Multi-Object Tracking with Deep Learning

Suvrat Bhooshan
Stanford University
suvrat@stanford.edu

Aditya Garg
Stanford University
agarg94@stanford.edu

Abstract

This project aims at solving the multi-object tracking problem using a deep-learning approach. It builds a detection-based online tracker that has perfect object detection for each individual video frame and simultaneously has no access to any future frames, hence building the trajectory of the objects one frame at a time. It uses a KNN classifier as a baseline to compare other deep-learning models tried for tracking. Two different approaches were taken to tracking, one involving an LSTM trained over the bounding boxes of the objects over the consecutive frames of a video and the other involving a Siamese network trained on the VGG features of the crops of the objects of every consecutive frame in the video sequence. While the LSTM did not perform well, the Siamese network came close to matching KNN and had the potential for further improvement by adding some more complexity to the network.

1. Introduction

This project aims to solve the multi-object tracking problem assuming a perfect object detector. Traditionally object-detection is done using deep-learning approaches while tracking is done using more traditional geometrical and mathematical analysis. This project only picks the tracking problem and runs several deep-learning techniques to accurately track multiple objects in high resolution surveillance videos with static cameras. The intuition behind deep learning was to see if some non-linear models could learn the mathematical and geometrical approaches currently used without any domain knowledge or expertise. The applications for this problem span from simple ideas like security and surveillance to other commercial applications like inventory management, automated checkouts etc.

1.1. Problem Statement

Multi-object tracking is a multi-variable estimation problem where the following is given:

- Sequence of frames $S_{1:t} = \{S_1, S_2, ..., S_t\}$
- Each frame $S_x = \{s_x^1, s_x^2, ..., s_x^m\}$
- s_x^y represents object y in frame x represented by a bounding box coordinates in the frame

The objective of a multi-tracking is to produce $s_{i_s:i_e}^i = \{s_{i_s}^i, s_{i_s+1}^i, ..., s_{i_e-1}^i, s_{i_e}^i\}$, that is the sequence of frames that object i exists in the video[1].

This project employs a detection based approach, that is assumes a separate detector provides the hypothesis for the objects in the image itself and that this detector works at 100% accuracy.

This project also uses online-tracking, that is for current time t, the input can only include frames and object observed up to t and not any future observations. This requires gradually extending the object trajectory contrary to an offline approach which would look at the entire video to propose the best trajectories across all frames combined.

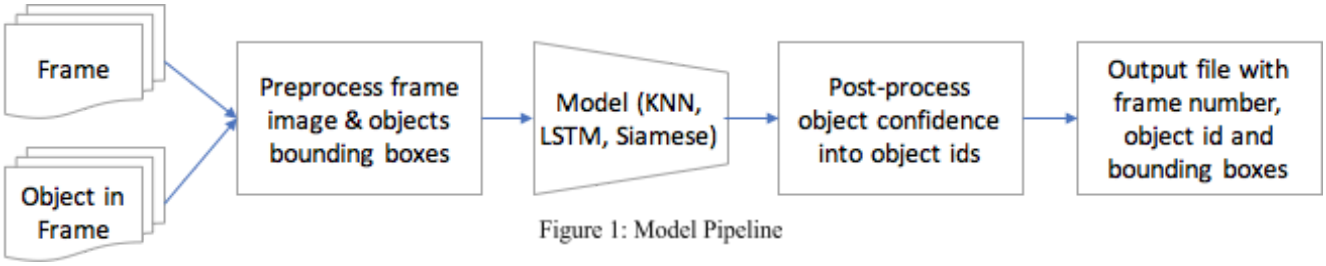

Figure 1: Model Pipeline

1.2. Input/Output Definition

The system takes in set of consecutive frames of a surveillance video along with all the objects detected in each frame described by the bounding box coordinates for each frame.

The data is then preprocessed for the model that is being run and then the processed data is passed to one of the models (KNN, LSTM, Siamese) being run.

The model outputs the confidence of how similar the objects are in consecutive frames and this is then run through a post-processor to assign the object an existing id suggesting it to continue tracking an old object or a new id suggesting the presence of a new object in the video.

The final output is a file containing for each frame, all the objects in the frame with their bounding boxes and the id assigned to the object by the post-processing unit represented by unique colors.

