Multiple Object Tracking Based on Faster-RCNN Detector and KCF Tracker

Fan Bu, Yingjie Cai, Yi Yang

Department of Mechanical Engineering
University of Michigan, Ann Arbor, Michigan 48109
Email: fanbu@umich.edu, caiyj@umich.edu, yeeyoung@umich.edu

***Abstract*—Tracking and detecting of object is one of the most popular topics recently, which used for motion detection of various objects on a given video or images. To achieve the goal of intelligent navigation of a moving platform operating on the sidewalk, our goal is to build the software that is able to detect the pedestrians and predict their trajectories, during which process MOT (multiple object tracking) plays an important role. Although different kinds of approaches have been introduced in the class, even in latest research papers, to tackle similar problem, there still exists many issues unsolved. In this report, we inspect the previous work and propose our method for better combination of detection and MOT algorithms. Here we take advantage of the high efficiency and celerity of Faster RCNN (Region-based Convolutional Neutral Network) and KCF (Kernelized Correlation Filter) for the purpose of real-time performance. To validate the effectiveness of our system, the algorithm is demonstrated on four video recordings from a standard dataset. The experimental results on our test sequences illustrate the reliability of our method.**

***Keywords*—Visual detection, Multiple object tracking, Neutral networks, Kalman Filter, Kernelized Correlation Filter.**

I. Introduction

Our project, inspired by ROAHM Lab and Drop Lab in University of Michigan, aims to build a pedestrian assistant system, which requires an intellectual Segway to automatically guide itself on the sidewalk and avoid pedestrian simultaneously. To achieve the goal of intelligent navigation, the whole project can be divided into four steps from the big picture. [redacted], recognize different individuals from the moving binocular cameras. [redacted] calculate relative coordinates of pedestrians and Segway in the camera frame. [redacted] Segway shall collect information and localize itself in the world frame. [redacted] but not the least, apply control strategies so that Segway has the ability to navigate itself and avoid obstacles, including pedestrian, automatically.

Our task for the course project is to solve the first and the third problems that are closely relating to computer vision field, which would be discussed in details vide post. Figure 1 illustrates the main concept for our frame-work.

1) Object: In computer vision, an object is considered as a continuous closed area in an image which is distinct from its surroundings. Objects to track can be pedestrians, vehicles, or a flock of animals. In this project, we mainly focus on the pedestrian. There are three underlying reasons.

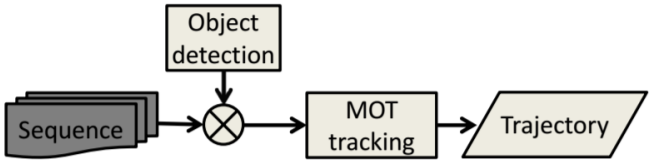


Fig. 1: Basic steps for tracking an object

Firstly, compared to other conventional objects in computer vision, pedestrians are typical non-rigid objects, which is an ideal example. Secondly, video of pedestrians raises a huge number of practical applications which further results in great commercial potential. Thirdly, according to incomplete statistics collected by ourselves, at least 70% of current MOT research efforts are devoted to pedestrian.

2) Detection: Detection is a computer vision task which localizes objects in images. It is usually conducted by training an object detector from a training dataset. In most situations, detection does not involve temporal information.

3) Tracking: Tracking is to localize an identical object in continuous frames. Thus tracking always involves in a video or image sequence with temporal information. In MOT, tracking means simultaneously localizing multiple objects and maintaining their identities.

4) Trajectory: Trajectory is the output of a MOT system. One trajectory corresponds to one target; thus a trajectory is unique.

However, the existing algorithms of tracking pedestrians exhibit a low performance when faced with problems like occlusions, moving cameras, illumination changes, motion blur and other environmental variations. In this paper, we propose an algorithm that to apply KCF (Kernelized correlation filters) tracker into pedestrian detection and tracking, combined with R-CNN based object detector. With advances of deep learning based such object detection technology, it becomes feasible to obtain the noise-rejected objects. For tracking, we create KCF trackers for each candidate region using a grayscale and color names appearance model. Sequentially, moving objects are tracked and their states are updated frame by frame. To handle scale variations, object fragmentation, occlusions and lost tracks,

we use simple data association between KCF trackers and targets detected by R-CNN. Object states are determined by using the information of both tracker and detector as they can both make errors at different times. Finally, the system is evaluated with four videos involving a boy, students, pedestrians, running athlete and basketball player. Results show that our method is competitive even if using simple data association.

The paper is organized as follows: In section II, we discuss previous work; In section III, we discuss our technical solution details including basic idea of R-CNN, Kalman Filter, mathematical concept for KCF tracker and, data updating strategy; In section IV, we test our method, including the classification training results (accuracy around 70%) draw comparison between different tracking methods and summarize our results; Finally, in section V, we conclude the paper.

