On-line fusion of trackers for single-object tracking

Isabelle Leang, Stéphane Herbin, Benoît Girard, Jacques Droulez

On-line Fusion of Trackers for Single-Object Tracking

Isabelle Leang[a], Stéphane Herbin[a], Benoît Girard[b], Jacques Droulez[b]

[a] *ONERA - The French Aerospace Lab, 91761 Palaiseau, France*
[b] *Sorbonne Universités, UPMC Univ Paris 06, CNRS, Institut des Systèmes Intelligents et de Robotique (ISIR), F-75005 Paris, France*

Abstract

Visual object tracking is a fundamental function of computer vision that has been the object of numerous studies. The diversity of the proposed approaches leads to the idea of trying to fuse them and take advantage of their individual strengths while controlling the noise they may introduce in some circumstances. The work presented here describes a generic framework for combining and/or selecting on-line the different components of the processing chain of a set of trackers, and examines the impact of various fusion strategies. The results are assessed from a repertoire of 9 state-of-the-art trackers evaluated over 46 fusion strategies on the VOT 2013, VOT 2015 and OTB-100 datasets. A complementarity measure able to predict the overall performance of a given set of trackers is also proposed.

Keywords: visual object tracking, fusion, on-line behavior analysis

1. Introduction

Single-object tracking is a computer vision function with a long research history. Indeed, mastering the capacity of pursuing reliably and efficiently

Email addresses: isabelleleang@gmail.com (Isabelle Leang), stephane.herbin@onera.fr (Stéphane Herbin), girard@isir.upmc.fr (Benoît Girard), droulez@isir.upmc.fr (Jacques Droulez)

a given target, and being robust to various nuisance phenomena, is the key to many off-line or on-line applications exploiting video data (security, video surveillance, road traffic control, production control, human-machine interaction, multimedia indexing). This research topic produces a huge amount of studies each year, sometimes accompanied by new evaluation benchmarks and metrics, e.g. the ALOV++ dataset [1], or the On-line Tracking Benchmark [2]. One of the recent outstanding benchmark actions has been the yearly VOT Challenges[1] that have emphasized the evaluation of single-object model-free (no pre-learned model) short-term (no re-detection function) tracking through two criteria: robustness to drift and localization accuracy. The main conclusions were that "None of the trackers consistently outperformed the others by all measures at all sequence attributes" and that "trackers tend to specialize either for robustness or accuracy" [3].

What is proposed in this article is to build on this empirical fact and study how an existing repertoire of available trackers can be generically combined in an efficient way. Tracker combination will be based on the traditional fusion concepts of redundancy — tracker outputs are combined — and complementarity — the repertoire of trackers samples different features and functional structures. A key component of a fusion scheme is the availability of a self-diagnosis capacity: its role is to prevent the propagation of errors during fusion, detect and discard trackers with noisy behaviors, and eventually correct them. It will be shown that possessing such a capacity greatly improves the fusion performance.

The emphasis of the present work will be put on analyzing the drifting behavior of trackers. From an operating point of view, losing the tracked

[1] `http://www.votchallenge.net/`

target is certainly the most impacting type of error, since it implies also loss of its awareness. Target re-acquisition is possible but is generally less reliable and more costly. The VOT challenge has been one of the first benchmarks to address specifically this question, and our evaluation will follow their methodology.

Fusion approaches can be segmented in two families [4]: passive or active. The passive approaches only combine tracker outputs with no interaction between the trackers, whereas the active ones integrate data provided by each tracker with the objective of correcting their inner model when necessary. We show that active fusion leads in general to better performances, but necessitates a control over tracker components and update mechanisms.

Determining what are the most efficient combinations of trackers, i.e. enabling a high level of robustness at low cost, has a practical impact. We introduce a complementarity measure between trackers based on individual drift measures to predict the fusion performance of the combined trackers in order to select it. The main contributions of this paper are the following:

(1) We describe a generic and parametric framework for the combination of trackers and identify several levels and modes where fusion can operate. The formalism results in 46 different configurations.