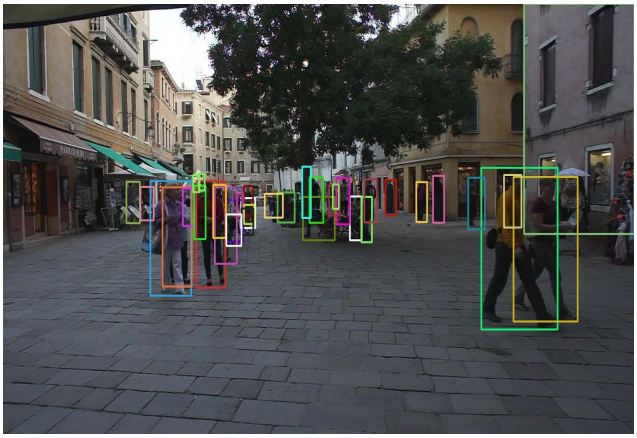

Figure 2 Example Output Image

2. Related Works

Several different approaches to Multiple Object Tracking have been employed. Most approaches can be categorized into two types: Detector Based Tracking (DBT) approaches and Detector Free Tracking (DFT) Approaches [2]. In DBT approaches, a detector is employed to get object hypothesis [3][4] and then a tracker is used to correlate objects across frames into trajectories. Most DBT approaches use a deep learning approach for their detectors before the performance of a DBT tracker is highly dependent on the tracker's performance. DFT trackers are initialized with starting locations of objects which they track in subsequent frames [5]. For our problem, we focus our work on DBT approaches.

Furthermore, tracking has a problem has been tackled in a sequential manner as well as in batch processing. Existing Sequential methods [6][7][8], also referred to as online algorithms, handle frames in a step by step method where only information up to the current frame can be used to track objects. On the other hand, methods which employ both past and future frame knowledge [9][10][11] are known as offline methods. For our problem, we study only online methods of tracking.

Kalman Filter [12] and Kernel tracking algorithms like a mean shift tracker [13] are examples of DBT trackers which require object detection in each frame. However, they are not precise enough to handle very dense frames with multiple objects simultaneously. A lot of domain knowledge and expertise goes in building trackers on the results of object detection to handle challenges like object occlusion, interaction and overlap, illumination changes, sensor noise, etc. Multi-Hypothesis Tracking (MHT) [14] and Joint Probabilistic Data Association Filters (JPDAF) [15] are two representative methods of tracking objects. Recent papers make use of contextual models [16][17][18] to avoid losing track of the object. However, often times there is not enough training data for these models, and we often want to track objects for a variety of contexts. Another high performing models which use this approach of avoiding errors is of min-cost flow framework for global optimal data association [9]. We distinguish our work from the existing work that has been done by developing new deep learning trackers which do not require any prior domain knowledge or expertise of the environment.

3. Methods

The pipeline for a simple tracking system consists of a frame coming into the stream as an input. An object detector breaks the frame down to objects we wish to track. A tracker will then assign the objects to existing trajectories from previous frames, or tag the object as a newly detected object and start a fresh trajectory with it.

We assume perfect object detection, and consider three different approaches to correlating objects across frames into trajectories: k-Nearest Neighbor, an LSTM based tracker, and a Siamese Network for Similarity Detection.

3.1. k-Nearest Neighbours

k-Nearest Neighbor takes in a list of objects detected in the current frame, and the last known location of objects being tracked as inputs. For every possible pair of current object location and previously detected object location, an Intersection over Union (IoU) score is calculated.

3.1.1 Intersection over Union

IoU, also known as, Jaccard index, is an evaluation metric which calculates the common area of two bounding

boxes, and divides it by the area of the union of the two.

The current object is assigned to the trajectory with which it has the highest IoU score for the last known location of the object of that trajectory, unless, the IoU score with all trajectories is below a certain threshold. In that case, the object is tagged as a newly detected object, and a fresh trajectory is started with it.

