Indian Journal of Science and Technology, Vol 9(27), DOI: 10.17485/ijst/2016/v9i27/97696, July 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Fast Multi-Object Tracking-by-Detection Using Tracker Affinity Matrix

Pavel Aleksandrovich Kazantsev, Pavel Vyacheslavovich Skribtsov and Sergey Olegovich Surikov

PAWLIN Technologies Ltd., Dubna, Russia; pak@pawlin.ru, pvs@pawlin.ru, sso@pawlin.ru

Abstract

Objectives: This study is an attempt to develop a fast real-time multi-object tracking algorithm with competitive precision and accuracy in crowded scenes. **Methods/Statistical Analysis:** This research extends multi-object tracking-by-detection framework by utilizing physical characteristics of trackers and their affinity for tracker guidance, in addition to detections and representation models. Guidance is done by particle filter which weights are updated through reworked observation model, featuring a term based upon tracker affinity. Tracker affinity is also used for tracker grouping that removes redundant trackers following the same target, and triggering a special propagation mode during occlusions that prevent identity switches. **Findings:** Our research has shown that trackers affinity matrix and algorithm features based on it yield substantial extra accuracy for tracking-by-detection framework, while taking a minor fraction of processing time. **Improvements/ Applications:** In this paper we have introduced an approach that makes it possible to use less accurate, but faster detectors and representation model classifiers, enabling real-time processing, while keeping competitive precision and accuracy.

Keywords: Multi-Object Tracking, Tracking-by-Detection, Particle Filter, Pedestrian Detection, Particle Filtering, Online Learning, Tracker Affinity, Surveillance

1. Introduction

Millions of surveillance cameras have already been installed throughout the world and their quantities continue to rise[1]. When aided by intelligent software, capable of processing data streams in real-time, the range of their applications becomes vast[2] with most notable use found in public security, traffic safety, crime prevention and investigation[3], marketing[4]. Majority of these applications deal with the behavior of moving objects – pedestrians and cars, mainly. Semantic information of higher levels is extracted from motion trajectories, obtained by tracking the objects using various algorithms.

State-of-the-art multi-object tracking algorithms use tracking-by-detection approach augmented by representation models of objects for guiding a specific search algorithm[5-8]. Algorithms proposed in these works can be classified into two categories: online tracking[5,7] and offline tracking[6,8]. Online tracking approach assumes that data association at any given time is made using only past entries, while offline approach uses global optimization by considering future entries in a batch in addition to past entries. Another advantage of offline tracking is that in order to guide a tracker it takes into account surrounding trackers. In other words, global batch optimization instructs a guiding algorithm on where certain tracker cannot move, because this or that position is most probably occupied by other trackers. Though offline algorithms currently dominate charts of challenging multi-object tracking benchmark, In[9] online algorithms are still more preferable in certain applications where immediate result is required. It is important to note that offline global optimization is an extra data association mechanism and does not exclude benefits of local optimization techniques used in online tracking.

As benchmark for single-object tracking shows[10], there are two distinctive winners among search algorithms: particle filters[11] and dense sampling.[12] Dense sampling alleviates local minima problem inherent to gradient descent methods[13,14], but at the cost of high computational load. Probabilistic approaches, such as particle filters, can be considered as a trade-off between two above-men-

*Author for correspondence

tioned approaches, featuring relative insensitivity to local minima and computational efficiency[15,16]. In this work we use particle filter as a search algorithm, since this work concerns computational efficiency entirely.

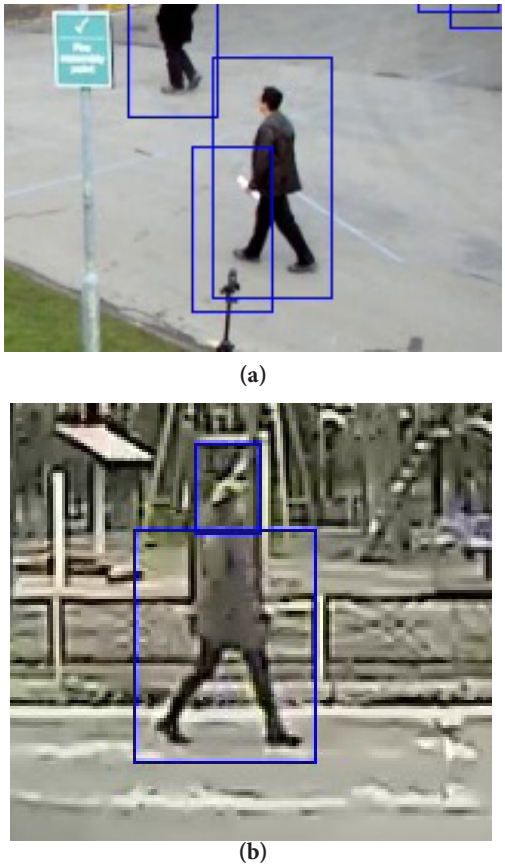
Online and offline tracking-by-detection approaches rely on relatively stable detections and models that are not a given in real-world applications, where camera field of view is arbitrary and computational power is rather limited.

In this regard, as our contribution, we introduce an approach that utilizes additional sources of information – physical characteristics of trackers and their affinity – that compensate failures of detectors and model classifiers, and, hence, make it possible to use simple and fast versions of the latter and still maintain high precision and accuracy. A major attention was paid to dealing with hard tracking cases that mainly occur when objects cross their trajectories producing occlusions and may lead to frequent identity switches. As a result, we obtain a tracking algorithm that works at 25-50 fps with competitive precision and accuracy as covered in the Results section.

2. Concept Headings

2.1 Algorithm Overview

In our research we use tracking-by-detection framework proposed in[5]. In addition to class-specific and target-specific information our algorithm incorporates physical characteristics of trackers and knowledge about other trackers that exists at any given time. A set of physical characteristics is represented by location, size and velocity. Physical characteristics are integrated into framework in two ways. Firstly, they are used for *tracker grouping*. Grouping mechanism takes into account size, location and velocity in order to cluster trackers and then conditionally merges them into one tracker, thus eliminating a problem where several trackers were initiated for one actual object, because of several detections triggered for this object (see Figure 1). Indeed, even if detections were spawned on object inaccurately, the trackers they have initiated will track the same actual object and, hence, their location and velocity after some time steps will become nearly identical. This grouping mechanism proved to be very useful when applying simple, fast, but less accurate detectors, like a blob detector that uses foreground mask as an input image. Secondly, physical characteristics are used to describe relationships between trackers, presented in a form of *tracker affinity matrix*.



(a)

(b)

Figure 1. Several detections on the same actual object: (**a**) HOG detector (**b**) blob detector applied on mask.

