A Comparison of OpenCV Algorithms for Human Tracking with a Moving Perspective Camera

Olfa Haggui, Matossouwé Agninoube Tchalim, Baptiste Magnier

A Comparison of OpenCV Algorithms for Human Tracking with a Moving Perspective Camera

Olfa HAGGUI, Matossouwé AGNINOUBE TCHALIM and Baptiste MAGNIER
EuroMov Digital Health in Motion, Université de Montpellier, IMT Mines Alès, Alès, France
{Olfa.Haggui, Baptiste.Magnier}@mines-ales.fr, Agninoubetchalim@gmail.com

***Abstract*—Visual tracking has received much attention in recent years, especially pedestrian tracking. People tracking represents an important computer vision problem with numerous real-world applications. While significant progress has been achieved for human tracking and detection, trackers are still prone to failures and inaccuracies to master all difficult situations that may arise during the process: changes in appearance, illumination, occlusions, camera movement or cluttered background. To overcome these limitations, tracking algorithms offered by the OpenCV software library are evaluated through this paper. These trackers are fast and easy to use. However, pedestrians are particularly difficult to track with a moving camera. This paper brings a benchmark of human tracking algorithms implementations using moving camera. Here, we propose a qualitative and quantitative assessment followed by a comparison with a particle filter algorithm based on histograms of both color and texture features. Finally, in order to compare to new developed tracking algorithms in the framework of a pedestrian tracking accuracy in an unknown environment, experiments with a new available dataset validate either the reliability of OpenCV trackers or an easy-to-use particle filter.**

***Index Terms*—Human tracking, OpenCV, particle filter.**

I. Introduction and Motivations

Visual object tracking is one of the most active research topics in computer vision. In this respect, commercial development as well as academic research contribute intensively in this area. Consequently, many visual trackers have been proposed [1][2] [3]. A specific issue concerns real time human tracking which represents a predominant and challenging task. The main goal is to automatically identify a human shape and localize him/her in each frame of a video sequence [4][5][6]. It has attracted researchers interest for decades an immense variety of applications such as robotics, smart video surveillance, patient monitoring, accident prediction, etc. Moving object may be detected by frame difference methods [7]. In order to track a person in a long-term sequence of a real-world scenarios, a number of issues need to be resolved, and almost every video is a special case, especially when the camera is moving because the background is constantly changing (excepted background composed by large homogeneous regions).

In this context, the tracking reliability depends of the situation of video acquisition: degree of camera movement and level of background change. This gets worse when, in the meantime, the person appearance changes in color or shape (because of shadow, displacements or movements), making the reference image irrelevant. Elsewhere, the clutter of the background may provide a lot of parasitic information disturbing the detection, as well as the presence of other moving objects, new object(s) in the scene and/or partial/total occlusions. Moreover, for optimal performance, the tracking algorithm [1][8] needs to run in real time, or by default, with a maximum of frames per second, depending on the desired application. Typically, these algorithms estimate the tracking of the target in a long sequence of images by trying to detect the target in each image, frame by frame.

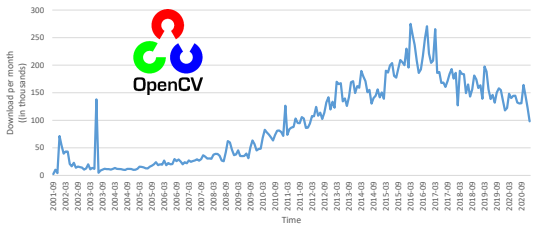


Fig. 1. OpenCV logo and timeline statistics, between years of 2001 and 2020. Source: https://sourceforge.net/projects/opencvlibrary/files/stats/timeline

The issue of analyzing and interpreting images and videos is a real challenge. Several programming libraries mainly intended for computer vision, some with real-time calculations, have been created. Hence, for its performance in computing time, the utilization of this library with minimal restrictions and also its large number of available free functions, OpenCV[1] is the most used: more than 100K download per month, since 10 years (see download statistics in Fig. 1). Basically, this library of programming functions mainly aims at real-time video and image processing and analysis. As a consequence, to obtain good execution time of an OpenCV program, the main language is C++ [9], but it supports a wide variety of other programming languages, with binding for Python, Java, Matlab, etc. Moreover, OpenCV libraries are designed for a way allowing to take advantage of hardware acceleration and expansion of multi-core systems. Also, OpenCV version 4.2.0 offers access to over 2,500 algorithms to be used for deployment of various machine learning and computer vision capabilities. Besides, it contains a tracker class, with several different trackers based on different tracking algorithms, such as TLD, Boosting, KCF, CSRT trackers. They all have their own pros and cons. Some run fast but not as accurate as others. Alternatively, some are accurate and more efficient but may run slowly. These OpenCV trackers are presented through this paper, with their computational time and reporting their quantitative and qualitative results on a database of videos containing human(s) walking acquired with a moving camera. In this framework of human tracking with moving