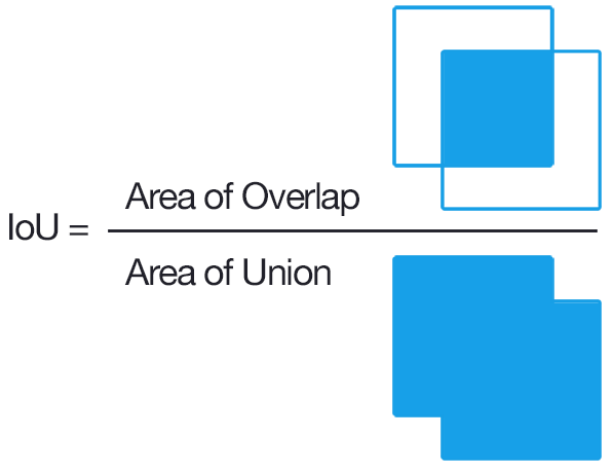

Figure 3: Computing the Intersection of Union is as simple as dividing the area of overlap between the bounding boxes by the area of union [19]

3.2 LSTM based Tracking

LSTM's are a standard deep learning method for learning time dependencies in data. LSTM's improve over RNNs by introducing four new gates, which control how much prior information is forgotten and retained. These gates make sure that the time dependencies are learnt over long periods of time as well, and are not forgotten. The update equations of the four gates of the LSTM are shown below.

$$i_t = \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1}\right)$$
$$f_t = \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1}\right)$$
$$o_t = \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1}\right)$$
$$\tilde{c}_t = \tanh\left(W^{(c)}x_t + U^{(c)}h_{t-1}\right)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$
$$h_t = o_t \circ \tanh(c_t)$$

Figure 4: LSTM update equations (Source: CS224N)

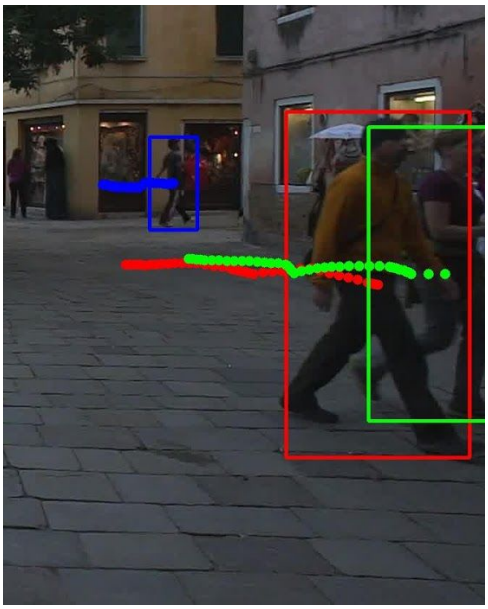

Figure 5: Smooth Trajectories of objects in the data

We can model our sequence of frames as a time-series in which the object is moving through. The intuition behind using an LSTM is that with enough training data, the model should learn what a smooth trajectory of an object moving in a video over time looks like, and therefore, can be used to pairwise determine if a detected object can belong to a trajectory or not. If the detected object does not cross a predetermined threshold for all trajectories, it is considered a newly detected object and a fresh trajectory is started for it.

The LSTM is initialized with the the past known bounding box coordinates of a particular trajectory in sequential order. The bounding box coordinate of the current box is given to the LSTM and the output of the LSTM is fed through a sigmoid classifier to see if the current bounding box will belong to the specified trajectory or not. This operation is done over all pairs of previous tracked trajectories and detected objects in the current frame, to either assign the object to a previous trajectory or classify it as a new object being tracked.

3.3 Siamese Network for similarity of objects

Various trackers use similarity measures such as chi-square and nearest neighbor on the blobs shown by the object detector [21]. We propose pairwise running the objects in the previous frames and the objects detected in the current frame through a siamese network with a sigmoid classifier on top which learns a similarity measure for blobs which represent the same person through multiple frames and distinguishing between different

people across frames.

3.3.1 Transfer Learning

The bounding boxes around an object specify a crop of an image which is resized and passed to a CNN network. We use the VGG architecture [22] for the CNN network to learn the representations of the image. However, since we do not have enough data to properly train such a deep network architecture for learning representations, we initialize the VGG network with weights trained on the Imagenet dataset [23], which are further fine tuned by training on the MOTC dataset.. This process is known as transfer-learning [24]. Transfer learning helps in improving the performance in a new task by transferring the knowledge it has learnt in performing a similar task. In our case, the VGG network had learnt to extract features or representations from an image on the imagenet dataset, on top of which a softmax classifier operated. In our problem domain, the VGG network will learn the image representations on the bounding box crops of people, and a sigmoid classifier will output if two of these representations belong to the same person or not. The network architecture of the full siamese network is shown below.