II. Related Work

A. Review of Previous Work

For tracking, there exist basically two approaches, either using optical flow, or using background subtraction. Methods based on pre-trained bounding box detectors are not applied in such scenario because it is difficult to design a universal detector that can detect all possible pedestrians. Object may be missed. The use of background subtraction and optical flow is not without limitation either, because although objects are not usually mis-detected, they are often fragmented into two of more parts.

1) Optical flow-based methods: In this family of methods, objects are detected using optical flow by studying the motion of tracked points in the video. Feature points that are moving together are considered as belonging to the same object, shown as in Figure 2. Several methods accomplished this process using Kanade-Lucas-Tomasi (KLT) tracker [1]. Because objects are identified only by the motion of feature points, nearby pedestrians moving at the same speed may be merged together. Furthermore, the exact area occupied in the frame by the pedestrians is unknown because it depends on the position of the feature points. Finally, when an object stops, its features flow interrupts which lead to fragmented trajectories.

2) Background subtraction-based methods: This second family of methods rely on background subtraction to detect objects in the video. Background subtraction gives blobs that can correspond to parts of objects, one, or many objects grouped together. The task is then distinguishing between merging, fragmentation, and splitting of objects. This approach works fairly well under conditions with no or little occlusion. However, under static or slow motion conditions, cameras are harder to extract pedestrians from the background, may partially occlude with each other and many objects may merge into a single blob. In this case, it is hard to divide them. Mendes et al.[2] proposed a method that combines KLT and background subtraction. This approach increases its accuracy by setting in and out ROI regions to reconfirm objects trajectory and by segmenting region when occlusions are detected.

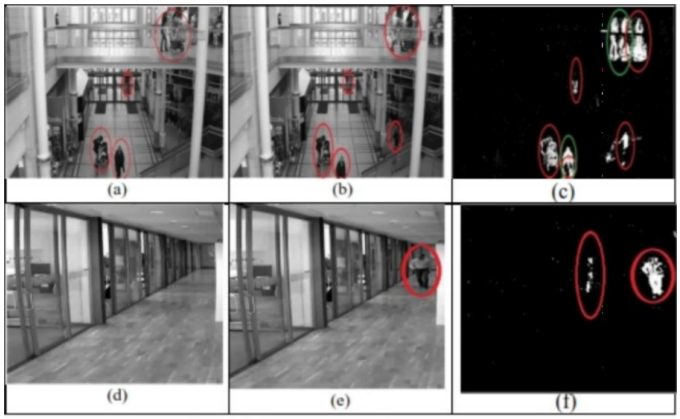
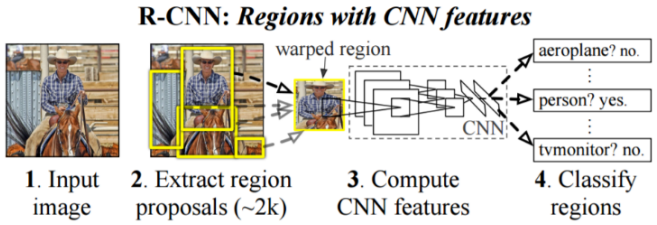
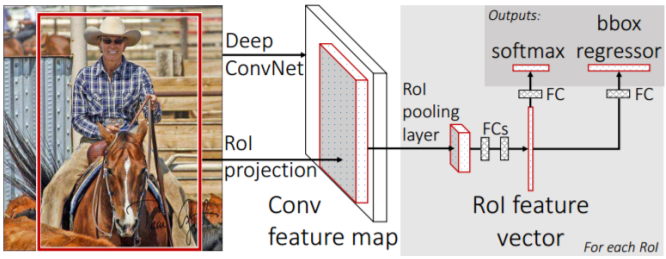


Fig. 2: By Optical flow-based tracker, even the shadow of pedestrians is detected as desired objects.



(a) R-CNN pipeline that uses Caffe for object detection.



(b) An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

Fig. 3: Faster R-CNN architecture

B. Contribution of Our Method

In our work, we combine KCF and R-CNN detector and propose a robust tracking method that can track any types of pedestrian, even applicable for real-time operating situations.

Figure 3 shows the Fast R-CNN architecture. The advantages of this method are:
1. Higher detection quality (mean AP) than R-CNN

2. Training is single-stage, using a multi-task loss
3. An insensitive stronger classifier for illustration variation and appearance model variation
4. Training can update all network layers
5. No disk storage is required for feature caching

As to KCF tracker:
1. Among the state-of-the-art methods, recently the algorithms based on correlation filters shows its advantages both in accuracy and robustness.
2. As a tracking-by-detection tracker, the pro-cessing speed of Kernelized Correlation Filter (KCF) fantastically is fantastically dozens of times that of TLD or Struck.
3. Can be supported by richer powerful features e.g., CNN features and textural features, rather than just using the raw greyscale pixels.

