International Journal of Computer Vision and Image Processing
Volume 12 • Issue 1

Visual Tracking With Object Center Displacement and CenterNet

Merouane Labeni, University of Jijel-Mohamed-Seddik BenYahia, Algeria
https://orcid.org/0000-0001-9478-056X

Chaouki Boufenar, University of Algiers 1, Algeria

Mokhtar Taffar, University of Jijel-Mohamed-Seddik BenYahia, Algeria
https://orcid.org/0000-0002-6086-2468

ABSTRACT

Modern artificial intelligence systems have revolutionized approaches to scientific and technological challenges in a variety of fields. Thus, remarkable improvements in the quality of state-of-the-art computer vision and other techniques are observed. Object tracking in video frames is a vital field of research that provides information about objects and their trajectories. This paper presents an object tracking method basing on optical flow generated between frames and a ConvNet method. Initially, optical center displacement is employed to detect possible the bounding box center of the tracked object. Then, CenterNet is used for object position correction. Given the initial set of points (i.e., bounding box) in first frame, the tracker tries to follow the motion of center of these points by looking at its direction of change in calculated optical flow. With the next frame, a correction mechanism takes place and waits for motions that surpass a correction threshold to launch position corrections.

KEYWORDS

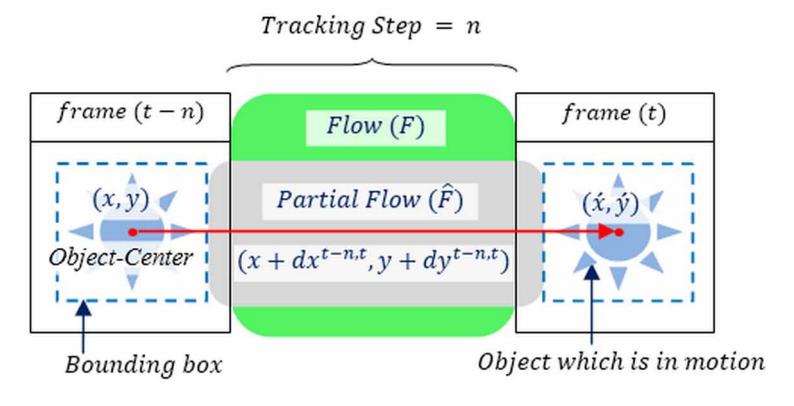
Center, Image/Video Processing, Keras, Net, OCDT, Optical Flow, Overlap Rate, Partial Flow, TensorFlow

1 INTRODUCTION

Visual tracking is an important research area in computer vision which is critical for many applications including surveillance, trafðc monitoring, video indexing, human-machine interaction, and autonomous vehicle driving. In spite of existing trackers have achieved impressive progress in the last years, designing a robust tracker is still a challenging problem. In practice, the probabilistic approaches (Kristan et al., 2008) and (Pérez et al., 2002) that globally model the tracked object's appearance, have demonstrated to be very successful. However, scenarios that contain signiðcant appearance changes caused by several factors that commonly occur in real-life scenarios, such as occlusion, scale variation, fast motion, deformation, and illumination variation present such models with serious problems. The reason is that such factors lead to reduced matches and drifting, which eventually result in the trackers' defeat. Improvements in the visual model (Babenko et al., 2011), (Kalal et al., 2010), (Bolme et al., 2010), and (Grabner et al., 2006) potentially increases the trackers' performance, but at the same time lead to additional questions regarding when should the visual

DOI: 10.4018/IJCVIP.290397

Figure 1. OCDT general process



model be improved and which parts of it should be appreciated. With the advent of high-performing object detection models (Ren et al., 2015) and (Zhou et al., 2019) a powerful alternative developed: Tracking-following-detection or tracking-by-detection (Zhou et al., 2020) and (Tang et al., 2017). Tracking-by-detection (or tracking-following-detection) influences the high power of deep-learning-based object detectors is actually the dominant tracking model. However the best object trackers are not without drawbacks.

This paper presents a method for the correction of object tracking coordinates basing on optical flow and a ConvNet method called CenterNet (Zhou et al., 2019), this last is based on the standard keypoint estimation method and stacked hourglasses network as its backbone network, like in (Law & Deng, 2018), which is trained on MS COCO datasets (Lin et al., 2014). An implementation of proposed technique has been performed using python programming language.

Figure 1 shows the general process of Object Center Displacement OCDTracker. The proposed technique consists on: Region of Interest selecting, Optical flow handling, slicing and tracking.

2 BACKGROUND INFORMATION

2.1 Optical Flow

Optical flow is the image motion of objects as the objects, scene or camera moves between two consecutive images. It is a two dimensions vector field of within-image translation (Solem, 2012).

Consider a pixel in first frame (a new dimension, time, is added), it moves by distance in next frame taken after time (Mordvintsev & Abid, 2017):

$$I\left(x,y,t\right) = I\left(x+dx, \quad y+dy, \quad t+dt\right) \tag{1}$$

OpenCV contains several optical flow implementations, the authors then use method based on (Farnebäck, 2003). That is considered one of the best methods for obtaining dense flow fields (Solem, 2012).

2.2 CenterNet

CenterNet (Zhou et al., 2019) model represents objects by one point at their bounding box center. In this model, particularities such as object size, dimension, orientation, and pose are regressed directly from image features at the center position. Objects are detected with the standard key-point

estimation method. Authors feed the input image to a FCNN that generates a heat-map. Peaks (i.e., local maxima) in this last correspond to object centers.