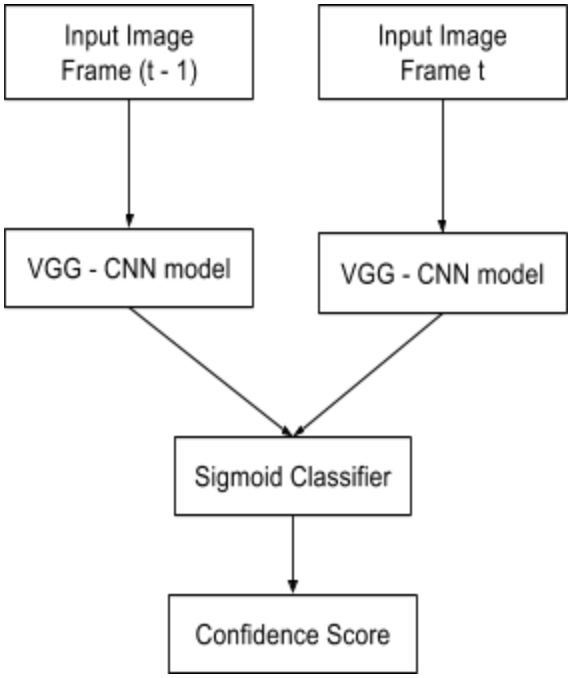

Figure 6: Siamese network architecture

4. Data & Features

The Multi-Object Tracking Benchmark 2016 [20] dataset was used for this project. The dataset details are as follows:

- 7 Training Videos
- 7 Test Videos
- Total Training Frames: 3579
- Total Testing Frames: 4725

This is a high density dataset with very high overlap. Almost all videos are very high resolution (1920x1080) and are mostly recorded at 30 frames per second.

In this dataset, each video is provided as a set of images where each image represents a specific frame of the video sequence. With that it provides a ground truth (GT) file in the form of a csv with the following columns:

- Frame Number
- Object Id
- Left Coordinate of Bounding Box
- Top Coordinate of Bounding Box
- Width of Bounding Box
- Height of Bounding Box
- Confidence Score (a flag in case of GT file)
- Class (Type of object)
- Visibility ratio

Only 6 videos were used for training and 6 for testing as one of them had non-static cameras. Different models required different forms of feature extraction and pre-processions.

Image crop resizing was done in some cases which required the extraction of each object from the frame and resizing it to a fixed size while retaining all the three layers. This could then be run through a feature extractor like VGG. Objects were matched across frames using an IOU technique where every object outputted by the model was correlated to the ground truth by finding exact/closes matches with the bounding boxes and ensuring the object assigned by the model over consecutive frame was the same.

5. Experiments

We had 6 training files and 7 testing files available to us. We used 5 of the 6 training files for training, and the 6th for validation purposes.

5.1 k-Nearest Neighbor

The kNN algorithm simply remembered the data it was fed in, and used it to classify subsequent objects detected in the future. There was no explicit “training” phase of the algorithm. We used the validation file to tune our threshold parameter, i.e. at which threshold should we assign an object to an existing trajectory vs. tagging it as a new object.

5.1.1 Threshold Parameter

Threshold values were searched in the space of 0.0 to 1.0 in intervals of 0.1. A finer search was conducted between the interval which minimized the switch rate on the validation file.

5.2 LSTM based Tracking

The LSTM network was passed in bounding boxes of the object in multiple frames as time-series. Positive examples were constructed by taking bounding boxes of an object from successive frames upto length t, where optimal length t was decided by validation on the 6th file. Negative examples were similarly constructed, however the bounding box of the last frame belonged to a different object than those of the first t -1 frames.

Adam optimizer was used to train the model. Validation on the dev file was conducted to find the optimal hidden size of the LSTM cell. 5 values for the LSTM size were searched: 32, 64, 128, 256, 512, with the one with the least switching rate on the dev set chosen for the final model. A threshold on the confidence score to distinguish between new objects and existing objects was determined in the same manner as for kNN described above.