Knowledge about other trackers, presented by their positions and affinity matrix, is incorporated into the particle filter weight update rule in the form of an additional negative term which we call *anti-weight*. Its purpose is to severely diminish weights of those particles, which occurred in the field of *enemy* trackers, i.e. trackers that substantially differ by physical characteristics. This mechanism prevents trackers from identity switches between close targets with similar representation models. This is especially important when we operate with weak (but fast) classifiers. Too high average anti-weight value triggers *bad tracking case mode* for this tracker. This mode virtually means that too many particles belonging to this tracker lie in the field of enemy trackers. This scenario mostly occurs when targets overlap. While bad tracking case mode is enabled, tracker is being led only by approximated location, since detector output and representation model are very unreliable in such circumstances.

2.2 Particle Filter

Our tracking algorithm is similar to one used in[5] with a few exceptions. That being said, target state $X = \{x, y, v_x, v_y\}$ is represented by image coordinates (x, y) and velocity components (v_x, v_y). The bigger likelihood of a new observation at the position of particle p, the higher probability for it to remain for the next time-step and propagate in accordance with the following model:

$$(x, y)_t = (x, y)_{t-1} + (v_x, v_y)_{t-1} \cdot \Delta t + \delta(x, y) \quad (1)$$

$$(v_x, v_y)_t = (v_x, v_y)_{t-1} \cdot \Delta t + \delta(v_x, v_y)$$

Processes noise $\delta(x, y)$ and $\delta(v_x, v_y)$ is obtained from zero-mean normal distributions with variances calculated as follows. Position variance $\sigma(x, y) = \{\sigma_x, \sigma_y\}$ depends on the size of the current tracker, and consists of two potentially different components - σ_x is derived from target width, while σ_y - from target height. Velocity variance $\delta(v_x, v_y)$ is given by:

$$\delta(v_x, v_y) = \chi(v_x, v_y) + \tau(v_x, v_y), \quad (2)$$

where $\chi(v_x, v_y)$ is a noise velocity along current movement direction (centripetal noise) and $\tau(v_x, v_y)$ - noise velocity perpendicular to the current movement direction (tangential noise). Variances $\sigma_\chi(v_x, v_y)$ and $\sigma_\tau(v_x, v_y)$ for these noises are different. Their values depend on the types of abrupt movement patterns to be handled. Abrupt stops of a target are handled with bigger centripetal variance, while severe changes of direction during time step are better handled with bigger tangential variance. If we mix these types of noise like in formula (2), we have to choose what scenario is more probable and regulate by increasing correspondent variance. Another approach is to divide particles into three groups that are re-sampled individually. For the first group, movement noise is given by (2) with equal variances for both movement noises. That would be equivalent to applying one noise with this variance value. For the second group we leave only centripetal noise, and for the third – only tangential one.

Multi-modal distribution detection: Standard particle filter technique assumes that there is a single mode distribution. Our tracker also complies with that rule, but cases where one group of particles happens to be quite distant from another group of particles are rather frequent in complicated scenes. In such cases a mean position may appear in-between two actual targets, thus yielding erroneous prediction of the target position. To prevent this, while keeping rather simple particle filter, we first check for multi-modality. In current version we detect only two modes, since it is most frequent case of actual multi-modal distribution of particles. This is done after new weights are calculated and ready to be used for re-sampling procedure. Firstly, we fit current particles into an ellipse. Then we sort particles into two baskets depending on which side of the minor axis of the found ellipse they lie. These two groups of particles are fitted into other two ellipses. If these two ellipses do not overlap or overlap poorly (some reasonable threshold value is required), then the two-mode case is triggered. Two mean position values are calculated for both clusters of particles. Particles whose cluster mean value is closer to the current tracker mean are kept for this tracker and re-sampled, while particles from another cluster are used to initiate a new temporary tracker. The latter can be either discarded right away, or used to spawn a new *detection-by-tracker*, that will be fed into the pipeline only on the next iteration. We use the latter, since due to overall idea of our approach it is better to occupy all valid targets with trackers as soon as possible, so they will not let other trackers to slip on these positions (see section 2.4).

Initialization, termination, rejuvenation and aging: Each detection that was not associated with any tracker and does not substantially overlap with existing trackers is used to initialize a new tracker. Detections located everywhere on image are eligible for tracker initialization. Each tracker has an attribute we call *age*. Initial age value is zero and when it reaches a certain value the tracker is terminated (or temporally deactivated). If the tracker has no detection associated with it on a current time step, it is aged, i.e. its age value is incremented. The tracker is rejuvenated, i.e. its age is decremented (but not below initial zero value), if the following criteria have been met: detection has been associated with this tracker, high representation model score, detections that overlap this tracker have been found nearby. In[5] the tracker is terminated (or temporally deactivated) when no detections have been associated with it for a certain amount of time steps. Hence, we generalize this termination rule by allowing a tracker to survive longer even without detections, provided its representation model is consistent and detections nearby resemble this tracker to some extent.

2.3 Data Association

In order to associate detections with trackers we use *greedy data association* algorithm explicitly described in[5], which is a simplified and, hence, less consuming version of Hungarian algorithm[17]. This algorithm takes matching score matrix S as an input and then finds best matches *(tr, d)* of the tracker *tr* and detection *d* by iteratively running across the matrix and choosing best match for each iteration, followed by removal of corresponding row and column from further consideration. Only those matches which score are above the threshold result in a successful association.

Matching score we use is similar to that of[5], but with altered components:

$$S(tr,d) = g(tr,d)\cdot\left(c_{tr}(d) + a\cdot\sum_{p\in tr}^{N} p_N(d-p)\right) \quad (3)$$

where terms $c_{tr}(d)$ and $\sum_{p\in r}^{N} p_N(d-p)$ do not differ from[5] and stand for the representation model score and the normal distribution $N(pos_d - pos_p;0;\sigma^2)$ evaluated for the distance between the position of detection d and particle p, respectively. A gating function $g(tr, d)$ is modified as opposed to [5], where it is given as:

$$g(tr,d) = p(size_d|tr)\cdot p(pos_d|tr) \quad (4)$$

$$= \begin{cases} p_N\left(\frac{size_{tr}-size_d}{size_{tr}}\right)\cdot p_N(|d-tr|) & \text{if } |v_{tr}| < t_v \\ p_N\left(\frac{size_{tr}-size_d}{size_{tr}}\right)\cdot p_N(dist(d,v_{tr})) & \text{otherwise} \end{cases}$$

Firstly, we totally remove dependence on velocity from this formula. The way this function is given in[5] implies that "fast-moving objects cannot change their course abruptly because of inertia". That is not a case in real world applications where due to low camera frame rate, arbitrary camera views, abrupt movements, which really exists in practice, hence, detection for current frame can be not consistent with the previous one in terms of direction predicted by the inertia rule. This, and even a slight inaccuracy in detection, may turn gating function to zero. Our gating function is given by formula:

$$g(tr,d) = \begin{cases} p_N\left(\frac{size_{tr}-size_d}{size_{tr}}\right)\cdot p_N(|d-tr|) & \text{if } B(tr) = false \\ 0 & otherwise \end{cases}$$

where $B(tr)$ is an indicator Boolean function that returns *true* if a bad tracking mode is enabled (see Sec. 2.6) for a particular tracker in which several operations, such as association or model update are prohibited.