(2) We show experimentally that the ability to predict drift is essential for good fusion performance and propose several on-line schemes to compute such a prediction that can be used in active fusion approaches.

(3) We evaluate and rank the fusion configurations on 4 databases using a repertoire of 9 state-of-the-art trackers.

(4) We propose a quantitative way to predict the complementarity of trackers to determine the best combinations for fusion.

The paper is organized as follows: Section 2 introduces the different notions of fusion applied to the on-line combination of trackers. Section 3 presents the corresponding related work. Section 4 evaluates a series of trackers and analyses their behavior with the idea of fusing them. Several ways to detect on-line abnormal tracker behavior are presented in Section 5. The fusion framework is described in Section 6. Section 7 gives material and implementation details for the experiments used to identify the key fusion strategies in Section 8. Section 9 concludes with final recommendations.

2. Exploiting multiple trackers

The fusion of trackers is based on a simple principle: single trackers have each their domain of expertise that can be enlarged by fusion at different levels. To understand how fusion can be realized, a better understanding of the structure and functioning of a single tracker is first needed.

2.1. Single tracker

The objective of a tracker is to predict on-line the position of a target in each frame of a video given an initial position. This position is represented by a rectangular bounding box (x, y, w, h) adapted to the region occupied by the target, (x, y) are the coordinates of the top-left corner, and (w, h), its width and height. To deal with the appearance or motion variations of the target, modern trackers exploit a time dependent target model which is updated when a new frame is made available so as to deal with the target appearance or motion variations. This model is commonly built using appearance (color, texture, gradient, spatio-temporal) or motion features [5]. A tracker usually operates in two stages: prediction and update. It predicts the position by measuring a consistency score with the target model, the position with the

highest score distance is usually taken as the target's new position. The model is then updated using this new prediction.

2.2. Fusing trackers

Given a collection of assorted trackers, each characterized by a specific type of image features, target model and updating mechanism, how can they be combined to improve globally tracking performances?

A first design step is to identify a functional architecture. Two types have been classically studied: cascade and parallel. In a cascade architecture, trackers run sequentially and decide at each step to operate or not the next tracker. One significant advantage of using cascade is the reduction of the computation costs by executing the fewest possible trackers. However, cascade architectures are often difficult to set-up, and are very sensitive to single tracker failures due to their chained dependencies. Since the main objective devoted to fusion in our paper is to increase the overall tracking robustness, this type of architecture is clearly not the best option.

In a parallel architecture, all trackers run simultaneously. In this family, two approaches of fusion can be distinguished: passive or active [4]. In *passive* approaches, trackers run independently from each others. The objective of fusion is simply to combine their individual predictions. One limitation of this scheme is that there is no functional way to correct drift: if a tracker from the fused family loses the target, it cannot recover or only by chance, for instance if the target trajectory crosses the wrong prediction. Thus, fusion schemes must find a way to identify erroneous predictions to discard them before combining the outputs. We will describe three different ways to do so in Section 6.

Drift has two consequences on a tracker behavior: it drives the target out

of the search window used to predict the next best target position, and it pollutes the samples used to update the target model, which may have lower discriminating capacity. One of the roles of an *active* fusion approach is to rely on the other trackers to correct these two misbehaving features. This can be done using actions such as controlling the samples used to update the model, or reinitializing the model, or correcting the search area from the pool of tracker outputs. The key feature is therefore the design of a reliable selection process analyzing tracker behavior, either individually or collectively.

3. Related Work

The idea of fusion in tracking has been addressed in the literature either as a control and selection of models exploited in the processing chain, or as a dynamic combination of the inputs and outputs of several modules. The information fusion domain sometimes speaks of centralized vs. decentralized functional architectures [6].

Most of the tracking approaches proposed in the literature can be considered as falling into the centralized architecture category: they describe different ways to combine either motion models [7, 8, 9, 10], observation models [9, 11] or appearance features [9, 12, 13, 14, 15, 16] to cite few of them. Since our concern in this study is the analysis of decentralized strategies for tracking, we won't describe these approaches further and refer to other recent surveys [17, 5, 1].