Experiments on synthetic data to see if the network was learning simple trajectories were also conducted. Their results are discussed in the section below.

5.3 Siamese Network for similarity of objects

In training the siamese network, the objects from each image frame were extracted and reshaped to size 224 by 224. Training batches were created by taking same objects from consecutive frames and negative examples by taking different objects from consecutive frames. Since, the number of negative examples was roughly 50 times higher than positive examples, the negative examples were downsampled to create a 1:9 ratio of positive to negative examples for better training. The weights of the VGG model were initialized by training on the Imagenet dataset and were shared across the two parallel layers of the siamese network. Adam optimizer was used to train the network, with a decaying learning rate. The threshold for the confidence score for distinguishing between a new object and an existing one were chosen the same way as in the case of the kNN algorithm described above.

6. Results/Discussion

6.1. Error Evaluation Metrics

Three different error metrics are conventionally used to evaluate multi-object tracking systems

6.1.1 ID Switches

The ID switches metric counts the number of times the id for a single object switches across all its frames of existence. This describes how many times the system misses an object by tagging it by a new id or simply an incorrect one.

- For all objects $O = \{O^1, O^1, ..., O^m\}$
- Each object $O^x = \{o_i^x, o_{i+1}^x, ..., o_j^x\}$, that is all the frames i through j, the object exists for
- where o_t^x is the object id of object x at frame t

Then:

$$ID\ Switches = \frac{\sum_{x=1}^{m} \sum_{i=1}^{t(x)-1} 1(o_i^x \neq o_{i+1}^x)}{\sum_{y=1}^{m} t(y)} \quad [25]$$

Here, ID Switches is the weighted average for the ID switches across each object, where the ID switches of each object is defined as the number of switches in the id of a given object divided by the number of frames, the object was existing in the entire video sequence. This simplifies to the above formula where m is the total number of true objects and t(m) represents the number of frames the object existed for in the given video sequence.

6.1.2 Multi-Object Tracking Accuracy (MOTA)

The Multi-Object tracking accuracy is a standard metric used to account for all the object configuration errors. It combines the false positive, misses and mismatch rates over all the frames.

For this system, since their is a perfect detector, the false positives and misses are all zero as the detector give exactly the same number of object hypothesis as the ground truth.

Mismatch rate is defined as follows:

$$Mismatch\ Rate = 1 - \frac{\sum_{x=1}^{m} max\left(\sum_{id}^{ids(x)} \sum_{i=1}^{t(x)} 1(o_i^x = id) \right)}{\sum_{y=1}^{m} t(y)} \quad [25]$$

Here, mismatch rate is the weighted for the mismatch rate of each object, where the mismatch rate of each object is defined as one minus the mismatch accuracy. Mismatch accuracy is the ratio of the largest number of frames the object was given the same id over the total frames the object existed for in the video sequence.

6.1.3 Robustness

Robustness is defined as the ability of the system to handle occlusion. Since this system only compared object pairs and hence was incapable of handling occlusion.

6.2. Result/Discussion

6.2.1 K-Nearest Neighbours

This was suppose to be the base case to see how a simple classier would perform on a dense dataset. It did really well, with the following results.

Table 1. KNN Results

	ID Switches	MOTA
Training	0.03	0.78
Testing	0.03	0.78

Plotting the outputs into the frames showed the classifier work very well because of the extremely high overlap between objects in consecutive frames. The perform did deteriorate when run on every 6th frame of the video (to match realistic real time object detection speeds). Moreover, the mismatches that did occur where mostly when two individuals walking in opposite directions cross paths. The classifier could not figure out if they were same, new objects and either assigned new ids or switched their ids.

This made sense, as the KNN only worked only on the bounding box coordinates of two consecutive and had no knowledge of the trajectory or the contents of the object in consideration.

6.2.2 LSTM

The LSTM was run with a sliding window of 10 frames at a time, however it performed really poorly with the results as follows:

Table 2. LSTM Results

	ID Switches	MOTA
Training	0.51	0.27
Testing	0.57	0.22

Even though the models training accuracy was fairly high (around 98%), it never seemed to learn anything as it mispredicted both the training and test examples when run through the entire pipeline. The high training accuracy for the model was due to class imbalance in the favor of negative results in the training dataset. Downsampling the number of negative examples did not help improve the final performance of the model.