Online combined classifier: In order to estimate similarity between the tracked target and detection we use a combined representation model that consists of color and texture models. The color model is based upon the color histogram and the texture model - upon the local binary patterns histogram. Since distance ranges in color and texture spaces do not match, we transform them in probabilities and bring to the range of [0;1] by formula:

$$P = \exp\left(-\frac{(d-d_m)^2}{2\cdot\sigma^2}\right), \quad (6)$$

where P is a score (probability) estimate for a particular model in the range of [0;1]; d- a distance between the detection model and tracked target model in the corresponding space; d_m - a minimal possible distance for this model, σ - sigma for this model. Distances for both color and texture models are computed as Euclidian distances of histograms. d_m and σ values are computed during model initialization and re-initialization. Model initialization and re-initialization, in fact, represent a learning process. Model initialization occurs when the tracker is being initialized by the non-associated detection. It consists of computing a histogram (color or LBP), d_m and σ. d_m, in general case, is computed as a distance between the stored signature (in our case it is a histogram) and the initialization image fragment, and in most cases must be equal to zero. σ is computed as follows. First we build a training set that consists of positive and negative samples. Positive samples are taken from initializing immediate neighborhood of the detection. For each positive sample we check that it does not overlap the existing trackers. Negative samples are taken from other tracker windows, their neighborhood, and background fragments. We calculate a distance to a stored signature for each sample in the training set and then find a threshold distance that provides a minimal error value for this training set. This threshold distance is chosen as σ. Model signature is updated at each iteration, provided that the tracker it corresponds to has a high representation model score (regulated by a threshold). If the representation model score is low, which means an indicator of model being outdated, model for this tracker is re-initialized, i.e. not only its signature is updated, but also its d_m and σ are recalculated. If the tracker overlaps other trackers or is in

a bad tracking mode, neither update, nor re-initialization is performed. Both color and LBP histograms used in the models contain low number of bins in order to make calculations faster. This, of course, reduces their discriminative effectiveness, but allows real-time processing.

2.4 Observation Model

As in[5], we also estimate likelihood of a particle, but using a reworked formula for computing weight $w_{tr,p}$ for particle p of tracker *tr*:

$$w_{tr,p} = \begin{cases} \beta(d_{tr}) \cdot p_N(p - d_{tr}) + \eta \cdot c_{tr}(p) - \mu \cdot w_a(p) & B(tr) = false \\ p_N(p - d_{est}) & B(tr) = true \end{cases} \quad (7)$$

where η and μ are fixed parameters and remain unchanged during entire video-sequence. $B(tr)$ is the same indicator function as in (5). Terms of (7) are described below. Also the reasoning behind the modification of observation model used in[5] is explained.

Detection term $\beta(d_{tr}).p_N(p - d_{tr})$ is computed as normal distribution p_N from the distance between the center of detection d_{tr} and location of the particle p.d_{tr} is defined by:

$$d_{tr} = \begin{cases} d_a & \text{if association occurred} \\ d_{est} & \text{otherwise} \end{cases} \quad (8)$$

where d_a is a detection associated with this tracker at the current time step (as explained in Sec. 2.3), and d_{est} is an estimated location of the tracker if no association occurred. Estimated tracker location is obtained by transitioning its previous location using average velocity vector. The average velocity vector is calculated using formula:

$$\vec{v} = \frac{\vec{p}_{t-1} - \vec{p}_{t-t_0}}{\Delta s} \quad (9)$$

where t is a current time step, t_0 - look-back depth (5-10 frames), Δs - tracker travel distance in the time range of $[t-1, t-t_0]$ in the image coordinate system. Note that throughout the rest of the paper, tracker velocity is given by (9). This is done to make direction component of velocity more stable.

$\beta(d_{tr})$ assumes different values, depending on cases presented by (8), but for each case it remains the same for the duration of video-sequence. Also inequality $\beta(d_a) > \beta(d_{est})$ holds true, meaning that associated detection will always have more influence on the particle likelihood than the estimated position. In[5] the detection term contains an indicator function that assumes two values: 1 if detection was associated with the tracker, and 0 – otherwise. Our reasoning behind maintaining at least some location guidance, even if no detection was associated with the tracker, concludes in that in cluttered scenes classifier term $c_{tr}(p)$ may be unreliable and even turned to zero (if targets overlap, for example), leaving tracker without detector-classifier guidance at all. Should we use another term from[5] – a detector confidence – then it would mean that in such cases (no detection, low classifier score) the tracker would lean towards locations with even very low detector confidences, caused by other targets or even background structures. That is why we, first, introduce the estimated location guidance, and, secondly, remove detection confidence term from the observation model. When $B(tr) = true$, the estimated location becomes the only source of information for the tracker guidance.

Classifier term $\eta \cdot c_{tr}(p)$ is computed as an output score of online combined classifier, as described in Sec. 2.3., in a location of particle p using current size of the tracker. This is the most time-consuming operation in the entire pipeline, since it needs to be applied at each particle location. With heavy representation models, calculation of this term for all particles almost guarantees that real-time tracking is unobtainable. That is why, as mentioned above, we use light, and fast, classifiers. This, of course, leads to limited guidance power of the classifier term. Hence, we usually set η parameter at a low value, so classifier term input is several times lower than the one of the detector term. This way, our usage of this term should be treated as slight correction, rather than the major source of guidance information, as it is done in[5]. The reasoning behind this concludes in that in easy cases, where targets are not surrounded by each other, the tracker is guided just fine by detection alone, and in the absence of such, even by a weak representation model. In the crowded scenes, where targets always occlude and overlap each other, even a powerful representation model fails to prevent identity switches (trackers "jumping" from one target to another), especially in the real-world scenarios with arbitrary camera views, low-resolution images, small, several-pixel-sized targets. The next term we introduce is intended to solve the problem of identity switches, while keeping simple classifiers.