III. Technical Part

A. Tracking Methodology

Our algorithm combines Faster R-CNN with both Kalman Filter and KCF tracker. Faster R-CNN is used for pedestrian detection (candidates to track), and Kalman Filter or KCF tracker is applied for data association and for tracking objects during occlusions. To improve the overall performance of our method, we added different update steps to solve problems like occlusion and noise reduction. Figure 4 shows the diagram of the steps of our method.

For each frame, we adopt Faster R-CNN detects pedestrian candidates object regions. Next, to update the tracked object states, we compare the Center of Detection Region $D_{[i]}$ with current tracks $T_{[j]}$ for data association. This is done by updating all the currently active KCF/Kalman trackers. The KCF/Kalman trackers $\overline{KCF}_j^{t-1}$ find the most plausible data association with a $D_{[i]}$ based on proximity and their internal model. Correspondences may be many to one, one to many, one to one, or an object may be entering and leaving scene. These different cases are determined by counting the number of KCF/Kalman trackers (one per track $T_{[j]}$) associated to each $D_{[i]}$. For objects in occlusion, we update their positions without adapting tracking patch (tracker is more trustworthy, we use the object state provided by KCF/Kalman tracker). For one to one association, we update the state of the tracked object using the information from Faster R-CNN (Faster R-CNN allows adjusting the tracking patch of the KCF/Kalman tracker, and in one to one association, Faster R-CNN is assumed to be often trustworthy). For a new object, we assign a new KCF/Kalman tracker. Finally, undesired objects will be deleted if not detected by Faster R-CNN during several frames. In the next subsections, we describe our method in details (refer to Figure 4).

B. Pedestrian Detection by Faster R-CNN

The complete pipeline of Faster R-CNN [3] from frame to bounding box output is shown in Figure 5. We start with running an algorithm to generate proposal bounding boxes, in this case we use selective search [4], which we cache for use across the process. We pass these bounding boxes along with the original image to the detector which is our convolutional neural network. The CNN produces softmax scores for each bounding box which are used in the final non-maximal suppression step. In our experiment, we only display bounding boxes which softmax scores are higher than 0.8.

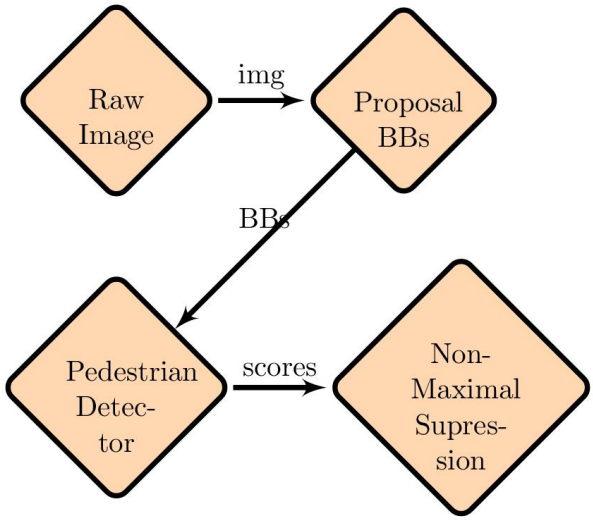


Fig. 5: Pedestrian detection pipeline

C. Kalman Filter

Kalman filter was proposed by Kalman in 1960 [5]. It addresses the general problem of trying to predict the state of a discrete-time controlled process which described by the linear stochastic difference equation:

$$x_k = A_{k,k-1}x_{k-1} + w_k$$

And it was designed for target tracking[6]. Where $x_{k-1} \in \mathbb{R}^n$ is the state at the previous time $k-1$. $x_k \in \mathbb{R}^n$ is the state at the current time k. $A_{k,k-1}$ is a transform matrix.

