testing phase we tried to use different configurations for our model. Post that, we tried different videos and camera angles to figure out which angle and lightning conditions give us the maximum accuracy and which leave our system vulnerable. We found that highly zoomed in videos are not great for tracking and cameras with wide fields of view perform much better. In order to improve the performance we even tried skipping few frames while tracking people because they have limited movement per frame. This approach gives us great performance but does not work well if there are too many occlusions in the frame. However, using alternate frames provides the right balance of performance and accuracy.

Range of number of people in frame	Actual number of People	Detected number of people without skipping	Detected number by skipping 4 frames	Detected number by skipping 1 frame
Few	3	3	3	3
Average	8	8	7	8
Many	17	16	22	13

Figure 1: Pose estimation on a frame of a video with high levels of movement.

4 Experiments

4.1 Pose detection/estimation

We perform pose estimation on a Youtube video of Mr.Bean dancing that involves frames with high as well as low levels of human movement. The image of the estimated pose in a sample frame with high levels of movement, overlayed on the original video frame is shown in Figure 1. Similarly, an image of the estimated pose in a sample frame with low levels of movement, overlayed on the original video frame is shown in Figure 2. From both the figures, we can observe that the model accurately detects the human pose at points such as ankle, joints, shoulders, face, etc. The intensity, speed or levels of movement/motion in a particular frame of the video does not affect the output of the model.

Figure 2: Pose estimation on a frame of a video with high levels of movement.

Figure 3: Pose estimation on a frame of a video with low levels of movement.

4.2 People tracking - 1 person in frame

We run our tracking model to track people in a video taken outside Rajen Kilachand Center for Integrated Life Sciences and Engineering at Boston university. The output generated by the model draws a bounding box around all the people detected in a particular frame and tracks the person in the frames following the detection. We first test the frames where only 1 person is present. We observe that in Figure 3, the model generates an accurate bounding box and tracks the person in the frame. However, in Figure 4, there is no bounding box generated and therefore, the person in the frame is not tracked. The evident reasoning for this observation would be the difference in the body exposure in both the images. In Figure 3, the entire body of the person is visible in the frame, hence it is easy to detect the pose. In contrast to this, in Figure 4, the person is only partially visible in the frame. Moreover, the person is wearing a hoodie, thereby occluding all the features of the body, and hence the model is not able to detect the person as a human.

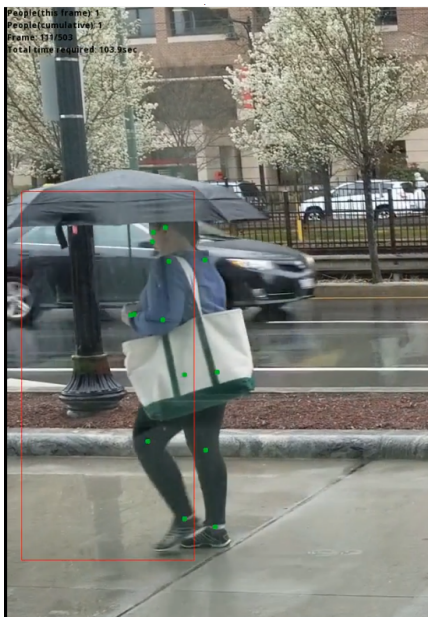

Figure 4: Correct bounding box generated and tracked.

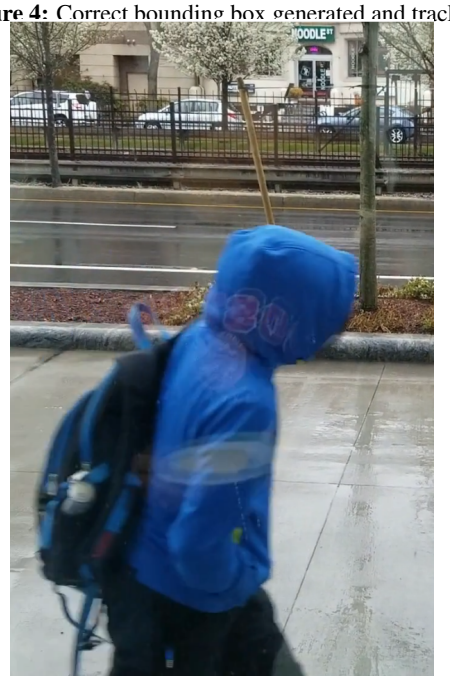

Figure 5: No bounding box generated or tracked.

Table 1: Data from people tracking system

Population movement			
People entering the venue	People exiting the venue	People at the venue	Time period
70	34	36	7am to 8am
50	43	43	8am to 9am
25	51	17	9am to 10am

4.3 People tracking - more than 1 person in frame

Here, we test our model using the same video from the above case. However, now we consider the frames where more than 1 person crossed the path and walked through the frame. The generated output is shown in Figure 5. We observe that the model correctly tracks both the persons in the frame, however, once they start walking together, the 2nd person gets occluded and hence, the model is not able to track the person properly. Hence, the model leaves the generated bounding box for the occluded person in a fixed location (last observed location). The other bounding box that is generated around the person who is entirely visible, correctly tracks the person. Another observation is that when the 2 people separate out later on in the video, the 2nd bounding box is still at its old location, however, a new bounding box is generated around the person who is no longer occluded.