[1] https://opencv.org/about/

camera, particle filter [10] [11] tracking algorithm can take into account the target and background appearance variations. Moreover, modelling a human appearance is difficult due to various issues such as variation in pose, clothing, scale change, illumination, and sometimes, partial/total occlusion. For the tracking process, particle filters may be based on color histograms, texture analysis or other features extraction in order to estimate the position of the desired target in several frames over the long term. This type of algorithm seems more reliable in the context of camera in movement, this study will also justify how to select the original target for a robust tracking in difficult conditions of video acquisition.

In this paper, the effectiveness of OpenCV trackers are reported in the framework of human tracking with a camera in movement. Thus, this study allows to justify whether it is necessary to compare new trackers with those implemented in the OpenCV library, or else, to use a straightforward particle filter to perform this comparison. In the next Section, the tracking methods implemented in OpenCV version 4.2.0 are detailed. Then, Section III is devoted to the basis of the particle filter algorithm and explanations how to apply this computing technique for tracking. In Section IV, the experimental tests of the algorithms are presented, by evaluating and comparing their performances on a real human tracking scenarios. Finally, Section V concludes this paper.

II. Usual Tracking Methods in OpenCV

OpenCV offers the possibility to compare a new method with other referenced algorithms [12]. Unfortunately, only few tracking algorithms are available in this library; they are presented in this section.

A. Tracking, learning, and detection (TLD)

TLD [13][14] decomposes the long-term tracking task into three components: (short-term) tracking, learning, and detection. In practical terms, TLD is based on the appearances, by re-sampling the bounding box of the target to a normalized resolution, regardless of the aspect ratio of the patch. Thus, the algorithm tries to track a blob of pixels from one frame to another. Then, the detector localizes all appearances that have been observed so far and corrects the tracker if necessary, and the learning estimates the detectors errors and updates it in order to avoid future errors. TLD provides stable tracking in the long-term when the object is visible. On the other side, the output of this tracker tends to jump around a bit. For example, it may fail when a pedestrian is tracked and there are other pedestrians in the scene.

B. Minimum Output Sum of Squared Error (MOSSE)

MOSSE [15] uses an adaptive correlation for object tracking which produces stable correlation filters when initialized using the first 7 frames. The major contribution of the MOSSE-based tracker is robust to variations in lighting, scale, pose, and non-rigid deformations. It also detects occlusion based upon the peak-to-side lobe ratio. Hence, the tracker is able to pause and to resume where it left off when the object reappears. MOSSE tracker is fast and can operate at a higher fps as represented in Fig. 3. Lastly, it is also very easy to implement.

C. Boosting

Tracker Boosting is based on an online version of AdaBoost [16][17]. This classifier needs to be trained at run-time with positive and negative examples of the object. The initial bounding box is taken as a positive example of the object using semi-supervised learning, and many image patches outside the bounding box are treated as the background searching regions of the same size as the target window from the surrounding background. Given a new frame, the classifier is run on every pixel in the neighborhood of the previous location and the score of the classifier is recorded transferring appearance changes, and also new observations. The new location of the object corresponds where the score is maximized.

D. Kernelized Correlation Filter (KCF)

KFC is built on the ideas presented in the previous detailed tracker (Boosting). Indeed, initially, KCF tracks the object based on Kernel [18], it extracts characteristics from Histogram of Oriented Gradients (HOG) to improve tracking accuracy [12]; it has a high processing speed and is superior for tracking the object in real time, as reported in Fig. 3. This tracker utilizes the fact that the multiple positive samples used in the Boosting tracker have large overlapping regions. Hence, this overlapping data leads to some properties that are exploited by this tracker to make tracking faster. Despite this speed of execution, the ordinary KCF tracker has certain shortcomings. As an example, edge effects that do not allow optimal tracking of the object when it is close to the image border, or, unfortunately, the inability to manage changes in target size.