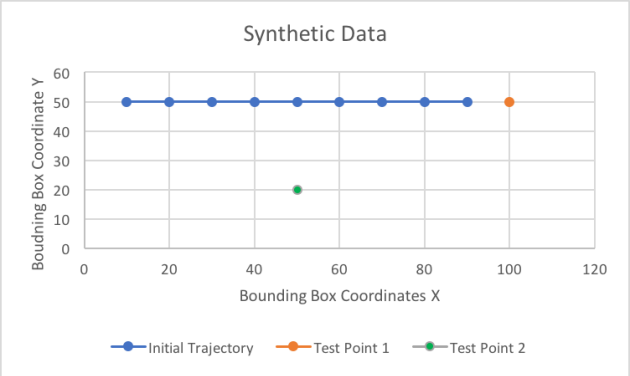

Figure 7:Synthetic Data Experiment

We ran some experiments on synthetic data to see if the network was able to learn simple trajectories. We initialized the LSTM with an object moving in a straight line for 9 consecutive frames (shown in the blue line in Figure 7). Scores were checked for two possible future objects: one in a straight line (Orange Dot), and a random point (Green dot). Intuitively, the orange dot should belong to the blue trajectory and therefore, its score should be much higher. However, the network predicted very similar low scores for both these points and hence, would initialize both these points as new objects being tagged.

The reason for the poor performance is not very clear, but training on the image crop or VGG features of the crop might have performed better and would be something worth testing [25].

6.2.3 Siamese Network

The Siamese was run with every consecutive frame in the video sequence and had the following results:

Table 3.Siamese Network Results

	ID Switches	MOTA
Training	0.11	0.81
Testing	0.15	0.76

(a) (b) (c)

Figure 8: Crop of object at Frames 1, 40 and 80 resized to 224x224x3

The accuracy was very similar to that of a KNN,

however the reason for the accuracy was very different. Unlike the KNN, Siamese looked at the actual image and its VGG features instead of the bounding box coordinates. This made it look for similarity and overlap in the image crop. Hence, it found images Figure 5.b and Figure 5.c to be similar with a very high confidence, but failed to recognize Figure 8.a since it was occluded by a different object.

Moreover, considering the time complexity of extracting each crop of each object in a frame, resizing and running it through a VGG network even before running through the actual Siamese network, makes this algorithm extremely slow and not feasible for any real time environments.

7. Conclusion

Tracking with deep-learning is still a relatively new problem and even though a simple KNN classifier achieved fairly good results, it has a lot of scope for improvement. This is because even the fastest object detectors operate at a speed of 5 frames per second [26]. Hence, in real time, you can run your tracking algorithm only on every 6th frame, instead of on every frame. And while kNNs high pixel overlap in consecutive frames makes it perform really well, the performance drops significantly when it is run every 6th frame. This is where deep learning models can outperform simple geometric algorithms. Siamese networks do not show significant drop in accuracy when they are run every 6th frame to track objects, because they are based on matching similar objects.

The failure of the LSTM was quite unexpected and needs deeper analysis before eliminating it as a viable option for tracking. Siamese networks came close to KNN with a very simple yet computationally expensive pipeline and would have the potential to solve this problem in a real world setting.

8. Future Work

Tracking with deep-learning is a challenging problem with lot more scope for experimentation. We plan to try the following in the near future

A deeper analysis needs to be performed on the LSTM to understand the reasons for its poor performance. Additionally, increase the complexity of the model by passing it both the bounding box coordinates as well as the object crop features to make the network learn both the image content as well as the position and velocity over time.

Siamese network performed well with just two parallel pipelines looking at consecutive frames. We think it might do much better with a 5-10 parallel pipelines looking at a window of previous frames. This will be something to try in the future.

Another challenge would be to predict the future bounding box instead of the current approach of matching detected object to previously known objects. Not only will this help handle object occlusion but potentially allow the system to run with a detection-free-system where the detection does not have to happen every frame.

The overall robustness of the system also needs to be tested in the future by connecting a real object detector to see how the errors from the detector compound onto the predictions made by the tracker.

We also plan to create a better visualization pipeline to have better demos as well as increase our own understanding of the models behaviour.