If defined $\hat{x}\prime \in \mathbb{R}^n$ to be a priori state estimate at time k and $\hat{x}_k \in \mathbb{R}^n$ to be a posterior state estimate. Then a priori and posterior estimate errors are defined as: $e'_k = x_k - \hat{x_k}'$, $e_k = x_k - \hat{x_k}$. A priori and a posteriori estimate error covariance are defined as: $P'_k = \mathbb{E}[e'_k {e'_k}^T]$, $P_k = \mathbb{E}[e_k {e_k}^T]$. Kalman filter function as follows:

$$\hat{x_k}' = A_{k,k-1}\hat{x_{k-1}}$$

$$P'_k = A_{k,k-1}P_{k-1}A^T_{k,k-1} + Q_k$$

$$K_k = P'_k H_k^T \left(H_k P'_k H_k^T + R_k\right)^{-1}$$

Then we have:

$$\hat{x_k} = \hat{x_k}' + K_k\left(z_k - H_k\hat{x_k}'\right)$$

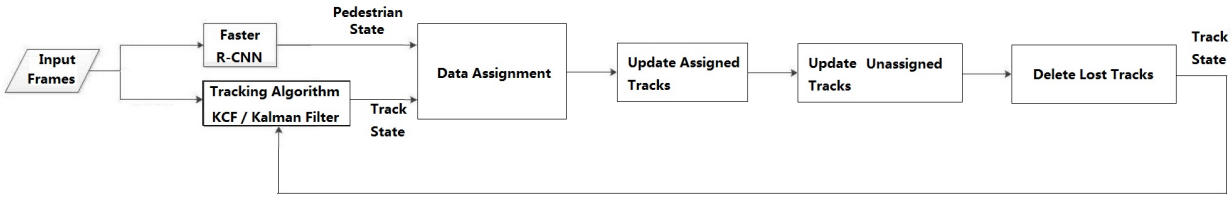


Fig. 4: Diagram of our tracking method

$$P_k = P'_k - K_k H_k P'_k$$

Where Q_k and R_k represent noise. In practice, we initialize $P_0 = 0$, and treated the previous posteriori etimate state $\hat{x_{k-1}}$, P_{k-1} as the current priori estimate state $\hat{x_k}'$, P'_k.

In our pedestrian tracking framework, we define the pedestrain state $X_k = (x_k,\ y_k,\ v_{kx},\ v_{ky})$, x_k and y_k are the coordinates of the pedestrain window center in the image, v_k is the velocity of the window center, v_{kx} and v_{ky} are the projections of v_k in x axes and y axes. In order to reduce the computation of the algorithm we simplify $A_{k,k-1} = A,\ Q_k = Q,\ H_k = H,\ R_k = R$, shown in the following equations.

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$Q = \begin{bmatrix} 0.0001 & 0 & 0 & 0 \\ 0 & 0.0001 & 0 & 0 \\ 0 & 0 & 0.0001 & 0 \\ 0 & 0 & 0 & 0.0001 \end{bmatrix},$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad R = \begin{bmatrix} 0.0001 & 0 \\ 0 & 0.0001 \end{bmatrix}$$

We track the pedestrian frame-by-frame. The Kalman filter tracker predicts the pedestrian state X_k in current frame rely on the previous state.

D. KCF Tracker

KCF [7] is a correlation-based tracker that analyzes frames in the Fourier domain for faster processing. It does not adapt to scale change but updates its appearance model at every frame. KCF can be used with various appearance models. In this work, we use the grayscale intensity suggested by the authors of [8].

1) Data Set: In KCF tracker, a classifier is trained using only a single image patch $x = [x_1, x_2, ..., x_n]$ and its cyclic shifts$\{P^i x | i = 0, ..., n-1\}$, P is the permutation matrix[9]. Unlike the other trackers which based on discriminative method, all of the samples $x_i = P^i x$ for the KCF classifier training are labeled with a Gaussian function y instead of using binary values, so thaty_i is the label for x_i.

2) Classifier: The classifier in the KCF tracker is trained by minimizing the cost function over w.

3) Fast Detection:

E. Object Tracking

Before deciding if at frame t a bounding box region R_i^t will be tracked mostly with the KCF tracker or with background subtraction, we need to determine in which condition it is currently observed. The R_i^t may be entering the scene, leaving the scene, isolated from other objects, or in occlusion. To determine the state of all R_i^t, the active KCF trackers at frame $t-1$, KCF_j^{t-1}, are applied to the original RGB color frame t. Then the resulting tracker outputs TO_j^t are tested for overlap with each R_i^t and vice versa.

1) object states: If a detected region R_i^t overlaps with only one existed track T_j^t , it means that the object is isolated and currently tracked (state: tracked).

If a detected region R_i^t overlaps with more than one existed track T_j^t, it means that the object is tracked but under occlusion (state: occluded).

If a detected region R_i^t does not overlap with any existed track T_j^t, it means that it is an object not currently tracked (state: new object or noise).

If a T_j^t does not overlap with any R_i^t, it means that it is an object not currently visible (state: invisible or object that has left).

The overlap is evaluated using

$$Overlap(x, y) = \frac{x \cap y}{x \cup y}$$

where x and y are two bounding boxes.