Anti-weight term $\mu \cdot w_a(p)$ is meant to guide the tracker by providing the observation model with information about where the tracker cannot propagate. Unlike other terms in (7), anti-weight comes with a negative sign, thus diminishing the resulting weight $w_{tr,p}$. The purpose

of anti-weight is to severely diminish likelihood of a particle if it occurs in a vicinity of other trackers that are not similar to the tracker this particle belongs to by physical characteristics (location, velocity, size). Anti-weight for each particle *p* of tracker *tr* is computed as a maximum of anti-weights computed towards all other existing trackers $tr' \in T$.

$$w_a(p_r) = \max_{tr' \in T, tr' \neq r} \{w_a(p_r, tr')\} \quad (10)$$

$$w_a(p_{tr}, tr') = \begin{cases} (1 - R(tr, tr')) \cdot p_N(p_{tr} - tr') & R(tr, tr') < r \\ 0 & R(tr, tr') \geq r, \quad p_{tr} \in box_{tr} \end{cases} \quad (11)$$

where R is a $M \times M$ tracker affinity matrix, where M is a total number of active trackers. $p_N(p - tr')$ is a normal distribution taken from the distance between the particle and other tracker tr'; ρ is a fixed threshold value, which purpose is to let particles hit the vicinity of *friendly* trackers, i.e. trackers that are similar to tracker *tr* by physical characteristics. That is needed in order to make trackers grouping possible (see Sec 2.5); box_{tr} denotes bounding box of tracker *tr*. Thus, according to (11), particles of the tracker tr that lie inside this tracker bounding box are not affected by anti-weights. This is important to take into account, when using introduced anti-weights concept. Otherwise, when tracked targets collide, they may ban all of the each other's particles, making both trackers unstable and propagate in random directions (since all particles weights may turn to zero). Anti-weighting of outer particles can be thought of as removing scout particles that hit already pre-occupied spots outside this tracker body.

Affinity score of the pair of trackers lies within the interval of [0;1] and is given by the formula:

$$R(tr, tr') = \frac{g(tr, tr') \cdot (\gamma \cdot p_N(tr - tr') + \upsilon \cdot p_{vel}(tr, tr') + \theta \cdot p_{area}(tr, tr'))}{R_{\max}} \quad (12)$$

where $g(tr, tr')$ is a gating function; γ, υ and θ - fixed parameters that define the input of location, velocity and area likelihoods into the overall affinity score. $R_{\max}$ is a maximum affinity score that stands for normalization term. Assuming that the gating function and likelihood terms lie in the range of [0;1], $R_{\max} = \gamma + \upsilon + \theta$. Gating function $g(tr, tr')$ is given by:

$$g(tr, tr') = g_{sp}(tr, tr') \cdot g_a(tr, tr') \quad (13)$$

All of the multipliers in (13) assume either 0 or 1 value. Hence, gating function $g(tr, tr')$ either zeroes the whole $g(tr, tr')$ score or leaves likelihood term as an actual score, as follows from (12). The purpose of the gating function in (12) is to nullify under certain circumstances possibly otherwise high affinity score between two trackers. These circumstances are explained by characteristic-specific gating functions in the right part of equitation (13).

$g_{sp}(tr, tr')$ is a speed gating function that is given by:

$$g_{sp}(tr, tr') = \begin{cases} 0 & |v_{tr}| < t_v \text{ and } |v_{tr'}| > t_v \\ 0 & |v_{tr'}| < t_v \text{ and } |v_{tr}| > t_v \\ 1 & otherwise \end{cases} \quad (14)$$

where v^{avg}_{r} and $v^{avg}_{tr'}$ are average velocities given by (9); τ_v is a speed threshold under which a target is considered stationary. Thus (14) nullifies affinity score if one tracker is stationary and another is not.

$g_{sp}(tr, tr')$ is an angle gating function that is given by:

$$g_a(tr, tr') = \begin{cases} 0 & \arccos\left(\frac{v_{tr} \cdot v_{tr'}}{|v_{tr}| \cdot |v_{tr'}|}\right) > j_v \\ 1 & otherwise \end{cases} \quad (15)$$

where ϕ_v is an angle threshold above which targets are considered moving in different directions. In our experiments we set its value equal to $\frac{\pi}{3}$. The way how angle gating function works is illustrated in section Sec. 2.6.

Likelihood terms from (12) are explained as follows: $P_N(tr - tr')$ is a normal distribution taken from the distance between trackers *tr* and *tr'*; $p_{vel}(r, tr')$ is a velocity likelihood that is given by (16) and $p_{area}(r, tr')$ is an area likelihood that is given by (17).

$$p_{vel}(tr, tr') = \begin{cases} 1 & |v_{tr}| < t_v \text{ and } |v_{tr'}| < t_v \\ p_N(|v_{tr}| - |v_{tr'}|) \cdot p_N\left(\arccos\left(\frac{v_{tr} \cdot v_{tr'}}{|v_{tr}| \cdot |v_{tr'}|}\right)\right) & otherwise \end{cases} \quad (16)$$

As follows from (17) velocity likelihood is maximum and equal to 1 if both targets are considered stationary. A first normal distribution measures agreement between trackers speeds, and the second one – between velocity directions.

$$p_{area}(tr, tr') = p_N\left(\frac{\min(s_{tr}, s_{tr'}) - S_{tr \cap tr'}}{\min(s_{tr}, s_{tr'})}\right) \quad (17)$$

In (17): s_{tr} and $s_{tr'}$ are areas of trackers *tr* and *tr'*, respectively; $S_{tr \cap tr'}$ is an intersection area of both trackers.

A maximum of P_{area} is achieved when a rectangle of a smaller tracker lies fully in the rectangle of the bigger tracker, or, when both trackers are of the same size and fully overlap each other.

2.5 Trackers Grouping

There is a possibility that several trackers may activate for the same target. In general, it is caused by detector's inaccuracy, as shown on Figure 1. In some cases, detection algorithm fails to group detections, because they are too far away from each other or have quite different sizes. In other cases, detection may trigger for certain target, but still not be associated with a tracker, that is already tracking it, because it is rather far away from it (low position likelihood) and contains a lot of background (low classifier score). This will lead to a situation where two or more trackers follow the same target, thus causing a huge input to false positive and identity switch rates, if measured by popular CLEAR MOT metrics[2]. These redundant trackers also increase computation time, since their representation and observation models need to be processed. If we would allow these trackers to multiply unrestrictedly, at the course of the time, their numbers will overwhelm the scene and, thus, drastically reduce accuracy and computational performance beyond any limit of usability (see Figure 2). This is especially devastating, when using fast and simple detectors (like blob detector) or reducing heavy detector's threshold in order to increase true accept rate by sacrificing accuracy.

In order to combat this issue we introduce trackers grouping technique. First we cluster all active trackers using classic labeling algorithm, described in[18]. This algorithm uses equivalency metric in order to assign a pair of elements to the same class. Equivalency metric between two trackers is given by affinity matrix (12) and, thus, based on trackers' velocity, location and size. If affinity score exceeds threshold value – trackers are considered equal.