9. References

[1] B. Yang, C. Huang, and R. Nevatia, "Learning affinities and dependencies for multi-target tracking using a CRF model," in
Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2011,
pp. 1233–1240.

[2] B. Yang and R. Nevatia, "Online learned discriminative partbased appearance models for multi-human tracking," in Proc. Eur. Conf. Comput. Vis., 2012, pp. 484–498.

[3] B. Bose, X. Wang, and E. Grimson, "Multi-class object tracking algorithm that handles fragmentation and grouping," in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2007, pp. 1–8.

[4] B. Song, T.-Y. Jeng, E. Staudt, and A. K. Roy-Chowdhury, "A stochastic graph evolution framework for robust multi-target tracking," in Proc. Eur. Conf. Comput. Vis., 2010, pp. 605–619

[5] M. Yang, T. Yu, and Y. Wu, "Game-theoretic multiple target tracking," in Proc. IEEE Int. Conf. Comput. Vis., 2007, pp. 1–8.

[6] W. Hu, X. Li, W. Luo, X. Zhang, S. Maybank, and Z. Zhang, "Single and multiple object tracking using log-euclidean riemannian subspace and block-division appearance model," IEEE Trans. Pattern Anal. Mach. Intel., vol. 34, no. 12, pp. 2420–2440, Dec. 2012

[7] L. Zhang and L. van der Maaten, "Structure preserving object tracking," in Proc. IEEE Comp

[8] J. Zhang, L. L. Presti, and S. Sclaroff, "Online multi-person tracking by tracker hierarchy," in Proc. IEEE Int. Conf. Advanced Video Signal-Based Surveillance, 2012, pp. 379–385.

[9] D. Sugimura, K. M. Kitani, T. Okabe, Y. Sato, and A. Sugimoto, "Using individuality to track individuals: Clustering individual trajectories in crowds using local appearance and frequency trait," in Proc. IEEE Int. Conf. Comput. Vis., 2009, pp. 1467–1474.

[10] C.-H. Kuo, C. Huang, and R. Nevatia, "Multi-target tracking by on-line learned discriminative appearance models," in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2010, pp. 685– 692.

[11] J. F. Henriques, R. Caseiro, and J. Batista, "Globally

optimal solution to multi-object tracking with merged measurements," in Proc. IEEE Int. Conf. Comput. Vis., 2011, pp. 2470–2477.
[12] Isard, M., Blake, A.: Condensation - Conditional Density Propagation for Visual Tracking. International Journal of Computer Vision (1998)
[13] Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based Object Tracking. IEEE Trans. on Pattern Analysis and Machine Intelligence (May 2003)
[14] Reid, D.: An Algorithm for Tracking Multiple Targets. IEEE Trans. Automatic Control 24(6), 843–854 (1979)
[15] Bar-Shalom, Y., Fortmann, T.: Tracking and Data Association. Academic Press, London (1988)
[16] Babenko, B., Yang, M., Belongie, S.: Visual Tracking with Online Multiple Instance Learning. In: IEEE CVPR (2009)
[17] Li, Y., Huang, C., Nevatia, R.: Learning to Associate: HybridBoosted Multi-Target Tracker for Crowded Scene. In: IEEE CVPR (2009)
[18] Yang, M., Wu, Y., Hua, G.: Context-Aware Visual Tracking. IEEE Trans. on Pattern Analysis and Machine Intelligence (July 2009)
[19] "Intersection over Union (IoU) for Object Detection." PyImageSearch. N.p., 27 Sept. 2016. Web. 11 June 2017. <http://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>.
[20] "MOT Challenge." MOT Challenge. N.p., n.d. Web. 11 June 2017. <https://motchallenge.net/>.
[21] Signal & Image Processing : An International Journal (Sipij) Vol.7, No.3, June 2016. SURVEILLANCE VIDEO USING COLOR AND HU MOMENTS (n.d.): n. pag. Web.
[22] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In Proc. BMVC., 2014.
[23] Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 2015.
[24] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," in IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, Oct. 2010.
[25] Bernardin, Keni, Alexander Elbs, and Rainer Stiefelhagen. "Multiple object tracking performance metrics and evaluation in a smart room environment." Sixth IEEE International Workshop on Visual Surveillance, in conjunction with ECCV. Vol. 90. 2006.
[26] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.