2) State: Tracked: If a R_i^t overlaps with one and only one T_j^t, we assume that the object is isolated and tracked correctly (but not necessarily precisely). Since the KCF tracker does not adapt to scale change, we add the R_i^t bounding box to the CT_j^{t-1} associated to KCF_j^t most of the time. The scale can be adapted progressively during tracking based on the Faster R-CNN R_i^t. In this case, we re-initialize KCF_j^t with the information for R_i^t to continue the tracking.

We use T_j^t only if R_i^t suddenly shrinks because of fragmentation of the tracked target. Indeed, background subtraction cannot always handle well abrupt environmental changes which cause over-segmentation. Therefore, when KCF tracker bounding box is much larger than the Faster R-CNN bounding box (as defined by thresholds T_{ol} and T_{oh}), we assume that KCF is more reliable. Formally,

$$CT_j^t = \begin{cases} CT_j^{t-1} \cup T_j^t & if T_{ol} \leq \frac{A(T_j^t)}{A(R_i^t)} \leq T_{oh} \\ CT_j^{t-1} \cup R_i^t & otherwise \end{cases}$$

where A() is the area of the bounding boxes. The upper bound threshold T_{oh} is used to allow adaptation of the scale when the object is leaving the scene. In this case, the object shrinks but it is not caused by over-segmentation, so the use of R_i^t should be considered. Note that when an object is fragmented, KCF_j^t will be associated with the larger fragment. So the shrinking is conditioned by the larger fragment, and as a result, normally the shrinking is limited. If too large, most of the time it is explained by an object that is leaving the video frame. This is why T_{oh} is used.

Finally, even if R_i^t is significantly larger than T_j^t, we still used the background subtraction blobs as shadows are not problematic in our application and their localization accuracy is better than KCF.

3) State: Over Segmented: If more than one T_j^t overlap with a R_i^t, we assume that several previously isolated objects are now in occlusion. Faster R-CNN fails to distinguish objects as they are merged into one box. In this case, we handle the occlusion problem in a straight-through manner and use the bounding boxes outputted by the KCF trackers. That is, we add the T_j^t to the CT_j^{t-1} related to R_i^t with

$$CT_j^t = CT_j^{t-1} \cup T_j^t$$

Faster R-CNN cannot handle over-segmentation problems in some conditions. As can be seen from Figure**??**, when one object starts moving, redundant detections may be created on its segmentation as the object may be temporarily over-segmented. This is not a real case of occlusion (As shown in Figure 6a. We proposed a method to address this problem. If the area of the region R_i^t is smaller than the sum of the bounding boxes TO_m^t and TO_n^t of these two objects for eight consecutive frames, it is more likely to be two detectors detecting the same object. In this case, we delete the shorter-lived box and update the other one with R_i^t. If not, we assume that two objects are really occluded and do not take any action (see Figure 6b). That is,

$$If\ A(TO_m^t) + A(TO_n^t) > A(R_i^t),\ Delete\ KCF_n^t$$
$$Else,\ No\ action.$$

where $A()$ correspond to the area of a bounding box, and KCF_n^t is the most recent of the two tracks tracking the same object.

(a) Redundant detection (b) Real occluded objects

Fig. 6: Example of redundant detections

(a) Group during occlusion (b) Two trackers on the same object (c) Relabel after split

Fig. 7: An example of tracker shifting

4) State: Occluded: Because the KCF tracker sometimes loses track of objects when occlusion occurs, we first need to identify whether R_i^t is not overlapping with any T_j^t because of lost track or because of a possible new object. A problem that sometimes occurs is that after an occlusion two KCF trackers are on a single object, while the other object that was in occlusion is associated with no tracker. In that case, a tracker should be re-assigned to the other object.

To address this issue, we keep track of group of objects. As can be seen from Figure 7, when occlusion occurs, we label occluding objects as being inside a specific group. As soon as a member splits from the group without being tracked, we search for redundant KCF trackers among other group members.When more that one KCF trackers are found within a same group member, we re-assign the tracker that is matching less.

If it is indeed a new object, a new track is initialized with

$$CT_j^t = R_i^t$$

5) State: New Objects: As it is mentioned in subsection III-E1,If a detected region R_i^t does not overlap with any existed track T_j^t, it means that it is an object not currently tracked. However, we should not immediately label the new track as a new object. Instead, it has possibility to be considered as a noise detection. As shown in Figure8, we observed a noise detection on the bottom of the lamp post. To solve this problem, we implement a strategy that counts a new track as a new object if and only if this new track has existed more than six frames and is visible more than 60% of its age. That is to say, objects that exist for less than six frames will be discarded.