2.3 Evaluation Criteria

For evaluation (i.e., quantitative comparison) the authors have used both the center position error (Euclidean distance between ground-truth region center (x_{GTC}, y_{GTC}) and estimated region center (x_{ERC}, y_{ERC}), see Figure 2) and the overlap rate (overlap between ground-truth region and estimated region). The Euclidean distance metric is employed for plotting the center position error (cpe) curve which is defined as:

$$cpe = \sqrt{\left(x_{GTC} - x_{ERC}\right)^2 + \left(y_{GTC} - y_{ERC}\right)^2} \tag{2}$$

The overlap score is employed for plotting the success rate curve. By fixing an overlap threshold *s*, which is defined as the minimum overlap ratio, the authors can decide whether an overlap score is correct or not. The success ratio R is calculated as:

$$R = \sum_{i=1}^{N} \frac{u_i}{N}, \qquad u_i = \begin{cases} 1 & if\, r_i > s \\ 0 & otherwise \end{cases} \tag{3}$$

Where N is the total number of frames, r_i is the overlap score, and *s* is the corresponding threshold. A robust tracker will produce a higher value for R. The overlap score can be calculated as (Cehovin et al., 2016):

$$r_i = \frac{Area\left(GTR_i \cap ER_i\right)}{Area\left(GTR_i \cup ER_i\right)} \tag{4}$$

Where ER_i is the region estimated by a tracker in the frame, and GTR_i is the corresponding ground-truth region.

Basing on Figure 2 with help of equation (4) the authors can compute the overlap score for i^{th} case as (Cehovin et al., 2016):

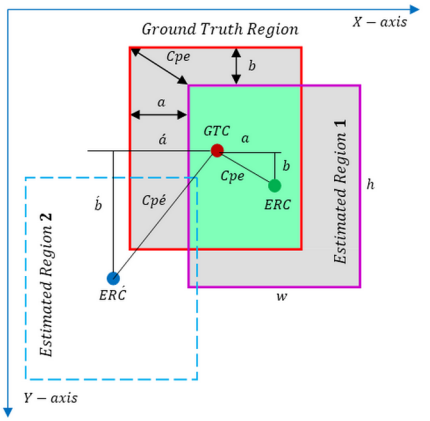
$$r_i = \frac{Area\left(green\right)}{Area\left(green + gray\right)} \tag{5}$$

$$Area\left(green\right) = \left(w - a\right) * \left(h - b\right) \tag{6}$$

Where $\begin{cases} a = \left|x_{GTC} - x_{ERC}\right| \\ b = \left|y_{GTC} - y_{ERC}\right| \end{cases}$

Accordingly if $a > w$ *or* $b > h$*: Area (green) = 0*

Figure 2. Overlap of ground-truth region with the estimated region for two different cases



$$Area\left(green + gray\right) = 2^{*}\left(w^{*}h\right) - Area\left(green\right) \tag{7}$$

3 RELATED WORKS

Several works using image/video processing and artificial intelligence methods for object tracking are mentioned these last years. Many approaches are considered to handle this challenge. First is using vigorous features. The color feature is employed by the MeanShift (Comaniciu et al., 2003) and CAMShift (Bradski, 1998) trackers, due to its robustness when there exist challenges like deformation and rotation. Both trackers are of high competency (Wong, 2014). However, when the surroundings have the same color, they easily shift from the target. To deal with this problem, Cross-Bin metric (Leichter, 2011), scale-invariant feature transform (SIFT) (Zhu, 2011), and texture feature (Bousetouane, 2013) were used into the mean shift-based tracker, and well performance was performed. The second way is to learn robust models, such as MIL (Multiple Instance Learning) (Babenko et al., 2009) and (Hu et al., 2014) and fast compressive tracking (Kaihua et al., 2014). As well of these trackers, the local-global tracker (LGT) (Cehovin et al., 2013), incremental visual tracker (IVT) (Ross et al., 2008) and TLD (Tracking Learning Detection) (Kalal et al., 2010) also perform nearly well.

These first trackers were fast, simple and reasonably robust. However, they were unsuccessful with the absence of well-based low-level features such as corners and edges. In another approach with the arrival of high performing objects detection models (Felzenszwalb et al., 2009) and (Ren et al., 2015), a great alternative emerged: tracking by detection (Bewley et al., 2016), (Tang et al., 2017) and (Xu et al., 2019). These models based on a given precise recognition to identify objects and then track them through time. Furthermore, the KCF (Henriques et al., 2015) that represents an improved version of CSK (Henriques et al., 2012), robust MIL tracker (Babenko et al., 2011), an improved version of (Babenko et al., 2009), uses a tracking by detection approach that shows well robustness to inaccuracies of the tracker and to incorrectly labeled training patterns.

A recent approach is joint detection and tracking to perform multi-object tracking by converting existing detectors into trackers by integrating both goals detection and tracking in the same framework. Authors in (Feichtenhofer et al., 2017) use a Siamese ConvNet that predicts the location in the second frame of the object shown in the center of the previous frame. In (Zhang et al., 2018) authors use tracked bounding boxes as supplementary region proposals to improve detection with a bipartite-matching-based bounding box allocation. In object detection system, (Kang et al., 2017) feed stacked

consecutive frames into the network and do object detection for a whole video sequences. Tracktor (Bergmann et al., 2019) removes the box association by directly propagating identities of region proposals using bounding box regression. And (Zhu et al., 2017) use ñow to warp intermediate features from previous frames to accelerate inference. In (Zhou et al., 2020) authors present a point-based framework where ach object is represented by a single point at the center of its bounding box. This center point is then tracked through time basing on CenterNet (Zhou et al., 2019) detector.