Decentralized fusion approaches applied to tracking differ on three aspects: the specificity of fused modules, the nature of interaction between modules (passive vs. active) and the way individual module outputs are

evaluated.

Specific fusion schemes

Computer vision literature offers many types of processing modules that can be combined in a specific way to build a complex tracker, typically by the association of generic and on-line learned detectors. The approach proposed by Siebel et al. [18] combines in a complex way two detectors (motion and face) and two trackers (region-based and shape-based) to track people specifically.

Other studies combine detectors and trackers of a whole target: Stenger et al. [19] propose parallel and sequential fusion schemes for hand and face tracking involving several trackers and an off-line trained detector, whose role is to reinitialize failing trackers. Trackers are evaluated on-line using their own confidence score: this score corresponds to an expected tracking precision error from an off-line training that is used to determine the good or bad operation of the tracker.

Santner et al. [20] exploit the specificities of three trackers based on different designs in a cascade: template correlation, on-line random forest detectors and optical flow, each tracker correcting the previous one.

The last two approaches can be considered as following an active fusion scheme since they sequentially combine tracker outputs.

Generic fusion

Generic fusion approach combine tracker outputs with little knowledge of the trackers. Bailer et al. [4] propose a passive fusion of a large amount of tracker outputs (29 tracking algorithms from [2]) by maximizing a bounding box attraction function and smoothing the final trajectory.

Zhong et al. [21] propose a passive fusion of tracker outputs using a weakly supervised learning to simultaneously infer the most likely object position and the probability of each tracker to provide accurate predictions.

Moujtahid et al. [22] combine a set of AdaBoost trackers [23] based on heterogeneous characteristics (color, texture, shape) and operating independently. At each moment, the best tracker is selected and updated from a standardized confidence score and a spatio-temporal coherence measure. In a more recent version, Moujtahid et al. [24] add a context predictor computed from image features in their tracker selection process.

On-line module selection

Most of the previously presented approaches exploit a confidence score produced along the predicted object bounding box and aiming at characterizing the quality or reliability of the provided information. The role of a selective strategy is to detect on-line each tracker abnormal behavior before fusion.

A first idea is to detect when the low level image processing behaves badly. A learning approach has been proposed by Zhang et al. [25] to predict computer vision system failures for application of semantic segmentation, vanishing point estimation and camera parameter estimation from a series of image features (SIFT, colors, textures, gist, HOG, line histograms, LBP, similarities) using a Support Vector Machine for classification or regression. Daftry et al. [26] proposes the same kind of approach but exploits Deep Convolutional Networks to extract image features and fine tune them on learning sequences. Grimmett et al. [27, 28] introduce the idea of *introspective classification* and apply it to predict the risk of taking a decision for specific families of classifiers in robotics applications. However, all those approaches

are limited to memoryless image processing and require the knowledge of an annotated learning database to specifically build the failure predictor off-line: they cannot be applied to assess the behavior of complex dynamic systems such as visual tracking algorithms.

Biresaw et al. [32] describe a particle filter based approach, where each particle is considered as a single tracker, and propose a selection and correction strategy based on the spread of each particle correlation matrix to detect abnormal behavior. Tracker selection is only produced from its spatial behavior.

Another approach is to compare the results of various trackers to assess their behavior. Kalal et al. [29] describe the TLD which is the cooperation of a flow-based tracker and an on-line learned detector. They are combined and updated using a low resolution template based detector assessing the quality of each estimate. A more recent version of a similar active fusion scheme where the combination law is described as a Hidden Markov Model, the HMMTxD, has been published by Vojir et al. [30] and uses a set of trackers and a high precision detector (zero false positive rate and 30% recall) to update the HMM parameters and reinitialize the incorrect trackers. The HMM estimates the most probable state of the trackers (correct or incorrect operation) and outputs an average bounding box from the correct trackers.