Fig. 8: Noise detection

6) State: Invisible or Exited Object: When a T_j^t does not overlap with any R_i^t, we considered that the object is invisible. If the object has exited the scene it will not reappear, but if it was hidden, it should reappear eventually. To distinguish these two cases, we use a rule based on the number of consecutive frames for which the object is invisible. If the object is invisible for more than eight consecutive frames, it will be removed from active tracks.

To improve tracking performance, when part of an object track are missing because the object was at time invisible, we restore the missing part by using interpolation from neighboring frames where the object was successfully tracked.

7) Pseudo Code: Our method is summarized by the following pseudo code.

```
Input: video
Output: trajectories + bounding boxes
    with numbers

for each video frame do
    Detection by Faster R-CNN
    for Each KCF tracker do
        Find Best matching box
    end
    Data assignment based on current
        state
    if State == Tracked
        Update KCF tracker with detection
    end
    if State == Occluded
        Track in straight through manner
    end
    if State == New object
        Create a new tracker
    end
    if State == Invisible or exited
        object
        Delete or retain trajectory
    end
end
```

IV. Experiments

A. Faster R-CNN

We accomplish pedestrian recognition implemented by Faster-RCNN [10–12] and tested with both default [13] and collected dataset, as Figure 9a. Algorithm is deployed in Python. To improve the accuracy, we retrain a new model, based on a combination of the default data and the real data we collected. Table I shows result of accuracy between two models.

TABLE I: Faster R-CNN detection accuracy calculated by $test.py$

	Default Model	Our Model
Average Precision	0.7964	0.7127

The accuracy of our model, 8% less than default, is a price for a more conservative detection strategy, as Figure 9b. Though the result produces an error with an excess detection on the right image, scarcities are successfully avoided.

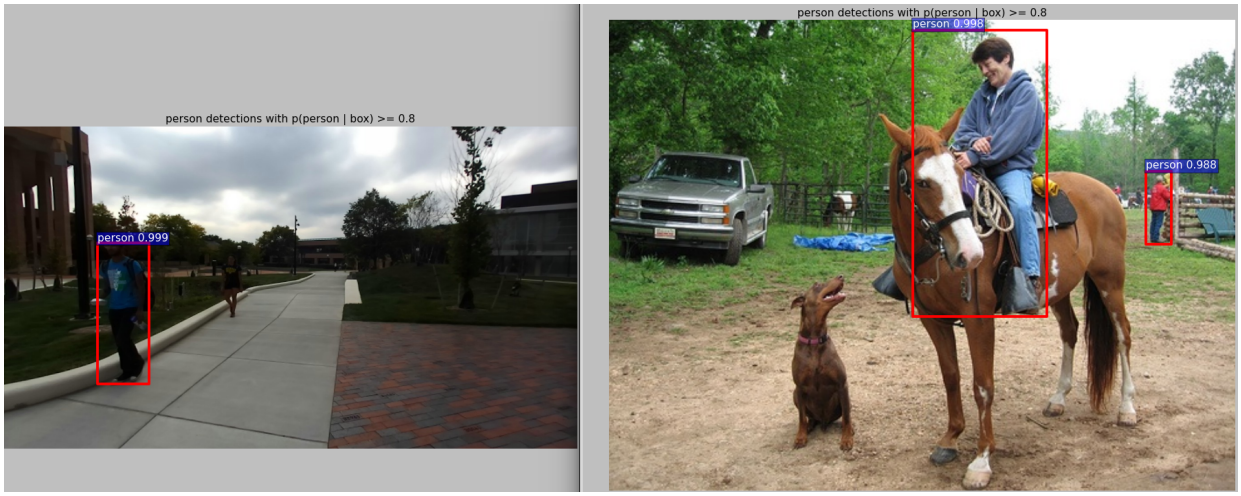
B. Multiple Object Tracking with Different Trackers

To verify the efficiency of our proposed method, we tested our algorithm on four challenging video sequences from KITTI dataset [14]. Sample frames of these four videos are shown in figure 10. These videos involve single object, multi-objects with steady camera, multi-objects with moving camera, similar multi-objects with complex motion by moving camera. The dataset provides tracks that are annotated for all objects that are sufficiently large. Due to limited time, we briefly evaluate our method by comparing Kalman Filter tracker with KCF tracker. Result shows in Table II.