E. Channel and Spatial Reliability Tracker (CSRT)

CSRT improves the Discriminative Correlation Filter (DCF) algorithm by introducing spatial and channel reliability [19]. The spatial reliability map is used to find out the optimal filter size, which makes the CSRT tracker better than the traditional DCF algorithm by adjusting non-rectangular targets. Actually, the spatial reliability map is used to adjust the filter support to the selected region for tracking. The channel reliability is measured to weigh the importance of each channel filter, then combines them to get the final response map. Using only the HOG and color histograms, the CSRT tracker achieves a high accuracy for object tracking with fast computational time (but less than previous algorithms, see Fig. 3).

III. Particle Filtering for Tracking

Particle filter is widely used for tracking problems. Each particle models the probability to find the desired object based on specific features in a Bayesian framework [20][6]. The main idea is to represent the posterior density by a set of random particles with associated weights and, then, to compute estimated positions based on these samples and weights.

A. Particle filtering basis

Particle filtering also called Sequential Monte Carlo method [11] has been widely described in the literature [10]. This is a useful algorithm in vision-based applications because it is a simple way to find an optimal solution for multidimensional problems by randomly generating a large number of possible system states. The main ideas of particle filter approach are recalled here. The question of state X_k estimation given the comments $z_{1:k}$, can be considered equivalent such as the estimation of the probability density function $p(X_k|z_{1:k-1})$, where $z_{1:k}$ represents $(z_1, ..., z_k)$. Assuming that these observations are independent and the system is Markovian, thus:

$$p(X_k|z_{1:k}) \propto p(z_k|X_k) \int p(X_k|X_{k-1}) \cdot p(X_{k-1}|z_{1:k-1}) \, \mathrm{d}X_k. \quad (1)$$

Thus, the Bayesian framework provides an optimal recursive solution to this problem based on two steps as illustrated by:

$$p(X_{k-1}|z_{1:k-1}) \xrightarrow{prediction} p(X_k|z_{1:k-1}) \xrightarrow{correction} p(X_k|z_{1:k}). \quad (2)$$

The idea of the particle approximation is based on the strong law of large numbers according to which the expectation calculated on the samples is an estimator of the expectation of the real density. Formally, denoting $p(X)$ the probability density that we are trying to approach, and $\{s^{(i)}\}_{i=1}^{N}$ a set of N independent and identically distributed samples according to p, then we have for any continuous and limited ϕ function:

$$\frac{1}{N} \sum_{i=1}^{N} \phi(s^{(i)}) \xrightarrow[N \to \infty]{} E_p[\phi(X)] = \int \phi(X) \cdot p^N(X) \mathrm{d}X, \quad (3)$$

where E_p designates the expectation taken in relation to density p. Let us note the quantity $p^N(X)$ defined by:

$$p^N(X) = \sum_{i=1}^{N} \pi^{(i)} \cdot \delta_{s^{(i)}}(X), \quad (4)$$

with $\pi^{(i)} \geqslant 0$, $\sum_{i=1}^{N} \pi^{(i)} = 1$ and $\delta_{s^{(i)}}$ a Dirac distribution centered on a particle $s^{(i)}$. Thus, it is possible to associate a weight $\pi^{(i)}$ to each particle $s^{(i)}$ by computing:

$$\pi_k^{(i)} = \frac{\pi_{k-1}^{(i)} \cdot p\left(z_k|X_k = s_k^{(i)}\right)}{\sum_{i=1}^{N} \pi_{k-1}^{(i)} \cdot p\left(z_k|X_k = s_k^{(i)}\right)}. \quad (5)$$

Consequently, for every ϕ, the estimation state is given by:

$$E_p[\phi(X)] \approx \int \phi(X) \cdot p^N(X) \mathrm{d}X = \sum_{i=1}^{N} \pi^{(i)} \cdot \phi\left(s^{(i)}\right), \quad (6)$$

representing the expectation of the particles $s^{(i)}$ depending on their weights and the density p. Finally, the algorithm 1 summarizes its basic form.

B. Particle filter implementation

In practice, the algorithm obtained by adding a resampling step is known as SIR (Sampling Importance Resampling), Bootstrap or Condensation. Knowing that the probability of a drawn particle is proportional to its weight, the crucial stage of resampling consists in a weighted draw among a choice of N particles. Hence, low-likelihood particles will therefore have little chance of being selected by the resampling step, while high-likelihood particles will be duplicated and then increase their chances of selection. Thereafter, a new equivalent weighting is given (an equivalent weight of $1/N$ for the N particles). Usually, in the event of the target loss, the tracker attempts to reacquire the target: the particles are uniformly distributed over the image area and the filter is allowed to converge again. Consequently, the stability of the filter depends on the number of particles: in the one hand, the more the number of particles increases, the more stable the filter is. On the other hand, the compilation time increases.