4 PROPOSED TECHNIQUE

In this paper authors address visual tracking as the problem of object center displacement across frames, at time t, an image of the current frame is represented by $I^t \in \mathbb{R}^{Width \times Height \times Channels}$ and the previous frame by $I^{t-n} \in \mathbb{R}^{Width \times Height \times Channels}$ where channels =1 for gray scale images and 3 for color images. The tracked objects in the previous frame are represented by $T^{t-n} = \left\{ Ob_i^{t-n} \left(p_i^{t-n} \left(x_i^{t-n}, y_i^{t-n} \right), s_i^{t-n}, v_i^{t-2n,t-n} \right) \right\}_{i=0}^{M-1}$, where M is the number of tracked objects, each object Ob_i^t, in current frame, is represented by a tuple $\left(p_i^t, s_i^t, v_i^{t-n,t} \right)$ where $p_i^t \left(x_i^t, y_i^t \right) \in \mathbb{R}^2$ represents its center location, $s_i^t \left(w_i^t, h_i^t \right) \in \mathbb{R}^2$ represents the width and height of its bounding box, and $v_i^{t-n,\,t} = \hat{F}\left(p_i^{t-n} \right)$, $v_i^{t-n,\,t} \in \mathbb{R}^2$ represents its center motion (used to perform an eventual object center displacement and calculate the Root Mean Squared Error to activate/deactivate acceleration mechanism, see Figure 4) within extracted partial flow $\hat{F} \subset F \in \mathbb{R}^{Width \times Height \times 2}$, where F represents optical flow calculated between the previous frame $t-n$ and the current one t according to the tracking step $Ts = n$ (see Figures 3 and 4). Note that motion vector $v_i^{t-n,\,t}$ is null at the first frame. There are two main challenges here. The ðrst is tracking the targeted objects across frames, this challenge is addressed by using the motion vector $v_i^{t-n,\,t} = \left(dx_i^{t-n,\,t}, dy_i^{t-n,\,t} \right)$ where n represents the tracking step (see Figures 1 and 4) thus the new position for a tracked object Ob_i^{t-n} in current frame t is:

$$Ob_i^t = \left(p_i^t \left(x_i^t, y_i^t \right),\ s_i^t, v_i^{t-n,t} \right) \tag{8}$$

$$\text{Where} \begin{cases} x_i^t = x_i^{t-n} + dx_i^{t-n,t} \\ y_i^t = y_i^{t-n} + dy_i^{t-n,t} \\ s_i^t = s_i^{t-n} \\ v_i^{t-n,t} = \hat{F}\left(p_i^{t-n} \right) \end{cases}$$

The second challenge is preventing a tracked object Ob_i^t to escape (i.e., by producing a fast motion) from its bounding box by adopting a correction mechanism, this last is associated to a sensory zone $Z = \left\{ \left(dx_l^{t-n,\,t}, dy_l^{t-n,\,t} \right),\ l \in \left(x_2^t - x_1^t \right) \times \left(y_2^t - y_1^t \right) \right\} \subset \hat{F}_i$. The absolute value function is applied for each element of each motion vector (represented by both $dx_i^{t-n,\,t}$ in X-axis and $dy_i^{t-n,\,t}$ in Y-axis) in Z and they are both compared with a correction threshold (fixed to 30 pixels) if one of them surpasses this threshold a correction of object coordinates is required and CenterNet (Zhou et al., 2019) detector is launched. CenterNet takes a single image $I \in \mathbb{R}^{Width \times Height \times 3}$ as input and produces a set of detections $D = \left\{ d_k^t \left(p_k^t,\ s_k^t \right) \right\}_{k=0}^{N-1}$ for each class $c \in \{1, .., C\}$ at frame t. Thus each detected

object Ob_k^t is identified through its center point $p_k^t \in \mathbb{R}^2$ and then regressed to a height and width $s_k^t \in \mathbb{R}^2$ of the object's bounding box. CenterNet produces a low-resolution heatmap $\hat{Y} \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$ and a size map $\hat{S} \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$ with a down-sampling factor $R = 4$. Each peak (also called local maximum) $\hat{c} \in \mathbb{R}^2$ in the heatmap $\hat{Y}$ corresponds to a center location of a detected object with conðdence $\hat{w} = \hat{Y}_{\hat{c}}$ and object size $\hat{s} = \hat{S}_{\hat{c}}$. The set of detections D produced by CenterNet is used to perform object position correction, in this case the new position for a tracked object Ob_i^{t-n} in t^{th} frame is calculated as follows:

$$Ob_i^t = \left(p_i^t,\ s_i^t, v_i^{t-n,t}\right) \tag{9}$$

$$Where \begin{cases} p_i^t = p_k^t \in d_k^t \\ s_i^t = s_k^t; \\ v_i^{t-n,t} = \hat{F}(p_k^{t-n}) \end{cases}$$

Where And $r_k = \dfrac{area\left(Ob_i^{t-n} \cap d_k^t\right)}{area\left(Ob_i^{t-n} \cup d_k^t\right)}$ represents the overlap score (calculated using equation (4)) between the i^{th} tracked object's bounding box, at previous frame $t-n$, represented by Ob_i^{t-n} and the one of each detect object, at current frame t, represented by d_k^t.

The tracking step is calculated as follows (see figure 4):

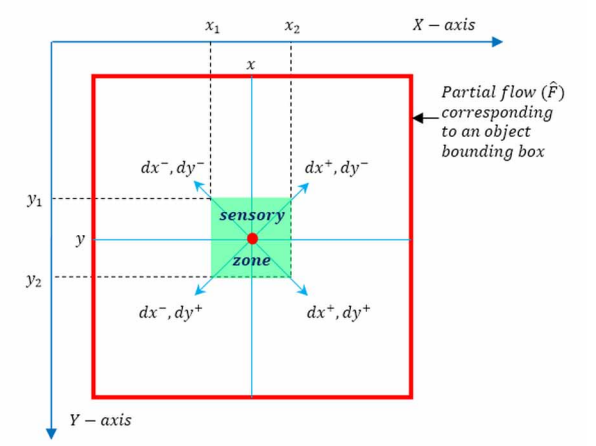
$$Ts = \begin{cases} n+=1 & if \max\left\{RMSE\left(\hat{F}\left(p_i^t\right), \hat{F}\left(\hat{p}_j^t\right)\right)\right\}_{j=1}^{K} < 1 \\ 1 & otherwise \end{cases} \tag{10}$$

Where n is the previous value of TS incremented by 1, $\hat{p}_j^t$ represents the j^{th} point that belongs to the object bounding box border at current frame t, and K represents the number of these points.