Table 1 summarizes the various works described in this section.

The fusion approach we propose in this article is the fusion of modules: our goal is to combine a heterogeneous repertoire of trackers, i.e. modules of variable performance and cost, and allowing a generic level of interaction between them. The combination is controlled through an on-line assessment of each tracker behavior, which will govern the global dynamics of selection, aggregation and correction of a pool of trackers. It will also be shown that

Table 1: Comparison of various fusion approaches. The second column gives the type of fusion implemented (passive vs. active). The third column gives the number of fused modules or n if unbounded and the fusion architecture ($\|$ for parallel, $\perp$ for cascade). The fourth column gives indication of the way tracker outputs are combined (*iteration* means that the fusion sequentially corrects tracker outputs, *selection* means that several tracker outputs are discarded, and *weighting* means that the resulting output is a weighting average of tracker predictions.)

Reference	Fusion type	Combined modules	Output combination
Siebel [18]	passive	4 in $\|$	iteration
Santner [20]	active	3 in $\perp$	iteration
Stenger [19]	active	4 in $\|$ et $\perp$	selection or iteration
Kalal [29]	active	2 in $\|$	selection
Vojir [31]	active	2 in $\|$	selection
Biresaw [32]	active	n in $\|$	weighting
Zhong [33]	passive	n in $\|$	weighting
Moujtahid [22]	passive	n in $\|$	selection
Bailer [4]	passive	n in $\|$	weighting
Our work	active	n in $\|$	selection and/or weighting

the simple fusion of tracker outputs has some benefit and can be easily introduced in our framework to discard drifting trackers.

4. Off-line Tracker Evaluation

A usual action before proposing a fusion approach is to state the quality of the components that will be combined. We present in this section how individual tracker performance is classically addressed (Section 4.1). Additionally, we describe a local analysis evaluating the potential complementarity of trackers and more suitable to fusion issues (Section 4.2).

4.1. Global Evaluation

The global evaluation follows the protocol and metrics defined in the VOT Challenge [34]. One of its interesting features is to propose a specific procedure to address drifting phenomena and exploit fully the annotated sequences. This is done simply by reinitializing the tracker 5 frames after a drift is detected, assuming that the new starting frame is uncorrelated with the previous ones to reduce the estimation bias. A drift is indicated by an overlap with null value, the overlap between two boxes B and B' being defined as a classical intersection over union area ratio:

$$IoU(B, B') = \frac{|B \cap B'|}{|B \cup B'|} \tag{1}$$

The two metrics used in the evaluation are accuracy and robustness. Accuracy is measured by averaging the overlap between the track and the ground truth over the video base, after removing the first 10 frames following each reinitialization. Robustness is defined as the total number of drifts. As announced in the introduction, our study is focused on drift control, which is more naturally measured by robustness. However, achieving a good

accuracy may be critical in several applications requiring geometric precision (grasping, human computer interaction).

4.2. Local Evaluation

To complement the global evaluation of the trackers, we conducted a more fine-grained analysis, by making a frame by frame account of tracker drifts in each sequence of the video base. Figure 1 shows the drifting behaviors of several trackers.

Tracker drifts give valuable information about their local behaviors, in particular about their expected complementarity, i.e. their capacity to substitute each other when one of them drifts. This local complementarity can be exploited for fusion. Figure 2 shows the drifting times of 8 typical trackers (Section 7) obtained by applying the VOT protocol. Trackers show heterogeneous behaviors from one sequence to another: CT [36] and DPM [38, 39] drift many times more in *handball1* than in the other sequences. There are both correlated drifts as in *gymnastics* at frame 100, and also complementary behaviors: ASMS [43] does not drift when the others are drifting. Similarly, DSST [44] and ASMS are complementary in *bolt*. However, trackers have also redundant behaviors: many are correctly running in *gymnastics* from frame 1 to 100, which involves a certain computational cost. A good fusion scheme should be able to balance between complementarity and redundancy.

