Literature Study Visual Detection and Tracking Today

Hanna Hamrell, hanha664@student.liu.se
Klara Hellgren, klahe156@student.liu.se
Denise Härnström, denha296@student.liu.se
Helena Kihlström, helki570@student.liu.se
Axel Nyström, axeny846@student.liu.se

October 2018

Contents

1 Visual Detection **1**
1.1 Object Detection 1
1.2 Face Detection 2
1.3 Detection of Garments 2

2 Tracking **3**
2.1 Multi Object Tracking 3
2.2 Multiple Hypothesis Tracking 3
2.3 High-Speed Tracking-by-Detection Without Using Image Information 3
2.4 SORT 4
2.5 SORT with Deep Association Metric 4
2.6 Person re-identification with AlignedReID 5

1 Visual Detection

Here both current ways to do both object detection and classification, as well of detection and classification of other things such as garments, faces and other attributes are presented.

1.1 Object Detection

In our study, we found that the currently best object detectors publicly available are Yolov3 [17], Mask R-CNN for Object Detection and Segmentation [1] and Keras RetinaNet [14]. The detectors are compared and discussed below. Further, Yolov3 is evaluated based on our hands on experience.

All of these object detectors takes an image or a video frame as an input and outputs coordinates of the bounding boxes around each detected object as well as object class for each bounding box and with what confidence the detection is made [17] [1] [14]. The Mask R-CNN also outputs segmentation masks and it is possible to apply the network to detect instance-specific poses [12].

Comparing the different Average Precision (AP) values using COCOs average mean AP metric, the currently most accurate detector is RetinaNet, but the other two detectors are not far off, see table 1.1.

The network speed is also of interest when comparing the networks, but it is not easily comparable only by reading their respective papers. At which speed the detections are performed is not always expressively written and even if they are, it is not always mentioned at which hardware, making a just comparison difficult. The creator of Yolov3 claims to be faster than RetinaNet [17].

Table 1: AP vales using COCOs mean Average Precision for different object detectors.

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Yolov3 [17]	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9
Mask R-CNN [12]	ResNetXt-101-FPN	37.1	60.0	39.4	16.9	39.9	**53.5**
RetinaNet [15]	ResNetXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2

The version of Yolov3 tested and used in our analysis module of surveillance videos is a version implemented in Pytorch [16]. It has run slower than the paper claims, which could be because of hardware or because of the python implementation. It has proved good at detecting people who are in the foreground and middleground of a surveillance camera video frame, but it struggles with detecting people in the frame background. It also struggles with detecting people who are very close to the camera and where half of the person is not in the image, as well as with detecting bags. It mainly manages to detect bags when the bag is seen from straight behind the person and then only momentarily.

Yolov3 is both trained using multi-scale and predicts boxes at three different scales using a concept similar to feature pyramid networks [17]. To improve the detections of people who are very small or large, the CNN could be made to extract features from more scales. However, a speed versus detection accuracy trade-off would have to be made.

The problems with detecting bags and people who are only half-visible is difficult to fix without retraining the network using training data with more and better examples. Possibly some of it will be helped by using more scales.

Yolov3 was chosen before RetinaNet and Mask R-CNN since it seems faster per default and since a segmentation mask is not necessary for the first implementations of the analysis module.

1.2 Face Detection

Today there are many commercial face detection API:s on the market, e.g. Trueface, Microsoft Face API and Kairos to only mention a few. However, this literature study will only cover open source projects since they are completely free. The most promising open source face detector we have found is DFace [6] and Face Recogniton [8]. These are both promising since their last updates, in the time of writing, is done within a year and both have many star marks on Github. Most importantly, they can both detect faces in images. This can e.g. be used in the analysis module to for each person object detection, detect all frames where the persons face is shown. The detected faces can then be extracted and presented to find a good image of a person who the police is looking for.

According to their own git repository, the Face Recognition model has "99.38% accuracy on the Labeled Faces in the Wild benchmark", but there is no paper to support this. There is no paper published for the DFace model either and no information about accuracy nor speed, which makes comparison difficult.

1.3 Detection of Garments

To do detection of garments, we propose use of the Part Grouping Network (PGN), which does instance-level human parsing [10] [9]. This network has proved to perform superior on the CIHP dataset, which is a diverse dataset including 38,280 images collected from real world scenarios containing two or more people in each image in challenging poses and viewpoints, severe occlusions, many different resolutions and various appearances. Once the PGN has done the instance-level segmentation, the dominant colors of a garment can easily be extracted using for example k-means clustering [10].

Another possibility using the PGN is to use it for face detection, since it also extracts each persons face. Our first testings have proved the PGN to be very slow, so whether using PGN is better than implementing a separate face detector API as well, depends on the possibilities to speed up the PGN versus how fast the system can run using both.
There are many other networks performing instance-level parsing of humans publicly available, but most focus on single-person parsing and are trained on e.g. fashion images, which severely limits its usability for analyzing videos from surveillance cameras [19].