Thus the methodology proposed in this paper uses a new algorithm for "object tracking" based on optical flow generated between frames (for estimating the direction and speed of moving objects) using the Farneback optical flow (Farnebäck, 2003) method and CenterNet (Zhou et al., 2019) ConvNet for object position correction. The method includes these steps: Bounding boxes selecting using a RoI function, optical flow computing according to the tracking step value and object tracking by bounding box center displacement according to computed optical flow or a corrected position using CenterNet (Zhou et al., 2019) detector. The detailed process of the proposed method is represented in Algorithm 1 and Algorithm 2.

For simplicity authors adopt a single-object tracking scenario, the objective in this case is to track the object $Ob_{i=0}^{t-n}$ (represented by *Initial_bounding_box* variable in the Algorithm 1) through frames. For each optical flow generated according to the tracking step the authors extract partial flow (see Figure 3) corresponding to object bounding box boundaries using slicing and with help of sensory zone which represents its kernel the authors control object displacement.

Figure 3. Object potential motions



In each time the authors capture the maximum motion in sensory zone (green zone in Figure 3, initialized to 20x20 pixels^2) and according to correction threshold the authors decide the displacement type (using only object center motion (dx, dy) or a totally displacement to a new detected position using CenterNet (Zhou et al., 2019) detector). The steps of this process and its detail are mentioned in Algorithm 1.

Algorithm 1 describes the main steps for OCDT tracking process, lines 1 and 2 do initializations, line 3 opens a loop through video frames, lines 4-6 allow to read frames according to tracking step value, line 7 and 8 allow to calculate optical flow between i^{th} frame and $(i+Ts)^{th}$ frame, line 9 extracts the partial flow corresponding to object bounding box, note that dealing with object partial flow is faster than the whole flow during tracking, line 10 recuperates the bounding box center coordinates, line 11 gets bounding box center motion within flow vector, lines 12-15 compute RMSE (Root Mean Squared Error) between motion of each point belongs to bounding box border (red rectangle in Figure 3) and object center's motion, line 16 gets the maximum RMSE that is used (in lines 17-21) to accelerate or slow down the tracking process (see Figure 4), lines 22 recuperates sensory zone coordinates (i.e., its top-left corner (x_1, y_1) and bottom-right corner (x_2, y_2), see Figure 3), line 23 extracts the maximum motion within sliced zone (i.e., sensory zone) using *maxzone* variable, note that the *abs* function calculates the absolute value for each motion within sensory zone, in lines 24-28 and according to the maximum motion extracted above and using correction threshold (assigned to 30 pixels) the authors decide if it concerns of a position correction using OPC module or simply a position shift using $\left(dx^{\pm}, dy^{\pm}\right)$ motions (see Figure 3), line 29 displays new bounding box and line 30 reinitializes the bounding box for next frame.

Algorithm 2 describes OPC (Object Position Correction) process, line 1 calls CenterNet (Zhou et al., 2019) detector with the current frame (i.e., frame of which a fast motion is detected) as parameter, the result is a set of detected objects (i.e., a set of detected bounding boxes represented by *detected_Bboxes* variable) within current frame using a free detection (i.e., the detection confidence equals zero) with a restriction on their bounding boxes areas (i.e., the authors only take objects that have at maximum double of tracked object area) that allows to eliminate large bounding boxes with respect to tracked object bounding box, the authors also eliminate imbricated bounding boxes for the same object, line 2 opens a loop through the whole detected bounding boxes, line 3 computes, using equation (5), and saves overlap rate between each detected bounding box and the one of tracked object, in this process the authors assume, according to equations (6) and (7), that estimated region ER_i and ground-truth region GTR_i (see Figure 2) have the same width and height for that and to compute the overlap

Algorithm 1: OCDTracker

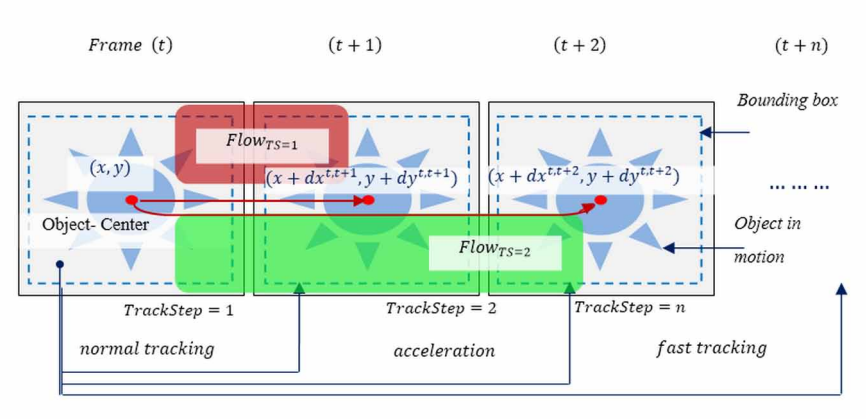
Input: Initial_bounding_box, old_frame, corr_threshold
1: old_Bbox = initial_bounding_box
2: tracking_step=1
3: **while** true {video is open}:
4: **for** i =1 **to** tracking_step **do**
5: current_frame=read(frame)
6: **end for**
7: calculate flow(old_frame, current_frame)
8: old_frame = current_frame
9: partial_flow=flow(old_Bbox)
10: (x, y) = center (old_Bbox)
11: dx, dy = partial_flow[x, y]
12: **for** $\hat{p} \in$ old_Bbox_borders **do**
13: dx', dy' =partial_flow[$x_{\hat{p}}, y_{\hat{p}}$]
14: Calculate and save RMSE(dx, dy, dx', dy')
15: **end for**
16: max_rmse = max(savedRMSEs)
17: **if** max_rmse $<$ 1 **then** {acceleration}
18: tracking_step= tracking_step+1
19: **else** {slowing down}
20: tracking_step = 1
21: **end if**
22: (x_1, y_1, x_2, y_2) = sensory_zone_coordinates
23: maxzone = max(abs(partial_flow[x_1: x_2, y_1: y_2]))
24: **if** maxzone $>$ corr_threshold **then**
25: new_Bbox = OPC(old_Bbox, current_frame)
26: **else**
27: new_Bbox = moving old_Bbox center with dx, dy values
28: **end if**
29: display(new_Bbox)
30: old_Bbox=new_Bbox
31: **end while**