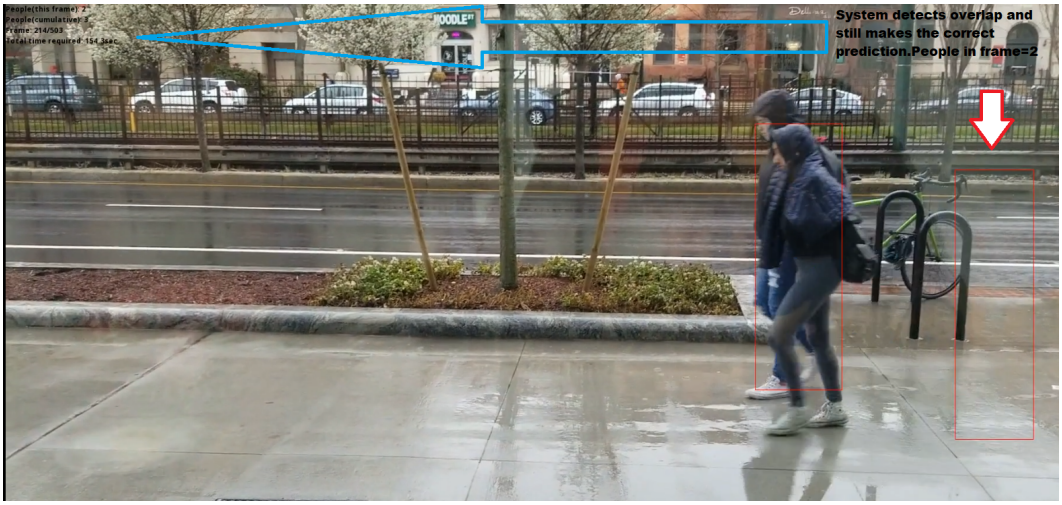

Figure 6: Correct bounding box generated and tracked.

5 Analysis of data

5.1 Tables

We use the people count data provided by our system and tabulate the results as shown in Table 1. This table gives the count of people entering, exiting and present at a particular venue for three different non-overlapping time periods: 7am-8am, 8am-9am and 9am-10am. This data can be used to visualize movement patterns of people and make intuitive changes with respect to events, closure times for a particular venue based on the count of people at the venue on an hourly basis.

5.2 Plots

The data from Table 1 is plotted as shown in Figure 7 and Figure 8. Figure 7 gives the hourly count of people entering, exiting and present at the venue. Figure 8 gives the flow of hourly demographics.