Algorithm 1: Particle Filter, usual framework

Require N particles $\{s_0^i\}_{i=1}^{N}$, at time k, and weight $\pi_0^i = \frac{1}{N}$
Prediction, see eq. 2
for *i=1 to N* **do**
  Generate $s_k^{(i)}$ according to $p(X_k|X_{k-1} = s_{k-1}^{(i)})$
Correction, see eq. 2
Update the particle weights
for *i=1 to N* **do**
  Compute $\pi_k^{(i)}$, eq. 5
Estimated particles at time $k-1$**:** $\left\{(s_{k-1}^{(i)}, \pi_{k-1}^{(i)})\right\}_{i=1}^{N}$
State estimation at time k**:** $E_p[X_k] = \sum_{i=1}^{N} \pi_k^{(i)} \cdot s_k^{(i)}$, eq. 6
Particle resampling: $\left\{(s_k^{(i)}, \frac{1}{N})\right\}_{i=1}^{N}$

Here, the output of the considered particle filtering is the expectancy of the estimator: $E_p[X_k] = \frac{1}{N} \sum_{i=1}^{N} s_k^{(i)}$. Other estimators can be used, such as the best particle (in term of likelihood), or the average in a portion of the reduced from the status space. However, the first leads to more outliers and the second takes more computational time. Here, the implementation of object tracking is based on a combination of color histograms [20] and LBP texture features [21] with popular Bhattacharyya coefficient [22] (each histogram contains 32 bins). Technically, this particle filter is implemented in C++ and runs with OpenCV.

IV. Experimental Results and Evaluations

A. Dataset: pedestrians, moving camera and ground truth

To assess the detailed approaches, a new dataset of videos has been created regarding human tracking with a perspective moving camera. Tab. I reports the details of 6 sequences under different climate conditions. The ground truth (the box containing the target) which is used for quantitative assessment is annotated manually for each frame (x and y coordinates, height and width of the box). In addition, videos and coordinates of these boxes are available online: https://github.com/agnihsv/human-walking-ground-truth.

Video	#person(s)	#frames	fps	Challenges
walk1	2	1250	20	Person walking against a similar target and crossing another person (occlusion) with changing background
walk2	1	405	20	Strong occlusion and shadow, low luminosity, similar color/texture between the target and the background
walk3	1	350	20	Person walking, backlighting and low luminosity
walk4	1	300	20	Person walking and rotating, target scale change, occlusion and no distinct color for the target
walk5	1	1279	20	Person walking alone, rotating camera, strong scale change of the target, high background change
walk6	2	1232	20	Target walking in the same direction as a similar person then the second person pass the target (occlusion)

TABLE I
Description of video sequences and their tied challenges

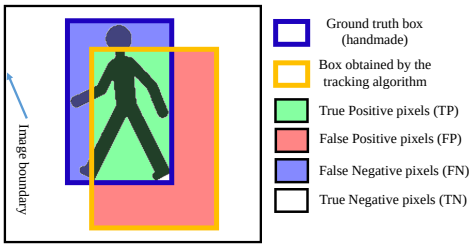


Fig. 2. Example of ground truth versus a desired bounding boxes.

B. Evaluation method: unbiaised Intersection over Union

To quantify the percentage of overlap between the target mask (i.e., handmade Ground Truth) and the prediction results (computed by a tracker), the unbiased "Intersection over Union" (IoU) measure developed in [23] is used by counting FN (False Negative), TP (True Positive), TN (True Negative) and FP (False Positive) points, as illustrated in Fig. 2. This metric is closely related to the Dice coefficient and computes an unbiased overlap score for large and small objects in the image with the following formulae:

$$Score = \frac{TP}{TP+FP+FN} \cdot w_0 + \frac{TN}{TN+FP+FN} \cdot w_{bg}, \quad (7)$$

with $w_0 = \frac{(TP+FP+FN)^2}{(TP+FP+FN)^2+(TN+FP+FN)^2}$ and $w_0 = 1 - w_{bg}$.

This measure assesses the trackers efficiency by computing a score between 0 and 1:

- a score close to 1, the tracker is qualified as suitable,
- a score close to 0 corresponds to a poor tracking.