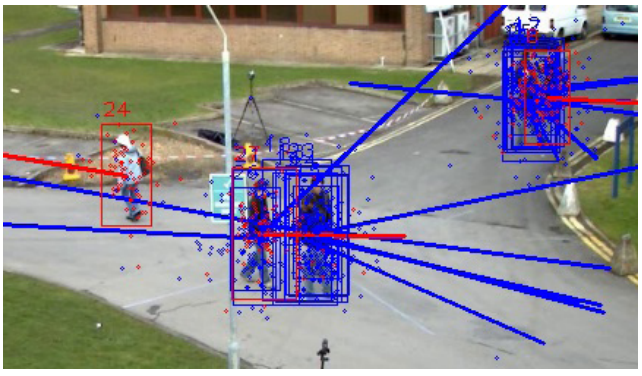


Figure 2. If no tracker grouping is performed, due to detector inaccuracies, trackers multiply at the course of the time and overwhelm the scene.

From each cluster we pick one tracker that has a biggest average score (given by (12)) towards other trackers in cluster. Let us denote it as a *winner* tracker. If there are only two trackers in a cluster, then they, obviously, will have identical average scores. As a matter of fact, it is a most common case of trackers grouping. To resolve this situation we apply a nested set of rules (see Table 1). In

Table 1. Nested rules to select a winner tracker from two trackers with identical average score

Nest level	Winning Rule	Commentary
1	Tracker tr is a winner if $len_{tr} > k \,.\, len_{tr'}$	len_{tr} and $len_{tr'}$ - trajectory lengths of trackers tr and tr', respectively. Coefficient k is supposed to make a clear distinction between the newly born trackers and long-living trackers, stably following a target. We set $k=10$.
2	Tracker tr is a winner if $len^*_{tr} < len^*_{tr'}$	len^*_{tr} and $len^*_{tr'}$- lengths of the last trajectory fragments without associated detections for trackers tr and tr', respectively. By this rule we select the tracker that is lately being tracked more robustly.
3	Tracker tr is a winner if $v_{tr} > v_{tr'}$	More stably propagated trackers attach to the moving target in a lower amount of time steps and, thus, adapt to its speed faster. This rule, however, works only for two newly born trackers and may play its role only during a shortperiod of time. This is explained by the fact that if one tracker happens to lag behind a target and the other tracks this target firmly, then they are unlikely to fall into the same cluster, first of all.

rare cases, where none of these rules helped to select a winner, no winner will be selected for this cluster on this iteration.

After winner tracker is selected in a cluster, its trajectory is concatenated with the oldest tracker in this cluster (a tracker with longest trajectory). Other trackers in this cluster are terminated completely and their trajectories are erased. These erased trackers are not included into final statistics calculation.

Figure 3 shows step by step illustration of trackers grouping process. Target appears in a camera's field of view, two detections trigger for it and spawn two new trackers (Figure 3a). It is worth mentioning, that in the case, shown in Figure 3, it is virtually impossible to avoid double-tracking of the same target, since these detections are two far away from each other and trigger on the same frame, so association is not possible at this time-step. During the rest of the frames shown, there is only one firm detection, but there are two trackers already following the same target (Figure 3b). Their affinity score is zero at this moment, since their velocity vectors point in different directions, thus zeroing angle gating function in accordance with (15). Since there is only one detection available for association, one tracker is mostly propagated by detection term and the other – by representation model and estimated location. In a few time-steps their positions almost coincide and velocity vectors start to align, since both trackers follow the same target (Figure 3c). Few more time-steps, and locations, rectangles, and, velocity vectors align enough to overcome equivalency threshold (Figure 3d). On the next frame trackers are merged into one (Figure 3e). Since there were only two trackers in a cluster, we use nested set of rules in order to select a winner (Table 1). This particular situation cannot be resolved using the first rule (trajectory sizes are equal, since both trackers were activated on the same time-step – Figure 3a). But the second rule is a clear-cut solution here – one tracker (winner) has been propagated by associated detection whole time, and the other – without one.

2.6 Bad Tracking Case Mode

Tracker affinity matrix is also used to handle events that occur during occlusion of one target by another[19,20]. In order to detect such events we use anti-weights, computed in accordance with (10)-(12), with one exception – anti-weights for all particles are taken into account, and not only these that lie inside the tracker box ($p_{i,r} \in box_r$ condition in (11) is excluded). All anti-weights, computed

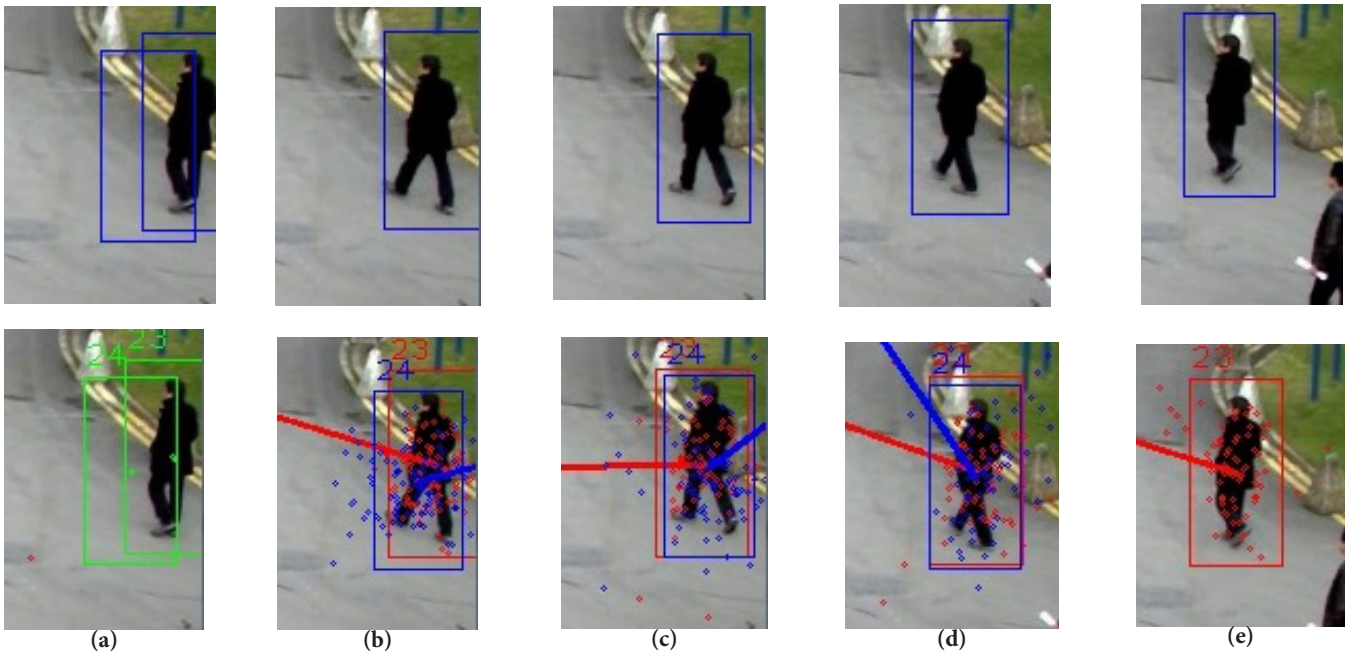


Figure 3. Two trackers activate for the same target. After several time-steps trackers are merged into one (upper row – detections, lower row – trackers). Lines drawn from the centers of the trackers depict velocity vectors. Green rectangles are newly activated trackers; red ones indicate trackers that have an associated detection on this frame, blue ones show trackers that are propagated without an associated detection.