Algorithm 2: OPC (Object Position Correction)

Input: old_Bbox, current_frame
1: detected_Bboxes= CenterNet(current_frame)
2: **for** each Bbox **in** detected_Bboxes **do**
3: calculate and save overlap_rate(old_Bbox, Bbox)
4: **end for**
5: x, y=center(Bbox having max_overlap_rate)
Output: Moving old_Bbox center to (x,y) point.

rate the authors use for each detected bounding box its width and height for both ER_i and GTR_i, this last allows to appreciate overlap rate for tracked object and ignores other detected objects located in its neighborhood, line 5 recuperates the center coordinates of bounding box having the great overlap rate, the output of this algorithm is moving the tracked object bounding box to its new detected position. In an object multi-tracking scenario users can open a loop through tracked objects within OCDT algorithm and for each object they decide the displacement type according to correction threshold, in this case optical flow is calculated (for each doublet of frames) for all objects and CentreNet (Zhou et al., 2019) is called once for all corrections if there are in a current frame.

Figure 4. OCDT acceleration mechanism



5 EXPERIMENTAL RESULTS

The authors have measured the performance of OCDT tracker on several challenging videos[1]. The videos include either a non-static object or an object that suffers significant form changes. OCDT tracker was implemented in Python and runs at approximately 03 frames per second (i.e., CenterNet (Zhou et al., 2019) consumption time is ignored), on an Intel(R) Core (TM) i5-2430M, using Keras API on Tensorflow (Abadi et al., 2015) CPU framework. Note that Tensorflow (Abadi et al., 2015) GPU framework represents the best execution environment for models based on ConvNets architectures.

For comparison, the authors compute the center position error between the manually labeled ground-truth region center (performed by picking objects centers points through frames using a RoI function and saving, for each frame, its two first returned results that correspond to object center coordinates) and estimated region center by each tracker. The basic properties of the experimental sequences are collected in Table 1:

The authors also recorded the tracked objects trajectories. The experiments implicated tracking a car, a bird and objects with significant view changes (see Figure 5).

The authors have compared the OCDT Tracker with six state-of-the-art trackers, which are:

1. The Kernelized Correlation Filter tracker (KCF) (Henriques et al., 2015).
2. The multiple instance tracker (MIL) (Babenko et al., 2011).
3. The P-N tracker (TLD) (Kalal et al., 2010).
4. The CSRT tracker (Lukezic et al., 2017).
5. The Minimum Output Sum of Squared Errors (MOSSE) tracker (Bolme et al., 2010).
6. The OAB tracker (Grabner et al., 2006).

The success rate was automatically determined by measuring the overlap, using equation (5), between the ground-truth region GTR_i and the region estimated by each tracker ER_i in each frame. In the first frame, each tracker is manually initialized by drawing a bounding box over the object.

As shown in Figure 6, results for the *car* sequence (camera follows tracked object), all the trackers perform well except TLD tracker. However, the bounding box calculated by the CSRT tracker has a relatively small scale in frames #235 #330 and #475, the same remark about OCDT tracker from 360th frame, the OAB tracker produces a significant error position for the bounding box from 235th frame. The TLD tracker failed to track the car from 106th frame. As seen in Figure 7b the OCDT tracker, with two unnecessary position corrections (i.e., activated by the car's windscreen wiper after

Figure 5. Samples from the experimental sequences

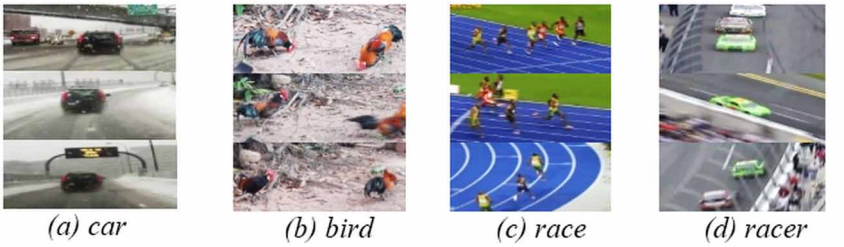


Table 1. An outline of the experimental sequences

Sequence	Type	Remarks	Ground-truth labeling step	frames
(a) car	car	Illumination variation, scale variation	one by five frames	500
(b) bird	body	Fast motion, occlusion, appearance change	variable value according to the tracking step TS	300
(c) race	body	Fast motion, scale variation, appearance change	one by three frames	339
(d) racer	car	Fast motion, appearance change, occlusion	one by frame	176

a long acceleration period) at frames #129 and #360 has a good success ratio when different overlap thresholds are selected and a more accurate center has been produced by the OCDT tracker as seen in Figures 6 and 7a.

Figure 6. Performance on the car sequence. The results are shown for all trackers with OCDT corrections.