Thus, the compared algorithms are evaluated by the scores computed by eq. 7 frame per frame and plotted in Fig. 4.

C. Reliability tests of OpenCV tracking methods

Tracking algorithms described in section II, namely, CSRT, TLD, KCF, MOSSE, Boosting and particle filter are qualitatively and quantitatively compared. To evaluate the performance of the these tracking methods, 6 videos containing human walking and captured with a moving camera were produced (namely walk1, walk2, walk3, walk4, walk5 and walk6, see Fig. 4). Each video contains different levels of complexity with challenging situations for detection and tracking. As detailed in Tab. I, these videos contain object similarities, target occlusions, background changes in the scene, decreased lighting, or scale changes of the target.

Firstly, Fig. 4 reports the qualitative and quantitative results of the different approaches using the experimental dataset. Hence, images on the left present the initial constraint boxes with the cyan color (handmade ground truth) and the predicted constraint boxes in the different frames for each compared detector. Usually, our experiments have shown that the majority of the tracker algorithms are inefficient with respect to total target occlusion on video walk2 (Fig. 4-a_2), to the changes of scale of the target, to the low luminosity and to the presence of objects resembling each other on video walk1 (Fig. 4-a_1). Moreover, the experimental results show that MOSSE is reliable in the face of low luminosity (Fig. 4-a_3) and background changes in the image of the walk3 video. In contrast, the CSRT obtains a robust tracking of the target even in the presence of similar objects, low brightness and background changes in the scene and scale changes, Fig. 4-a_1 of the walk1 video and 4-a_6 of the walk6 video, but, is inefficient, with a total occlusion (Fig. 4-a_2) on video walk2.

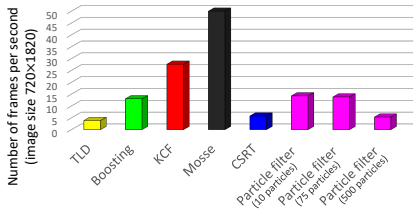


Fig. 3. Number of treated frames per second for the compared trackers under Ubuntu 18.04.5 LTS, Intel®, Core™ i7-9850H CPU@2.6GHz, 4Go RAM.

In terms of accuracy metric (eq. 7), the basic evaluation of the tracking algorithms is correlated with the overlap rate calculated for each frame of the video sequence. Figs. 4-$b_{i,i\in\{1,\ldots,6\}}$ shows the overlap for each video sequence.

1) Video walk1: Regarding this video, 2 similar persons are crossing CSRT (blue curves in Figs. 4-*b*) gives a score close to 1 and is therefore the most accurate compared to other OpenCV trackers. In contrast, the scores of TLD (in yellow), show that it is the most vulnerable tracker, because several false positives appear in the video, see all screened frames. Also, for KCF, MOSSE and Boosting, the curves highlight that they are unreliable for this dataset, i.e., they lose the target and the scores fall back to zero with the presence of two crossing persons in the video (besides MOSSE loses the target before crossing). Indeed, KCF and MOSSE calculate respectively the correlation and a mean square error, but here a similar person is hiding the target, so the matching remains always high and the trackers follow the undesirable crossed person instead of desired the target. Boosting behaves quite identically, when two similar persons meet and the second person hides the desired target, it will update the classifier taking into account the second person, hence the failure of the tracker for this sequence. TLD behaves also well until the crossing of persons, then tracks only the undesirable target.

2) Video walk2: There are two main difficulties in this video. Firstly, it contains an occlusion of the target, as illustrated in Fig. 4-a_2. Secondly, both the color and the texture of the target are not discriminated, compared to the background. Consequently, the total occlusion provides the loss of the target with all the trackers and decrease the score at a glance until 0 for the remaining frames (Fig. 4-b_2).

3) Video walk3: Fig. 4-a_3-b_3 exhibit the performance of the trackers for this third video. Clearly, due to the back-lighting and the low luminosity, all the algorithms fail here around half of the video, excepted MOSSE. Indeed the decreasing brightness acts on the target colors, which considerably changes the appearance of the target during the tracking.

4) Video walk4: In this video, the person if turning on himself and, at the same time, going away (Fig. 4-b_4). All the trackers lose the target before the half of the video, excepted MOSSE after 200 frames. Indeed, especially based histogram of gradient, KCF tracker fails from the outset to update the filter because of the person rotation and has no background learning. Therefore, the background change causes the correlation rate values to drop considerably and the loss of the target (therefore a null score). These difficulties are the same for the Boosting classifier. Here, TLD is especially disturbed by the particular movements of the target and its scale changes.