for a certain tracker this way, are averaged, and this average anti-weight is compared with a threshold t_{wa}. If the average anti-weight exceeds t_{wa}, *bad tracking case mode* is enabled for this tracker. Bad tracking case mode virtually means that the tracker's propagation by means of detections and its representation model is unreliable. Our reasoning behind this is as follows. When two or more trackers which clearly belong to different targets (*enemy* trackers, i.e. trackers with the low affinity score) partially or fully overlap, detections that occur at the location where overlapping takes place cannot be reliably associated. The same refers to the representation models –bounding boxes of overlapping trackers include contents of each other's targets, making classifier terms in (7) misleading. This is especially true when using weak classifiers and tracking similar-looking targets. So the only more-or-less reliable information to be used for propagation in such cases is a history of previous locations for all trackers, involved in the occlusion event, approximated by $p_{\mathrm{N}}(p-d_{est})$ likelihood term in (7). However, not only we are unable to rely on detections and representation models in such events – we also cannot allow wrongful upcoming associations and noised representation model updates to happen. That is why, for a tracker in a bad tracking case mode detection association and representation model update/re-initialization is prohibited. When average anti-weight value drops below threshold, the bad tracking case mode is switched off after few time-steps, and the tracker is being propagated normally again.

Figure 4 illustrates the bad tracking case mode usage for an event where two targets collide. Two different targets approach each other (Figure 4a) at almost right angle that makes their affinity score equal to zero, according to (15). Few moments later the trackers collide (Figure 4b), their velocity vectors point in almost opposite directions, which ensures that their affinity score is still a zero. Almost all of particle locations are shared by the bounding boxes of both trackers, which in combination with zero affinity score yields a high average anti-weight value. The bad tracking case mode is triggered for both trackers, and they are being propagated by the estimated detection. Also, a single detection that erroneously contains both targets (Figure 4b top) is associated with neither of trackers. When the trackers successfully part their ways (Figure 4c), the bad tracking case mode is still being maintained for a few more time-steps, then it is switched off, and the trackers are being propagated normally with the associated detections (Figure 4d).

3. Results

In order to assess performance of our algorithm we used PETS'09 S2.L1–S2.L3[21] benchmarks which feature the following challenges: occlusions by tracking sign and

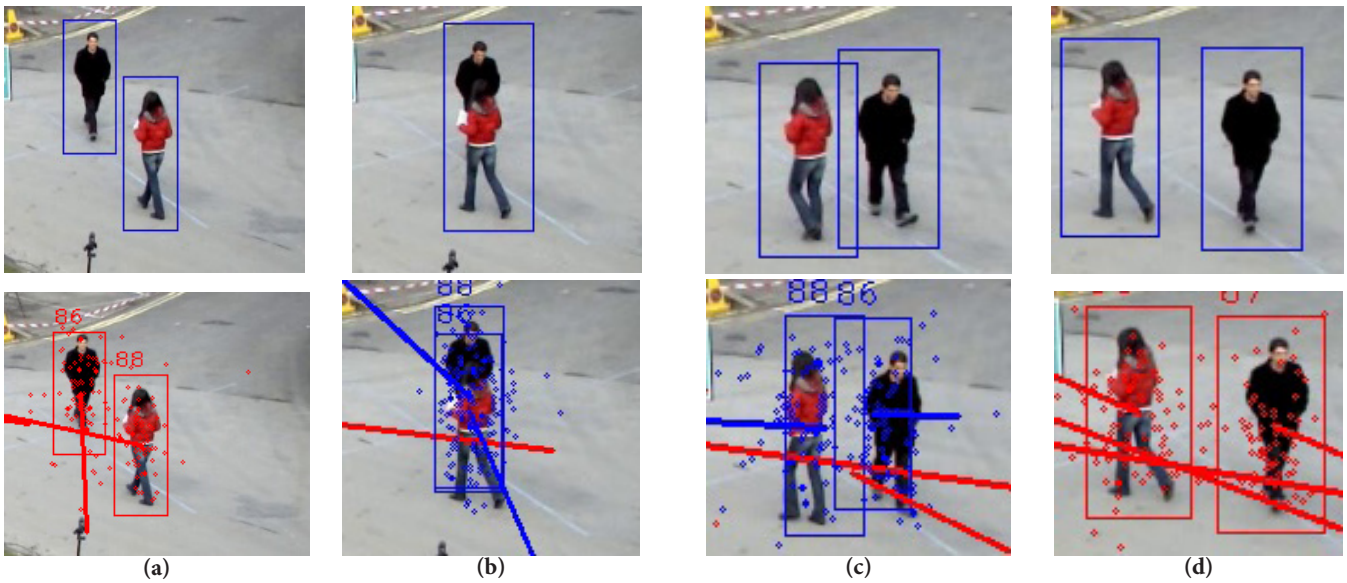


Figure 4. Bad tracking case mode specific propagation (upper row – detections, lower row – trackers). After collision, trackers part their way successfully, no identity switch occurs. Red rectangles indicate trackers that have associated detection on this frame; blue ones show trackers that are propagated without an associated detection.

tracking targets, non-linear movement patters (sudden stops, abrupt change of direction, walking in circles), change of lightning conditions. To evaluate precision and accuracy we use CLEAR MOT metrics[22]. Precision score (MOTP) is computed from the intersection areas of bounding boxes of a real target and a tracker. Accuracy score MOTA consists of the false negative (FN) and false positive (FP) rate, and the number of identity switches (ID Sw). Evaluation is done on the trackers which trajectory size is non-zero; thus, trackers that were terminated due to merging with the winner trackers during the tracker grouping procedure (see Sec. 2.5) are not taken into account. We have also evaluated processing times for each video experiment.