2 Tracking

Here both current algorithms used for tracking and person re-identification are presented.

2.1 Multi Object Tracking

Keeping track of several objects in a video is called Multi object tracking, shortening MOT. In a crowded scene a useful approach is tracking-by-detection. This method distinguish between the detection and tracking of the object. The general idea for tracking-by-detection is to first localize all objects with an object detector and then associate the detections between frames using object location and appearance.

A popular benchmark for evaluate multiple object tracking is the MOTChallenge dataset. The MOTChallenge is a yearly competition with a public leaderboard that evaluates tracking performance on a collection of videos with difference in resolution, frame rate and illumination.

In surveillance tracking both performance and speed are of interest. Real-time tracking requires online models that only uses information from current and past frame. The focus in this study will therefore be on online tracking-by-detection models. The focus will also be on recent trackers that have publicly available source code. All presented trackers have been competed in the MOTChallenge.

2.2 Multiple Hypothesis Tracking

Multiple Hypotheses Tracking (MHT) is a traditional method that was originally proposed in 1979. The MHT first calculate the probability of each potential track and then assign all potential track to an object. This creates multiple possible tracks and the tracker keeps multiple hypothesis active and can therefore delay the data association. Each track is weighted with a score from motion and appearance.

MHT is very useful in situations where motion is unpredictable and MHT have recently been revisited in tracking-by-detection scenarios. With better detection the MHT show promising result and perform well in the MOTChallenge. One of these revisited MHT is MHD-DAM [5][18]. However, MHT have high computational and implementation complexity and the tracker is memory intensive and generally slow.

2.3 High-Speed Tracking-by-Detection Without Using Image Information

As the name suggests High-Speed Tracking-by-Detection Without Using Image Information is a very fast tracking algorithm [7][4]. The speed of the tracker makes it usefull for online applications. The tracking algorithm matches a detection with a track by only finding the highest Intersection of Union (IoU) between the bounding boxes from the current frame with bounding boxes from previously detected objects. This simple model greatly depends on the object detector. The main drawback of this approach is that it cannot handle occlusion and demands accurate detections in every frame.

2.4 SORT

Another fast but more advanced tracking algorithm is Simple Online Real Time Tracker (SORT) [3][2]. The algorithm SORT keep track of each object by estimating an object model for every frame. The object model contains current spatial information about object position, scale and bounding box ration. The object model also contain motion prediction for the next frame that is estimated using Kalman filtering.

The SORT algorithm solves the data association problem by calculating the bounding box similarity between objects and detections. This is done by calculating the bounding box IoU distance between the current bounding box geometry for the previously detected object and the current bounding box geometry for the new detection. For the previously detected objects, the current bounding box geometry are estimated using the motion prediction. After calculating the IoU distance, the final assignment problem is solved by using the Hungarian method.

2.5 SORT with Deep Association Metric

This algorithm will be referred to as *deep-sort*, and it is an extension of the SORT algorithm described in the previous section. SORT is fast and simple, while it performs very well in terms of precision and accuracy. However, it also delivers a relatively high number of identity switches. This motivates an improvement using descriptors for the visual appearance of the detected objects, which are used when matching detected objects from one frame to another to keep track of identities throughout a video sequence.

The descriptor used in deep-sort is obtained from a convolutional neural network (CNN) that has been pre-trained on a large re-identification dataset with the purpose to discriminate pedestrians. This means that the network has been trained to produce descriptor vectors that are far apart for detected pedestrians with different identities, and very close apart for images of the same person. [21] In deep metric learning methods, the notion of similarity is included directly in the training objective. The feature vectors that are generated for re-identification of the persons in the scene using deep-sort are based on cosine similarity. [20]

The Kalman filtering handles occlusion, but when an object has been occluded for several frames, the prediction becomes more uncertain. Hence, the probability mass spreads out in the state space and there is a risk that a larger uncertainty is prioritized because of the reduced distance in standard deviations of any detection towards the projected track mean. Deep-sort solves this problem by using a matching cascade that prioritizes more frequently seen objects. [21]

As previously stated; the deep-sort algorithm is an *extension* of the SORT algorithm. Comparing the cosine distance between the feature vectors is a complement to measuring the IOU distance and the distances between Kalman state estimations. Deep-sort takes all these things into account, which is why it is considered to be an improvement of the SORT algorithm. [21]

2.6 Person re-identification with AlignedReID

A paper[22] from 2018 introduces an algorithm called AlignedReID which the authors claim performs person re-identification better than human annotators. AlignedReID uses a CNN to jointly learn global and local features to represent a person image.