5) Video walk5: This video allows to evaluate the tracker effectiveness in the presence of high scale changes of the target (camera close to the person, or people walking towards the camera), with slow movements and no occlusion. In this way, Fig. 4-b_5 reports the evaluation of the different trackers. Here, TLD is the worse tracker because it loses totally the target. Alternatively, CSRT, MOSSE and Boosting are relatively effective, and their scores are always close to 1. Nevertheless, KCF obtains a tolerable tracking, but less suitable than the 3 previous trackers. Note that all the trackers (excepted TLD) are able to capture the target at the end of the sequence.

6) Video walk6: This last video contains two similar persons walking in the same direction. Here, most of the algorithms correctly track the target until the second person passes the first one, creating an occlusion (Fig. 4-a_6-b_6). The curves clearly highlight that they are unreliable, where CSRT, TLD and MOSSE track the wrong person while both Boosting and KCF do not track anybody.

7) Evaluation of particle filter: The detailed particle filter based tracker is also implemented in C++ and runs with OpenCV. It has been tested and evaluated on the 6 same videos, see Fig. 4-$c_{i,i\in\{1,\dots,6\}}$ for the quantitative evaluation. Clearly, the robustness of the particle filter is highlighted here for each walk video, compared to other OpenCV trackers. Indeed, curves for the particle filter show a good precision compared to other trackers. One key parameter of particle filters is the number of particles; here it is tested by tuning the filter with 10, 75 and 500 particles respectively to evaluate the performance of the implemented particle filter. Hence, scores corresponding to 10 and 75 particles are more unstable compared to with 500 particles which obtains a very good tracking precision. With 75 or 500 particles, it enables to correctly distinguish the target even in the presence of similar person. These experimental results are confirmed by the performance curves in Fig. 4-$c_{i,i\in\{1,\dots,6\}}$ where the score is very close to 1 during of target tracking. In the presence of an occlusion (Figs. 4-a_1-a_2-a_5), the filter widens its search area to find again the target. While the target has not yet appeared in the search area of the target, the filter is not updated, so, the likelihood that the particles decrease thus drastically. The tracking particle filter provides also a good reliability to the brightness change and low luminosity (Figs. 4-a_2 and a_3).

Generally, the more the number of particles increases, the more the tracked box tends towards its desired position, thus illustrating the interest of using a large number of particles. This result is confirmed by the plotted curves in Fig. 4-$c_{i,i\in\{1,\dots,6\}}$. With these curves, the obtained scores have frequent oscillations using 75 particles and are null for curves tied to 10 particles. Actually, a higher number of particles can improve the estimated bounding box of the target but, unfortunately, sacrifices performance speed, because the algorithm has to process more particles. It may result in a strong disadvantage; the consumption at computing time can technically interrupt the tracking. To that end, quantitative results obtained by the particle filtering with 75 particles demonstrate a suitable tracker with reasonable execution time (similar to Boosting, see Fig. 3). Consequently, the detailed particle filter based on both color histogram and LBP texture descriptor using 75 particles is a good compromise between tracking performance and execution time, compared to other tracking algorithms. Note that the execution of this filter is fast for 75 particles (see Fig. 3), it is also fast by down-sampling the image and choosing 500 particles (40 fps, 78 fps and 150 fps for images of size 960×590, 640×360 and 320×180 respectively, computation on video walk2 with the same computer characteristics as in Fig. 3). Another point is that the reference image is a crucial element for the initialization of this particle filter. Indeed, as these features relate to the color histogram and the LBP descriptor, the original selection zone must (imperatively!) be contained inside the target. As soon as pixels belonging to the reference image are not contained in the current target (and inversely), the tracking is less reliable and may fail, especially when the target passes through an area with a change in brightness or after an occlusion.

V. Conclusion

This study presents a comparison of commonly used OpenCV trackers in the framework of human tracking with a perspective and moving camera, namely: TLD, Boosting, KCF, MOSSE and CSRT. These algorithms serve as references to compare news trackers. Even though their computational time is fast, they are not reliable to track human (deformable targets) using a camera in movement. Indeed, several scenarios through these videos (backlight, low luminosity, occlusion, background...) highlight the defectiveness of these algorithms. Nevertheless, experiments and evaluations provided by this study advise to compare this tracking sort using particle filters based on both color and texture descriptor (LBP) histograms and outperform trackers available in the OpenCV library software regarding many complex scenes.