On PETS'09 S2.L1 we test two versions of our algorithm. The first one is more accurate and computationally heavy, since it uses the HOG-detector and the representation model, described in Sec 2.3. The second one is less accurate, but much faster, due to utilizing the background model, described in[23]. Using this background model we build an object mask that becomes source information for both the detector and representation model. Thus, in this simple version of algorithm, the HOG-detector is substituted by a simple blob detector and the representation model is reduced to the "mass" of an object, calculated as a number of non-zero pixels in a bounding box. For PETS'09 S2.L2-L3 we do not use light version, since it relies on a mask, and these two videos feature unstable lightning conditions.

Our results are compared with[5], since it is still the best algorithm among tracking-by-detection family. Also, our hardware and software tools are almost identical to the ones used in[5]. We have also implemented our system using C++ and did not make use of GPU. Our runtime performance results were obtained on the workstation with Intel Core2Duo 3GHz having 4GB of memory (versus Intel Core2Duo 2.13GHz with 2GB in[5]). Table 2 shows evaluation results.

The advantage of our algorithm, as compared to [5], is higher MOTP score and runtime performance (FPS). MOTP score, in fact, just shows how well the algorithm positions bounding boxes of trackers around the actual targets (missed targets are taken into account by FN score). Better MOTP results are mostly determined by the tracker grouping.

Trackers grouping technique plays a major role in the precise positioning and handles inaccurate detections in two ways, which need to be explained more thoroughly.

Table 2. CLEAR MOT evaluation results

Dataset	MOTP	MOTA	FN	FP	ID sw.	FPS
PETS'09 S2.L1						
Ours heavy (HOG-detector and RBG+LBP classifier)	87.2%	76.5%	19.6%	3.9%	2	22
Ours light (blob detector + non-zero pixels count classifier)	69.5%	63.2%	32.8%	4.0%	5	50
Those of [5]	56.3%	79.7	not reported	not reported	not reported	0.2-4 fps
PETS'09 S2.L2						
ours heavy	72.8%	61%	32.8%	6.2%	12	13
Those of [5]	51%	50%	not reported	not reported	not reported	0.2-4 fps
PETS'09 S2.L3 (ours heavy)						
ours heavy	74.5%	65.8%	26.8%	7.4%	10	12
Those of [5]	52%	67.5%	not reported	not reported	not reported	0.2-4 fps

Firstly, it allows avoiding rough averaging of detections, and secondly, it makes it possible to keep low position variances in the data association algorithm. By rough averaging we mean a grouping of detections that trigger in a vicinity of each other, but overlap quite poorly, as it is shown in Figure 1. Normally, the HOG-like detectors group output rectangles that have almost equal sizes and are located few pixels away from each other. Detections that are as distant from each other, as it is shown in Figure 1, for example, will not be grouped with any reasonable grouping threshold. In such cases, holding on the strict "one target-one tracker"-rule, as in[5] and other tracking-by-detection algorithms, leaves two options – either roughly group detections, which makes tracker positioning much less accurate, or let redundant trackers live, which reduces MOTA by increasing false positives error. Terminating trackers that do not have associations for few time-steps also cannot be relied upon, because even initially inaccurate trackers can easily propagate

to the target, stick to it, and subsequently associate with accurate detections in turns with an initially precise tracker, thus breeding identity switches. When tracker already exists and inaccurate detection triggers nearby, one either needs to associate these two with a large position variance, thus, again, making positioning less accurate, or to activate a new tracker, which spawns problems described above. With tracker grouping technique we allow redundant trackers to spawn, avoiding rough grouping and data association with the large position variance, thus keeping tracker that follows a target firmly undisturbed. But when these redundant trackers gain high affinity score with an accurate tracker (and it takes them to have similar positions, speed, velocity vector and intersection areas, which virtually means that they follow the same target), we cluster them together, select a winner tracker in the cluster, and deactivate redundant trackers. This can be thought of as "association by physical characteristics".

Our FPS advantage is determined by the fact that we use much weaker classifiers and do not re-train them on each frame (we retrain them only when the classifier score becomes too low). The weakness of our classifiers is compensated by anti-weights which, firstly, prevent trackers to jump to other targets and, secondly, because they trigger bad tracking case mode. Computing affinity matrix, which is used in anti-weights calculations, even with a large number of trackers, in its turn, does not take much time (about 5-10% of total processing time).

4. Discussion

In this section we discuss the influence of the proposed algorithm features based upon tracker affinity matrix on the overall accuracy, namely: tracker grouping, anti-weight term in the observation model and bad tracking case mode. The influence is illustrated as a precision and accuracy drop (see Table 3) when abovementioned features are switched-off. The anti-weight term is switched-off by setting zero value to μ parameter in (7). The bad tracking case mode is switched-off by setting theoretically unreachable threshold value of t_w (see Sec. 2.6). The tracker grouping is switched-off by a Boolean flag in the program and substituted by averaging of detections and higher values of position variance in the data association algorithm. This experiment was carried out on a PETS'09 S2.L1 dataset using the heavy version of algorithm (HOG-detector and RBG+LBP classifier).

Table 3. CLEAR MOT results for PETS'09 S2.L1 dataset using heavy version of algorithm (HOG-detector and RBG+LBP classifier), obtained with different combinations of the proposed features, based upon tracker affinity matrix: anti-weight term in observation mode (AW), bad tracking case mode (B), trackers grouping (TG)

Features	MOTP	MOTA	FN	FP	ID sw.	FPS
AW + B + TG	87.2%	76.5%	19.6%	3.9%	2	22
B+TG	85.6%	53.2%	41.2%	4.1%	5	~22
AW + TG	87.8%	57.4%	38.3%	3.4%	4	~22
AW + B	51.3%	66,6%	22.8%	10.6%	2	19
TG	86.8%	46,6%	52.1%	1.7%	12	~22
AW	53.4%	45.4%	42.3%	12.4%	4	18
B	51.6%	41.6%	43.6%	14.8%	5	18
no proposed features used (equivalent to[5] with weaker classifiers)	50.2%	40.4%	43.3%	16.3%	13	16

Evaluation results presented in Table 3 confirm our theorizing about effectiveness of the algorithm features based upon tracker affinity matrix. When used altogether, these features almost compensate the weakness of fast classifiers, while yielding higher MOTP metric and FPS, as compared to[5]. Each feature, though, has its own sphere of influence. Tracker grouping greatly improves MOTP, has moderate positive influence on the rate of false positives and FPS. Bigger FP rate without tracker grouping is caused by the fact that not all inaccurate detections can be grouped with accurate ones or associated with trackers, even with a large grouping distance threshold and location variance, respectively. Thus, some redundant trackers still keep roaming the scene. This is also a reason for lower FPS, since more trackers mean more processing time required to propagate them. The anti-weight term in the observation model and the bad tracking case mode make false negative and identity switch rate lower, as intended. The anti-weight term does not allow trackers to propagate towards similar looking nearby targets and the bad tracking case mode handles occlusion events.