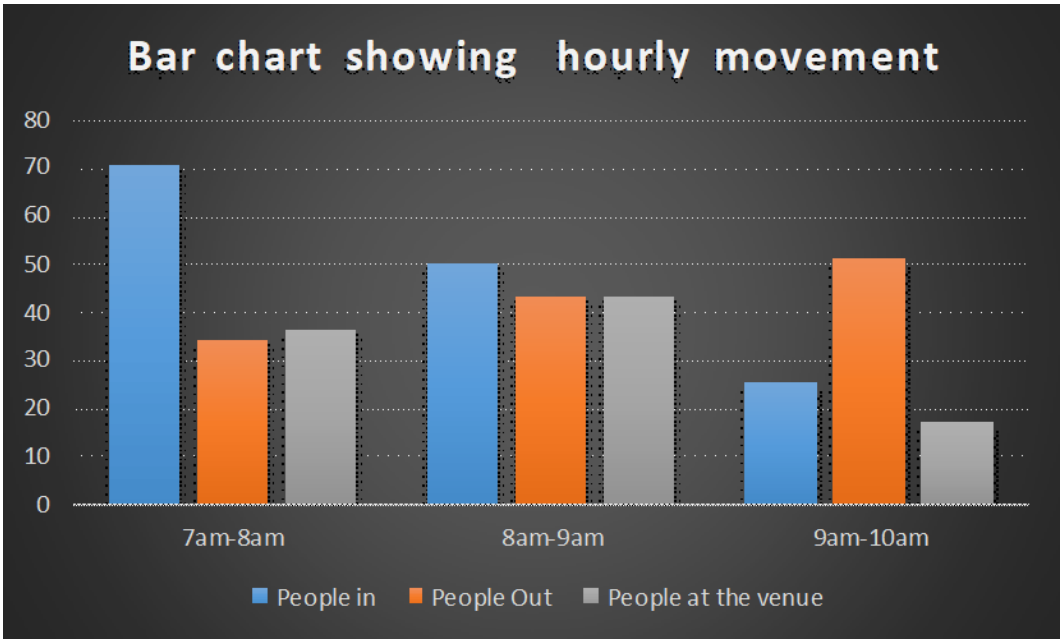

Figure 7: Hourly movement

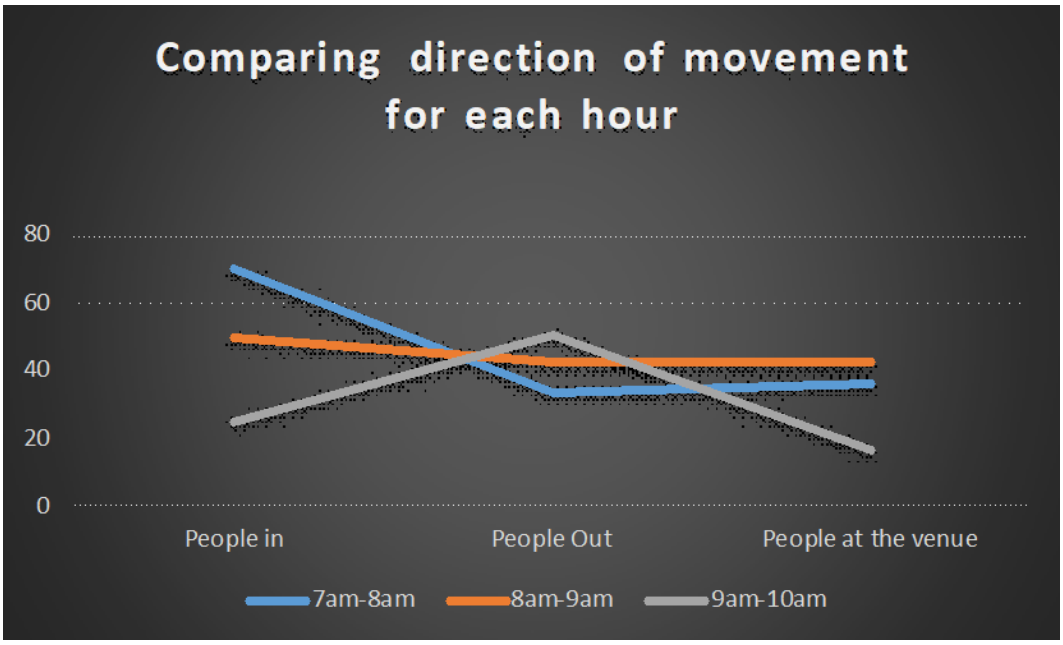

Figure 8: Movement trends

6 Discussion

When it comes to pose detection, euclidean distance between proposals is informative for finding correspondences for slow motions, but fails for faster motions and in the presence of multiple people. DeepMatching is usually effective in finding corresponding regions between the two images, but occasionally fails in the case of sudden background changes due to fast motion or large changes in body limb orientation. In these cases SIFT is often still able to provide a meaningful measure of similarity due to its rotation invariance.[3]

Secondly, by observing the trends of people count from the plots in Figure 6, 7, we can analyze and plan a variety of intuitive demographic changes. The data generated by the model is very

informative and can be used as a reference for strategy making with respect to public events, closure times during the week, maintenance schedules, etc.

Finally, the system generated heatmap of vehicle movement trends shows the most frequently used routes and traffic patterns. This data can be used to plan infrastructure changes in significant locations in cities and towns.

7 Future work

We plan to use our system generated heatmaps as a reference to intuitively adapt to traffic flow in cities and towns and propose effective and efficient changes that can be incorporated into the traffic infrastructure. Our current system generates a comma separated values (.csv) file describing the data collected by tracking people and vehicles in a given video footage. Hence, instead of using Microsoft Excel to plot relevant graphs, we plan to directly illustrate the tracking data in the form of plots using native python packages. In addition, we plan to tweak our model and modify our system to use a video footage to detect vehicle collision using approaches described in a few research papers we have read on the topic.Also, we would like to use background subtraction to improve efficiency of our detections.Finally, we would like to use other neural network models like YOLO and modify relevant parameters to improve testing accuracy of the video footage.

Acknowledgments

We would like to thank our professor Prof. Sang ("Peter") Chin for giving us an opportunity to come up with this idea and work on its implementation as a part of our Machine Learning course. We would also like to thank our course TAs Gavin Brown and Xiao Wang for supporting us and guiding us in our work.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. (2015) *Deep residual learning for image recognition.* arXiv preprint arXiv:1512.03385

[2] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele. (2016) *Deepercut: A deeper, stronger, and faster multiperson pose estimation model.*

[3] Eldar Insafutdinov, Mykhaylo Andriluka, Leonid Pishchulin, Siyu Tang, Evgeny Levinkov, Bjoern Andres, Bernt Schiele (2016) ArtTrack: Articulated Multi-person Tracking in the Wild *https://arxiv.org/abs/1612.01465*

[4] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter Gehler and Bernt Schiele (2016) *DeepCut: Joint Subset Partition and Labeling for Multi Person Pose Estimation.* IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

[5] Eldar Insafutdinov (2017) *Human Pose estimation with TensorFlow framework* `https://github.com/eldar/pose-tensorflow`

[6] Bikramjot Hanzra (2015) *Object Tracker written in Python using dlib and OpenCV* `https://github.com/bikz05/object-tracker`