Eventually, a new dataset of videos has been created regarding human tracking with a perspective moving camera video. The ground truths of this database are also available online.

References

[1] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM CSUR*, vol. 38, no. 4, pp. 13–es, 2006.

[2] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE TPAMI*, vol. 36, no. 7, pp. 1442–1468, 2013.

[3] M. Ullah and F. Alaya Cheikh, "A directed sparse graphical model for multi-target tracking," in *IEEE CVPR Workshops*, 2018.

[4] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *IEEE CVPR*, 2017, pp. 7291–7299.

[5] A. Khalifa, I. Alouani, M. A. Mahjoub, and N. E. B. Amara, "Pedestrian detection using a moving camera: A novel framework for foreground detection," *Cogn. Syst. Research*, vol. 60, pp. 77–96, 2020.

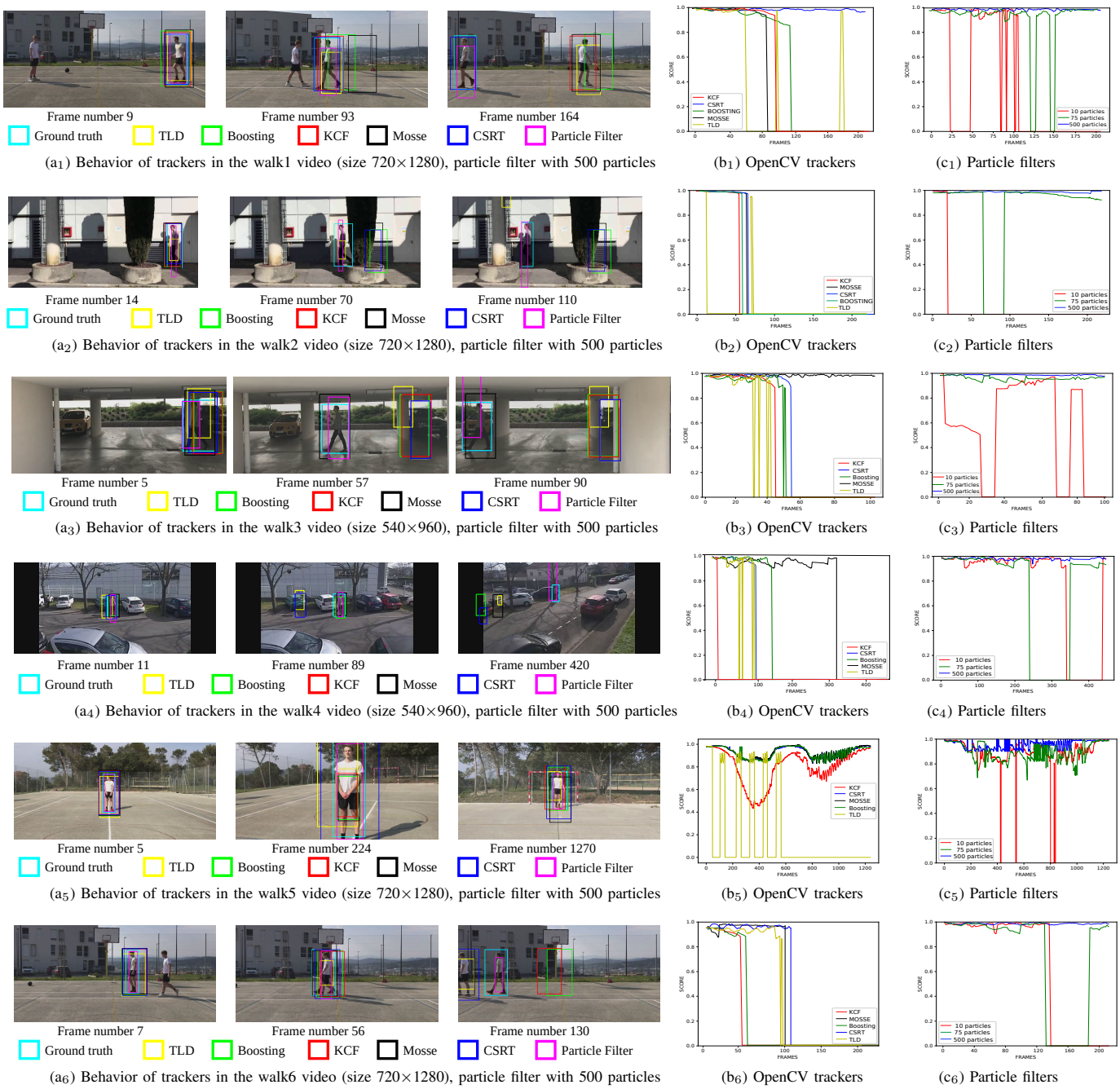
[6] A. Mekonnen, F. Lerasle, and A. Herbulot, "Cooperative passers-by tracking with a mobile robot and external cameras," *CVIU*, vol. 117, no. 10, pp. 1229–1244, 2013.