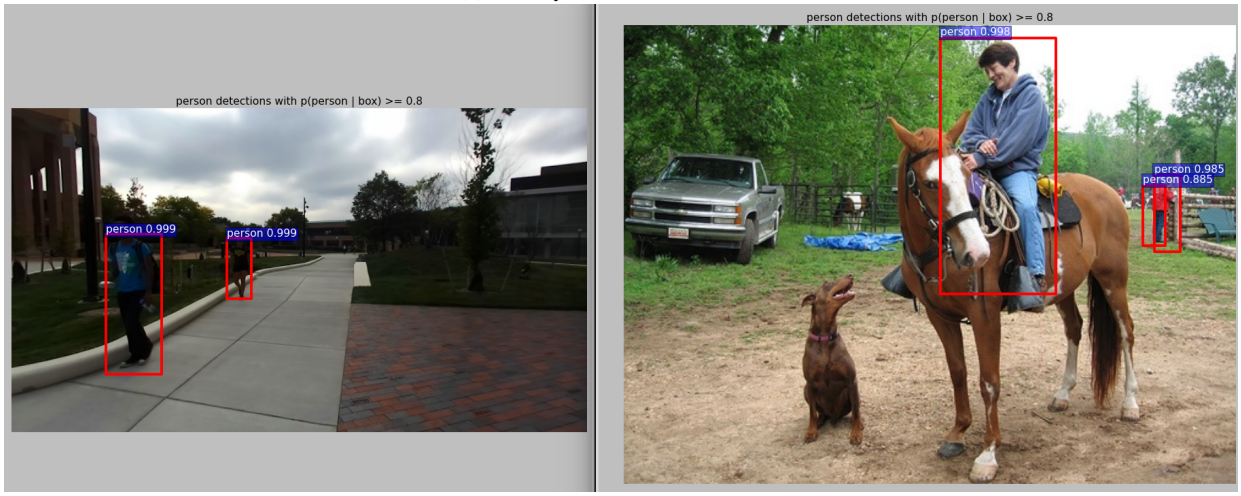
For the Boy video, both methods works very well. In this video, we noticed few false predictions which is brought by the detection. It is difficult to distinguish redundant boxes from many other detection results. Since we trust detection result more than tracker results, we assume that those over-segmentations are unavoidable in this video. Therefore, a more complex scheme would be required to handle this case.

For the Steay Camera video, our methods also behave well. Kalman filter appears more false detection,which may be limited by the linear assumption of the object speed. While KCF tracker seems more sensitive to over segmentation.

For the Bolt video, although many athletes were missing in the results. The limited behaviour is decided by the detection, which is far more reliable than other detectors(for example Background subtraction-based method). Same reasoning as Steay Camera video, Kalman filter appears more false



(a) Test by default RCNN model



(b) Test by self-trained RCNN model

Fig. 9: Faster R-CNN test on different dataset

TABLE II: Comparison of two trackers

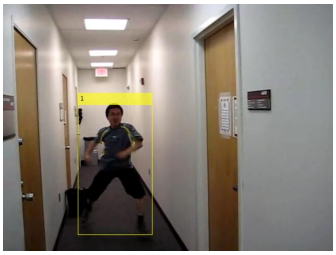
Video	Kalman Filter MOTA	KCF MOTA
Boy	Very good, only one over-segmentation	Very good, only one over-segmentation
Multi-objects with Steady Camera	Very good, with few false detection	Very good, a bit more over-segmentation
Bolt	Good, with some false prediction	Good, more false ID changes
Basketball	Fair, with more false prediction	Fair, more over-segmentation

predictions in Bolt video. We could suspect that Kalman Filter performs worse than KCF tracker when objects have unpredictable trajectories.

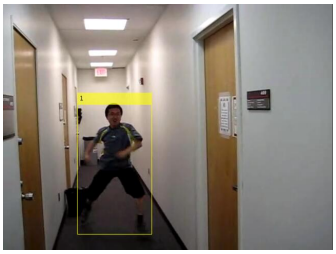
Finally, for the Basketball video, numerous objects are miss-detected in the results. Thought many objects are not detected in our method, the results are still acceptable. We noticed that the MOT(Moltiple Object Tracking) accuracy is restricted by detection accuracy, and both method has a large number of false ID changes. Thus, future works should be focused on eliminating false data associations.

V. Conclusion

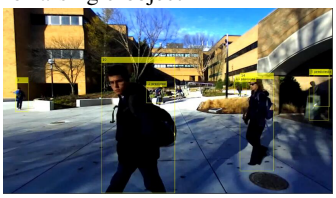
In this paper, we present a multiple object tracker that combines object trackers(Kalman Filter and KCF) with Faster R-CNN. Faster R-CNN is applied to extract pedestrian objects and get their scale and size in combination with tracker outputs, while KCF / Kalman Filter is used for data association and to handle fragmentation and occlusion problems. While we implement Faster R-CNN as our pedestrian detector, we trained Faster R-CNN model with our own data. As a result, KCF and Faster R-CNN help each



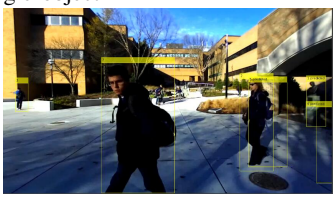
(a) Tracking with Kalman Filter for a single object