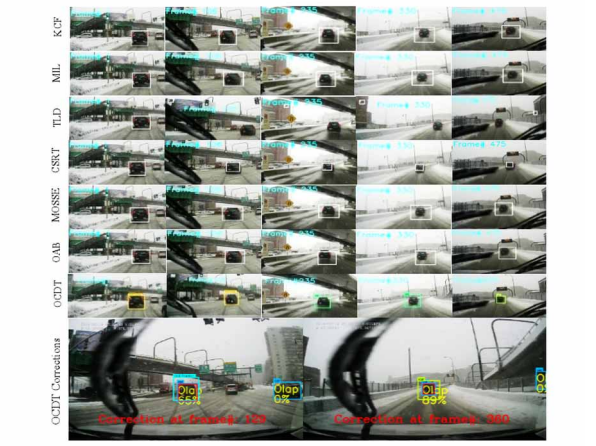


Figure 6. Performance on the car sequence. The results are shown for all trackers with OCDT corrections.

Figure 7. The quantitative plots for all trackers of the car sequence: (a) Center position error plots; (b) Success rate plots.

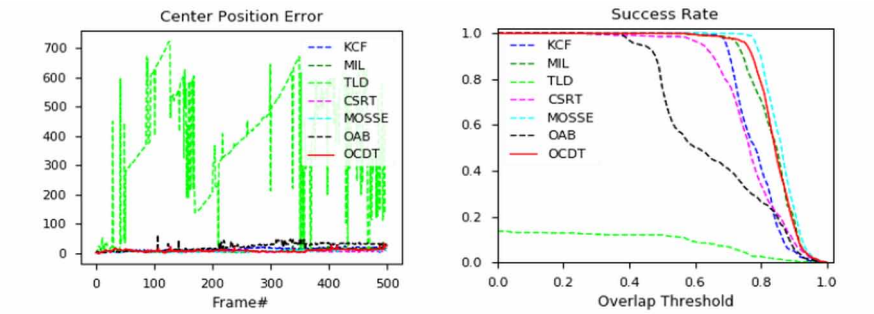


Figure 7. The quantitative plots for all trackers of the car sequence: (a) Center position error plots; (b) Success rate plots.

In Figure 8, Results for the *bird* sequence, in this sequence there is a short speed motion of tracked object at frames #45 #46 and #47 and camera speed shift from frame #107 to #117, all the trackers perform bad except CSRT and OCDT trackers and for lower degree MIL tracker. As shown in figures 9a and 9b, the OCDT tracker has a good success ratio and the center position error for whole frames is reasonable. Due to the OCDT correction module (with three corrections at frames #46 #108 and #117) OCDT has a more appropriate scale than the other trackers and the center location of the tracked object is closer to the ground truth center in most frames.

Figure 8. Performance on the bird sequence. The results are shown for all trackers with OCDT corrections.

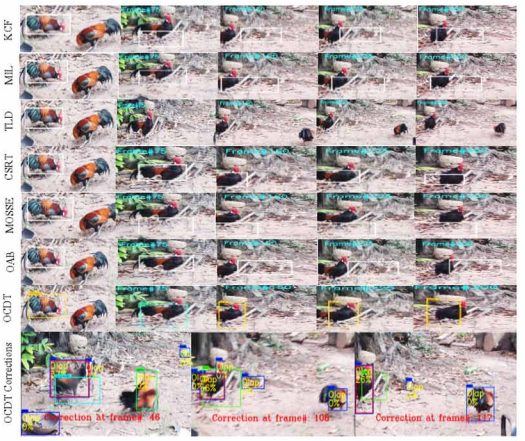


Figure 8. Performance on the bird sequence. The results are shown for all trackers with OCDT corrections.

Figure 9. The quantitative plots for all trackers of the bird sequence: (a) Center position error plots; (b) Success rate plots.

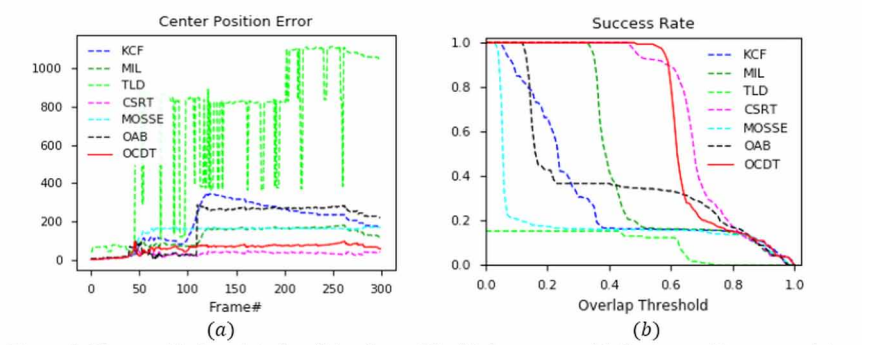


Figure 9. The quantitative plots for all trackers of the bird sequence: (a) Center position error plots; (b) Success rate plots.

As shown in Figures 10 and 11b, results for the *race* sequence, KCF, MIL, and CSRT work better than OCDT tracker, when estimating the success rate evaluation criteria, this is because of correction threshold assigned to 30 pixels, for a better execution of OCDT in this sequence the correction threshold should be decreased (e.g., 20 pixels, which increases the number of corrections up to 12 corrections). Note that decreasing correction threshold to zero pixel requires a correction in each frame. Other trackers have weaker results in object tracking from frame #70, due to the fast motion and important object form changes. The OCDT position correction module succeeded to track the athlete with only three object position corrections at frames #66, #190 and #257.

Figure 10. Performance on the race sequence. The results are shown for all trackers with OCDT corrections.