[7] B. Magnier, G. Ekszterowicz, J. Laurent, M. Rival, and F. Pfister, "Bee hive traffic monitoring by tracking bee flight paths," in *VISAPP*, 2018, pp. 563–571.

[8] M. Fiaz, A. Mahmood, and S. K. Jung, "Tracking noisy targets: A review of recent object tracking approaches," *arXiv:1802.03098*, 2018.

[9] R. Pereira, M. Couto, F. Ribeiro, R. Rua, J. Cunha, J. P. Fernandes, and J. Saraiva, "Energy efficiency across programming languages: how do energy, time, and memory relate?" in *ACM SIGPLAN International Conference on Software Language Engineering*, 2017, pp. 256–267.

[10] M. Isard and A. Blake, "Condensation—conditional density propagation for visual tracking," *IJCV*, vol. 29, no. 1, pp. 5–28, 1998.



(a_1) Behavior of trackers in the walk1 video (size 720×1280), particle filter with 500 particles (b_1) OpenCV trackers (c_1) Particle filters

(a_2) Behavior of trackers in the walk2 video (size 720×1280), particle filter with 500 particles (b_2) OpenCV trackers (c_2) Particle filters

(a_3) Behavior of trackers in the walk3 video (size 540×960), particle filter with 500 particles (b_3) OpenCV trackers (c_3) Particle filters

(a_4) Behavior of trackers in the walk4 video (size 540×960), particle filter with 500 particles (b_4) OpenCV trackers (c_4) Particle filters

(a_5) Behavior of trackers in the walk5 video (size 720×1280), particle filter with 500 particles (b_5) OpenCV trackers (c_5) Particle filters

(a_6) Behavior of trackers in the walk6 video (size 720×1280), particle filter with 500 particles (b_6) OpenCV trackers (c_6) Particle filters

Fig. 4. Comparison of particle filter and tested OpenCV trackers on videos containing human(s) acquired with a moving camera.

[11] J. S. Liu and R. Chen, "Sequential monte carlo methods for dynamic systems," *J. of the American Stat. Assoc.*, vol. 93, no. 443, pp. 1032–1044, 1998.

[12] V. Lehtola, H. Huttunen, F. Christophe, and T. Mikkonen, "Evaluation of visual tracking algorithms for embedded devices," in *SCIA*. Springer, 2017, pp. 88–97.

[13] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE TPAMI*, vol. 34, no. 7, pp. 1409–1422, 2011.

[14] W. Hailong, W. Guangyu, and L. Jianxun, "An improved tracking-learning-detection method," in *CCC*. IEEE, 2015, pp. 3858–3863.

[15] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *CVPR*. IEEE, 2010, pp. 2544–2550.

[16] C. Gao, N. Sang, and R. Huang, "Online transfer boosting for object tracking," in *ICPR*. IEEE, 2012, pp. 906–909.

[17] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting." in *BMVC*, vol. 1, no. 5, 2006, p. 6.

[18] J. Shin, H. Kim, D. Kim, and J. Paik, "Fast and robust object tracking using tracking failure detection in kernelized correlation filter," *Applied Sciences*, vol. 10, no. 2, p. 713, 2020.

[19] A. Lukezic, T. Vojir, L. Cehovin Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter with channel and spatial reliability," in *CVPR*, 2017, pp. 6309–6318.

[20] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Robust tracking-by-detection using a detector confidence particle filter," in *ICCV*. IEEE, 2009, pp. 1515–1522.

[21] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE TPAMI*, vol. 24, no. 7, pp. 971–987, 2002.

[22] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE TPAMI*, vol. 25, no. 5, pp. 564–577, 2003.

[23] G. Häger, M. Felsberg, and F. S. Khan, "Countering bias in tracking evaluations," in *VISAPP*, vol. 5. SciTePress, 2018, pp. 581–587.