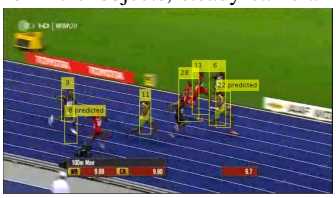
(b) Tracking with KCF for a single object



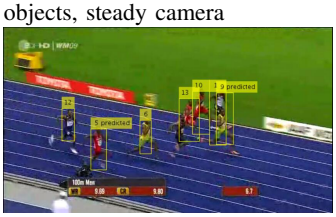
(c) Tracking with Kalman Filter for multi-objects, steady camera



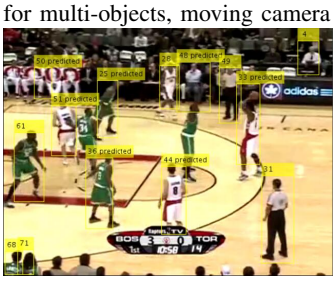
(d) Tracking with KCF for multi-objects, steady camera



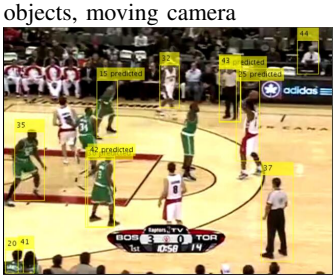
(e) Tracking with Kalman Filter for multi-objects, moving camera



(f) Tracking with KCF for multi-objects, moving camera



(g) Tracking with Kalman Filter for similar-objects with more unpredictable motion



(h) Tracking with KCF for similar-objects with more unpredictable motion

Fig. 10: Our Algorithm Test on Four Datasets

other to take tracking decision at every frame. This mixed strategy enables us to get competitive results and reduce errors under complex environment.

The advantage of our method is that it is suitable for moving camera with both multiple moving and stopping objects and it addresses the occlusion problem by tracking object individually inside a merged box. Backtracking is not necessary, neither is the use of an explicit region-based segmentation.

Future work is to promote the accuracy of data assignment. We are planning to apply Kalman filter prediction algorithm directly to a 2D SLAM map by the help of a stereo-camera. In this case, tracks of each pedestrian will be localized in world frame instead of image frame, which make prediction an easier task. To improve the existed algorithm components, we may look for a more reliable prediction/tracking algorithm or reformulate data association methods. Besides, since it is a long-term project which requires to implement the software into the hardware platform, it is necessary to qualitatively evaluate. As for a more scientific evaluation, we will apply some tools to do qualitative analysis, mainly focusing on MOTA and MOTP. MOTA is for calculating the multiple object tracking accuracy , which is evaluated based on missing rate, false positive and ID changes. MOTP is for identifying multiple object tracking precision. I t measures the average precision of instantaneous object matches. So far there is very few open source code that aims at analysis for Faster R-CNN detector so we will consider to build it ourselves when applying it to the hardware platform for safety.

Reference

[1] Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[2] Jean Carlo Mendes, Andrea Gomes Campos Bianchi, and Alvaro R Pereira Júnior. Vehicle tracking and origin-destination counting system for urban environment. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP 2015)*, 2015.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, Xiangyu Zhang, and Jian Sun. Object detection networks on convolutional feature maps. *arXiv preprint arXiv:1504.06066*, 2015.

[4] K E A van de Sande, J.R.R. Uijlings, T Gevers, and A.W.M. Smeulders. Segmentation as Selective Search for Object Recognition. In *ICCV*, 2011.

[5] Zhi Qiang Qu, Dan Tu, and Jun Lei. Online pedestrian tracking with kalman filter and random ferns. In *Applied Mechanics and Materials*, volume 536, pages 205–212. Trans Tech Publ, 2014.

[6] FA Faruqi and RC Davis. Kalman filter design for target tracking. *IEEE Transactions on Aerospace and Electronic Systems*, (4):500–508, 1980.

[7] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[8] Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg, and Joost Van de Weijer. Adaptive color attributes for real-time visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1090–1097, 2014.

[9] Liyang Yu, Chunxiao Fan, and Yue Ming. A visual tracker based on improved kernel correlation filter. In *Proceedings of the 7th International Conference on Internet Multimedia Computing and Service*, page 60. ACM, 2015.

[10] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE*

International Conference on Computer Vision, pages 1440–1448, 2015.

[11] Koen EA Van de Sande, Jasper RR Uijlings, Theo Gevers, and Arnold WM Smeulders. Segmentation as selective search for object recognition. In *2011 International Conference on Computer Vision*, pages 1879–1886. IEEE, 2011.

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[13] Pascal voc. 2007.

[14] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.