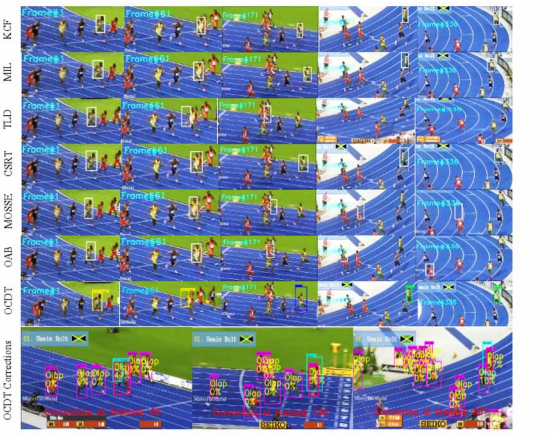
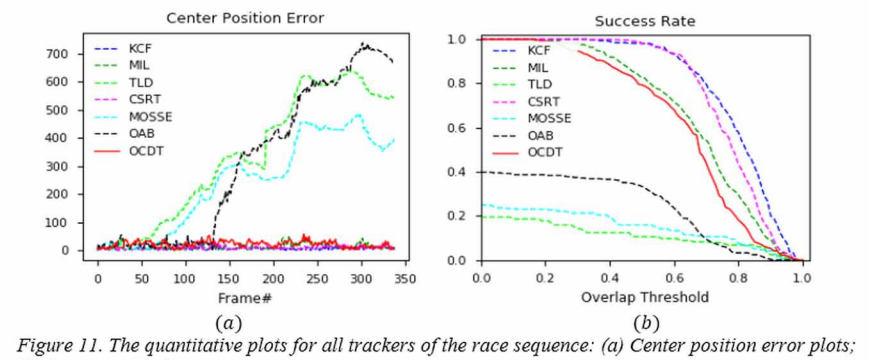


Figure 10. Performance on the race sequence. The results are shown for all trackers with OCDT corrections.

Figure 11. The quantitative plots for all trackers of the race sequence: (a) Center position error plots; (b) Success rate plots.



(a) (b)

Figure 11. The quantitative plots for all trackers of the race sequence: (a) Center position error plots; (b) Success rate plots.

Figure 12. Performance on the racer sequence. The results are shown for all trackers with OCDT corrections.

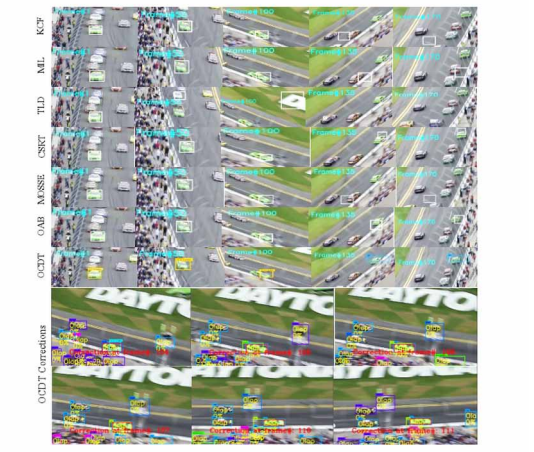


Figure 12. Performance on the racer sequence. The results are shown for all trackers with OCDT corrections.

As shown in Figure 12, results for the *racer* sequence, OCDT, OAB, and CSRT trackers perform well, other trackers have weaker results from frame #105, due to the challenge of serious appearance change and fast motion. The TLD tracker failed to track the racer-car from frame #75. As shown in Figures 13a and 13b, OCDT tracker has minimal position errors (with six corrections, at frames #104, #105, #106, #107, #110, and #111) also it has a good success ratio.

Figure 13. The quantitative plots for all trackers of the racer sequence: (a) Center position error plots; (b) Success rate plots.

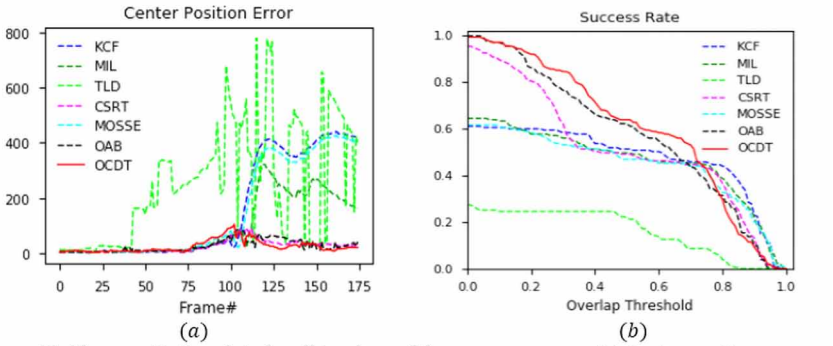


Figure 13. The quantitative plots for all trackers of the racer sequence: (a) Center position error plots; (b) Success rate plots.

In summary OCDT tracker is good for tracking objects in most cases with help of sensory zone and CenterNet (Zhou et al., 2019) detector except in the case where tracked object disappears and reappears through frames. For real-time tracking users should use Tensorflow (Abadi et al., 2015) GPU framework, in this case users can use seamlessly CenterNet (Zhou et al., 2019) or CornerNet (Law & Deng, 2018) in object position correction process.

6 CONCLUSION

In this paper, authors have formulated the visual tracking as object center displacement based on optical flow vector and CenterNet (Zhou et al., 2019) method for the correction of object position coordinates, which is robust to the appearance variation caused by important form changes, deformation, and rapid motion. Two algorithms are presented for the proposed formulation and authors further give detail on the working steps of them. Authors also include a detailed experimental analysis and perform the comparative experiments using four video sequences and six reference trackers as well as several figures are mentioned that illustrate the properties of the proposed visual model.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., . . . Zheng, X. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. http://www.tensorñow.org/

Babenko, B., Yang, M. H., & Belongie, S. (2009). Visual tracking with online multiple instance learning. *Proceedings of the IEEE Conference on Computer Vision Pattern Recognition (CVPR)*, 983–990. doi:10.1109/CVPR.2009.5206737

Babenko, B., Yang, M. H., & Belongie, S. (2011). Robust object tracking with online multiple instance learning. TPAMI, 33(8), 1619–1632. doi:10.1109/TPAMI.2010.226

Bergmann, P., Meinhardt, T., & Leal-Taixe, L. (2019). Tracking without bells and whistles. ICCV. doi:10.1109/ICCV.2019.00103

Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple online and real time tracking. ICIP.