A feature map of size $C \times H \times W$ is taken from the last convolutional layer of a CNN, for example ResNet50[11]. A global feature vector is then created by using global max pooling, i.e. pooling with a $H \times W$ kernel, which gives a C-dim global feature vector. Local feature vectors are then created by first horizontally max pooling the original feature map to create H different local feature maps and then convolving each local feature map with a 1×1 kernel to reduce the number of channels from C to c. After this a person image is represented by a C-dim global vector and H different c-dim local vectors, where each c-dim vector represent a row of the person image.

The local distance between two person images is calculated by finding the alignment of local features which gives the smallest total distance. It is done by first creating the distance matrix D containing elements $d_{i,j}$. The element $d_{i,j}$ is a normalized distance between local feature vector f_i from one image and local feature vector g_j from another image. The normalizing transformation is done as in equation 1 below.

$$d_{i,j} = \frac{e^{||f_i - g_j||_2} - 1}{e^{||f_i - g_j||_2} + 1} \qquad i, j \in 1, 2, .., H \tag{1}$$

The local distance between two images is then calculated as the shortest path from $(1, 1)$ to (H, H) in matrix D. The global distance is calculated as the L2 distance between the global feature vectors of the images. Finally the total distance between two person images is simply the sum of the local and global distances.

The procedure above with both global and local features is used during the training stage. However, during the inference stage only the global features are used to compare similarity between person images. The authors of AlignedReID found that only using the global feature during inference worked almost as good as the combined global and local features. In [22] they speculate that the reason for this is that the structure prior of the person image in the learning stage makes the model learn better global features and that the local matching makes the global feature pay more attention to the person instead of the background.

AlignedReID uses TriHard[13] loss as metric learning loss and hard samples are mined using the global distance only. Mutual learning is used during training, details about the mutual learning used can be found in [22] and [23].

References

[1] Waleed Abdulla. *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow.* https://github.com/matterport/Mask_RCNN. 2017.

[2] Alex Bewley. *Simple online and realtime tracking.* https://github.com/abewley/sort. 2016.

[3] Alex Bewley et al. "Simple online and realtime tracking". In: *2016 IEEE International Conference on Image Processing (ICIP).* 2016, pp. 3464–3468. DOI: 10.1109/ICIP.2016.7533003.

[4] Erik Bochinski. *High-Speed tracking-by-detection without using image information.* https://github.com/bochinski/iou-tracker. 2018.

[5] Kim Fuxin Chanho, Li Arridhana Ciptadi, and James Rehg. "Multiple Hypothesis Tracking Revisited". In: (2018). DOI: 10.1109/WACV.2018.00087.

[6] *DFace.* https://github.com/kuaikuaikim/DFace.

[7] Thomas Sikora Erik Bochinski Volker Eiselein. "High-Speed tracking-by-detection without using image information". In: (2017). DOI: 10.1109/AVSS.2017.8078516.

[8] *Face Recognition.* https://github.com/ageitgey/face_recognition.

[9] Ke Gong et al. *Part Grouping Network (PGN).* https://github.com/Engineering-Course/CIHP_PGN. 2018.

[10] K. Gong et al. "Instance-level Human Parsing via Part Grouping Network". In: *ArXiv e-prints* (July 2018). arXiv: 1808.00157 [cs.CV].

[11] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *arXiv preprint arXiv:1512.03385* (2015).

[12] Kaiming He et al. "Mask R-CNN". In: *CoRR* abs/1703.06870 (2017). arXiv: 1703.06870. URL: http://arxiv.org/abs/1703.06870.

[13] Alexander Hermans*, Lucas Beyer*, and Bastian Leibe. "In Defense of the Triplet Loss for Person Re-Identification". In: *arXiv preprint arXiv:1703.07737* (2017).

[14] *Keras RetinaNet.* https://github.com/fizyr/keras-retinanet.

[15] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *CoRR* abs/1708.02002 (2017). arXiv: 1708.02002. URL: http://arxiv.org/abs/1708.02002.

[16] *pytorch-yolo3.* https://github.com/marvis/pytorch-yolo3.

[17] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *arXiv* (2018).

[18] Jim Rehg. *Multiple Hypothesis Tracking Revisited.* http://rehg.org/mht/. 2018.

[19] Edgar Simo-Serra et al. "A High Performance CRF Model for Clothes Parsing". In: *Proceedings of the Asian Conference on Computer Vision (2014).* 2014.

[20] Nicolai Wojke and Alex Bewley. "Deep Cosine Metric Learning for Person Re-identification". In: (2018), pp. 748–756. DOI: 10.1109/WACV.2018.00087.

[21] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. "Simple Online and Realtime Tracking with a Deep Association Metric". In: (2017), pp. 3645–3649. DOI: 10.1109/ICIP.2017.8296962.

[22] Xuan Zhang et al. "Alignedreid: Surpassing human-level performance in person re-identification". In: *arXiv preprint arXiv:1711.08184* (2017).

[23] Ying Zhang et al. "Deep Mutual Learning". In: *CVPR*. 2018.