It is worth mentioning that no matter how appealing our overall results may seem to a reader (a bit lower MOTA value, and higher MOTP and FPS values), the tracker affinity matrix and algorithm features based on it still yield substantial extra accuracy for tracking-by-detection framework, and, we believe, can be efficiently used

with stronger and heavier classifiers that are re-trained on each iteration, in order to achieve higher MOTA, even at the cost of lower FPS.

Our light version of algorithm that uses the blob detector and non-zero pixels count classifier is, surely, much less accurate, but still yields reasonable results, while operating at 50 FPS at Intel Core2Duo 3GHz processor without using any special optimization. Such computation efficiency promises a successful real-time implementation of our algorithm even on board of IP cameras.

5. Conclusion

We have proposed an algorithm that extends multiple-object tracking-by-detection framework that mainly uses detections and representation models for tracker guidance. Our extension concludes in using an additional source of information–relationships between trackers based on their physical characteristics (location, speed, velocity vectors and sizes), and formalized in the form of tracker affinity matrix. Tracker affinity matrix is a basis for three important algorithm features we introduced in this paper: (1) Tracker grouping mechanism that uses affinity scores to cluster trackers, making it possible to dispatch redundant trackers following the same target; (2) The negative anti-weight term in the observation model that prevents a tracker to propagate towards targets which are already followed by other non-affine trackers; (3) The bad tracking case mode that triggers when non-affine trackers collide and resolves occlusion events by altering the observation model, data association and representation model update order. Tracker grouping substantially improves precision and lowers false positives rate, while the anti-weight term and bad tracking case mode reduce false negatives and identity switch rates.

Experiments have shown that using abovementioned algorithm features makes it possible to use less accurate, but faster detectors and representation model classifiers, enabling real-time processing, while keeping competitive precision and accuracy. A light version of our algorithm that uses primitive and very fast detector and representation model works at 50 FPS on the decade-old processor without any special optimization, while featuring reasonable precision and accuracy.

We see the future development of our algorithm in utilizing global optimization techniques, incorporated into tracking-by-detection framework as special extra terms in the data association and observation model equations.

Acknowledgements

The research was financed by the Ministry of Education and Science of the Russian Federation under the Grant Contract dated 24 November, 2014, (grant number 14.579.21.0070, contract unique identifier RFMEFI57914X0070), the grant was assigned for the research into "2D- and 3D-data fusion and processing software for object detection and recognition".

6. References

1. Welsh BC, Farrington DP. Public Area CCTV and Crime Prevention: An Updated Systematic Review and Meta-Analysis. Justice Quarterly. 2009 October; 26(4):716-45.
2. Wang L, Hu W, Tan T. Recent developments in human motion analysis. Pattern Recognit. 2003; 36(3):585-601.
3. La Vigne N, Lowry S, Markman J, Dwyer A. Evaluating the use of public surveillance cameras for crime control and prevention. Washington, DC: US Department of Justice, Office of Community Oriented Policing Services. Urban Institute, Justice Policy Center. 2011.
4. Burke RR. The Third Wave of Marketing Intelligence. In: Retailing in the 21st Century. Krafft M and Mantrala MK (Eds.) Springer Berlin Heidelberg, 2010; p.159-71.
5. Breitenstein MD, Reichlin F, Leibe B, Koller-Meier E, Gool LV, Online Multiperson Tracking-by-Detection from a Single, Uncalibrated Camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 2011; 33(9):1820-33.
6. Choi W. Near-Online Multi-target Tracking with Aggregated Local Flow Descriptor. ICCV, 2015 1-11. Date accessed: 20/04/2016: Available from: http://arxiv.org/pdf/1504.02340.pdf.
7. Grabner H, Leistner C, Bischof H. Semi-supervised on-line boosting for robust tracking. Computer Vision – ECCV, Series Lecture Notes in Computer Science. 2008; 5302:234-47.
8. Wang B, Wang G, Chan KL, Wang L. Tracklet association with online target-specific metric learning. Proc. of IEEE Conf. CVPR. 2015; p. 1234-41.
9. Leal-Taixe L, Milan A, Reid I, Roth S, Schindler K. MOT Challenge 2015: Towards a benchmark for multi-target tracking. arXiv:1504.01942 [cs], 2015 April.
10. Wu Y, Lim J, Yang MH. Online object tracking: A benchmark. Proc. Comput. Vis. Pattern Recognit. 2013; p. 2411-18.
11. Zhong W, Lu H, Yang M-H. Robust Object Tracking via Sparsity-based Collaborative Model. Proc. IEEE Conf. on CVPR. 2012; p. 1838-45. Date accessed: 23/04/2016: Available from: http://faculty.ucmerced.edu/mhyang/papers/cvpr12b.pdf.

12. Hare S, Saffari A, Torr PHS. Struck: Structured Output Tracking with Kernels. Proc. IEEE International Conference on Computer Vision (ICCV). 2011; p. 263-70.
13. Comaniciu D, Ramesh V, Meer P. Kernel-Based Object Tracking. PAMI. 2003; 25(5):564-77.
14. Sevilla-Lara L, Learned-Miller E. Distribution Fields for Tracking. Proc. IEEE International Conference on CVPR. 2012; p. 1910-17.
15. Jia X, Lu H, Yang M-H. Visual Tracking via Adaptive Structural Local Sparse Appearance Model. Proc. IEEE International Conference on CVPR. 2012; p. 1822-29.
16. Mei X, Ling H. Robust Visual Tracking using L1 Minimization. Proc. IEEE 12th International Conference on ICCV. 2009; p. 1436-43.
17. Kuhn H. The Hungarian method for the assignment problem. Naval Research Logistics Quarterly. 1955; 2:83-87.
18. Cormen ThH, Leiserson ChE, Rivest RL, Stein C. MIT Press: Chapter 21: Data structures for Disjoint Sets. Introduction to Algorithms (2nd ed.). 2001; p. 498-524.
19. Sheshasayee A, Lakshmi JNV. Comparison of Machine Learning Algorithm on Map Reduction for Performance Improvement in Big Data. IJST. 2015 November; 8(29).
20. Mekhtiche MA, et al. Real Time Object Detection & Tracking over a Mobile Platform. IJST. 2015 November; 8(32).
21. Ferryman JM. (Ed.) Snowbird, USA: Proceedings of the 12th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance. IEEE Computer Society. 2009.
22. Bernardin K, Stiefelhagen R. Evaluating multiple object tracking performance: The CLEAR MOT metrics. J. Image and Video Processing. 2008; 3:1-10.
23. Skribtsov PV, Surikov SO, Yakovlev MA. Background Image Estimation with MRF and DBSCAN Algorithms. 2015; 8(S10):1-6.