Bolme, D. S., Beveridge, J. R., Draper, B. A., & Lui, Y. M. (2010). Visual object tracking using adaptive correlation filters. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi:10.1109/CVPR.2010.5539960

Bousetouane, F., Dib, L., & Snoussi, H. (2013). Improved mean shift integrating texture and color features for robust real time object tracking. *The Visual Computer*, *29*(3), 155–170. doi:10.1007/s00371-012-0677-0

Bradski, G. R. (1998). Real time face and object tracking as a component of a perceptual user interface. *Proceedings of the Fourth IEEE Workshop on Applications of Computer Vision*, 214–219. doi:10.1109/ACV.1998.732882

Cehovin, L., Kristan, M., & Leonardis, A. (2013). Robust visual tracking using an adaptive coupled-layer visual model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(4), 941–953. doi:10.1109/TPAMI.2012.145 PMID:22802114

Cehovin, L., Leonardis, A., & Kristan, M. (2016). *Visual object tracking performance measures revisited.* arXiv: 1502.05803v3 [cs.CV].

Comaniciu, D., Ramesh, V., & Meer, P. (2003). Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(5), 564–577. doi:10.1109/TPAMI.2003.1195991

Farnebäck, G. (2003). Two-frame motion estimation based on polynomial expansion. *Proceedings of the 13th Scandinavian Conference on Image Analysis*, 363–370. doi:10.1007/3-540-45103-X_50

Feichtenhofer, C., Pinz, A., & Zisserman, A. (2017). Detect to track and track to detect. ICCV. doi:10.1109/ICCV.2017.330

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2009). Object detection with discriminatively trained part-based models. TPAMI.

Grabner, H., Grabner, M., & Bischof, H. (2006). Real-time tracking via on-line boosting. BMVC, 1, 6. doi:10.5244/C.20.6

Henriques, J. F., Caseiro, R., Martins, P., & Batista, J. (2015). High-speed tracking with kernelized correlation ðlters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(3), 583–596. doi:10.1109/TPAMI.2014.2345390 PMID:26353263

Henriques, J. F., Caseiro, R., Martins, P., & Batista, J. (2012). Exploiting the circulant structure of tracking by detection with kernels. ECCV, 702–715. doi:10.1007/978-3-642-33765-9_50

Hu, K., Zhang, X., Gu, Y., & Wang, Y. (2014). Fusing target information from multiple views for robust visual tracking. *IET Computer Vision*, *8*(2), 86–97. doi:10.1049/iet-cvi.2013.0026

Kaihua, Z., Lei, Z., & Yang, M. H. (2014). Fast compressive tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(10), 2002–2015. doi:10.1109/TPAMI.2014.2315808 PMID:26352631

Kalal, Z., Matas, J., & Mikolajczyk, K. (2010). P-N learning: Bootstrapping binary classifiers by structural constraints. CVPR, 49–56.

Kang, K., Li, H., Xiao, T., Ouyang, W., Yan, J., Liu, X., & Wang, X. (2017). Object detection in videos with tubelet proposal networks. CVPR. doi:10.1109/CVPR.2017.101

KristanM., Per

Kristan, M., Perš, J., & Kova, čič, S., & Leonardis, A. (2008). A local motion-based probabilistic model for visual tracking. *Pattern Recognition*.

Law, H., & Deng, J. (2018). CornerNet: Detecting objects as paired key points. ECCV.

Leichter, I. (2011). Mean shift trackers with cross-bin metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(4), 695–706. doi:10.1109/TPAMI.2011.167 PMID:21844621

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. ECCV.

Lukezic, A., Vojir, T., Zajc, L.-C., Matas, J., & Kristan, M. (2017). Discriminative correlation filter with channel and spatial reliability. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6309-6318. doi:10.1109/CVPR.2017.515

Mordvintsev, A., & Abid, K. (2017). *OpenCV-Python*. Tutorials Documentation, Release 1.

Pérez, P., Hue, C., Vermaak, J., & Gangnet, M. (2002). Color-Based probabilistic tracking. In *ECCV* (Vol. 1, pp. 661–675). Springer-Verlag.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. NIPS.

Ross, D., Lim, J., Lin, R. S., & Yang, M. H. (2008). Incremental learning for robust visual tracking. *International Journal of Computer Vision*, *77*(1-3), 125–141. doi:10.1007/s11263-007-0075-7

Solem, J.-E. (2012). *Programming Computer Vision with Python*. Creative Commons.

Tang, S., Andriluka, M., Andres, B., & Schiele, B. (2017). Multiple people tracking by lifted multi-cut and person re-identification. CVPR.

Wong, S. (2014). *An fpga Implementation of the Mean-Shift Algorithm for Object Tracking*. AUT University.

Xu, J., Cao, Y., Zhang, Z., & Hu, H. (2019). Spatial-temporal relation networks for multi-object tracking. ICCV. doi:10.1109/ICCV.2019.00409

Zhang, Z., Cheng, D., Zhu, X., Lin, S., & Dai, J. (2018). *Integrated object detection and tracking with tracklet-conditioned detection.* arXiv:1811.11167.

Zhou, X., Koltun, V., & Kr a henb u hl, P. (2020). *Tracking Objects as Points.* arXiv:2004.01177v1 [cs.CV].

Zhu, C. (2011). *Video Object Tracking Using Sift and Mean Shift* (Master's Thesis). Chalmers University of Technology, Gothenburg, Sweden.

Zhu, X., Wang, Y., Dai, J., Yuan, L., & Wei, Y. (2017). Flow-guided feature aggregation for video object detection. ICCV. doi:10.1109/ICCV.2017.52

ENDNOTE

[1] The experimental sequences and experiments data are available at: https://drive.google.com/drive/folders/1Pa22zSw8uyO36FWVrqA5dLPS_VYi5_a-