Complementarity of trackers. Bailer et al. [45] introduce the concept of fusibility, which compares the impact of a tracker when added to fusion or removed from it (positive/negative impact, gain, etc.). However, its fusibility measure is more fusion-method dependent since it is calculated after fusion.

We propose instead a complementarity measure between two or more trackers on a given database, called "incompleteness", independent from

Figure 1: Drifting behaviors of 6 trackers (NCC [35], KLT [29], CT [36], STRUCK [37], DPM [38, 39], MS [40]) for particular events like occlusion (a), brightness changes (e,f), blur (c,e), background clutter (c,d,f), scale (b) and appearance changes (d,e,f) on 6 videos taken from VOT2013 (a) [34], our GoPro videos (b), VOT2015 (c,d,e) [41] and VOT-TIR2015 (f) [42] using the VOT protocol. Active trackers are displayed, each outputs a color bounding box. The cyclist is occluded by a post (a), the motorbike reduces in size (b), the dinosaur is of the same color as the background (c), the fish warps and is very similar to the background (d), the motorbike is rotating with important brightness changes and blur (e), the quadrocopter is less textured in infra-red images than in RGB (f). Refer to Section 4.2.

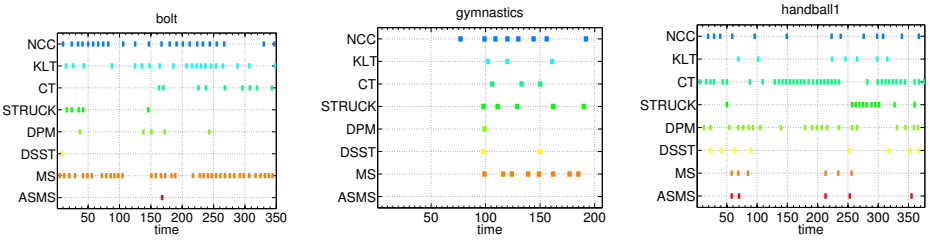


Figure 2: Drifts of 8 trackers (NCC [35], KLT [29], CT [36], STRUCK [37], DPM [38, 39], DSST [44], MS [40], ASMS [43]) in 3 difficult sequences taken from VOT2013 (*bolt*, *gymnastics*) [34] and VOT2015 (*handball1*) [41]. Drifts are represented by color dots, a different color for each tracker. See details in Section 4.2.

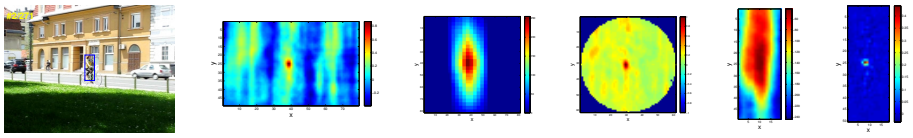
the fusion method. It requires only the drifts recorded individually from each tracker as shown in Figure 2. Contrary to Bailer et al., it does not measure the positive or negative contribution of one tracker relatively to a group of trackers but the complementarity of a whole. We express the complementarity of a set of trackers by its incompleteness defined below.

Incompleteness. The incompleteness measure expresses the inability of the trackers to compensate collectively for drifting, and is computed as the number of times when all trackers are simultaneously drifting at the same moment. At each time t, define a variable d_t^i for each indicating that tracker T_i for $i \in [1, M]$ is drifting.

We define the incompleteness I of a set of M trackers on a database of N frames as:

$$I = \sum_{t=1}^{N} \prod_{i=1}^{M} d_t^i \tag{2}$$

Remind that, in the evaluation protocol inherited from the VOT challenge, reinitialization takes place 5 frames after a drift is detected. This neutral interval can be interpreted as an uncertainty on the drifting time. A



(a) bicycle (b) (c) (d) (e) (f)

Figure 3: Score Maps of 5 trackers. (a) image from *bicycle* in VOT2013 [34], the blue bounding box is the target ground truth. (b), (c), (d), (e) and (f) are respectively the Score Maps of 5 trackers: NCC [35], KLT [29], STRUCK [37], CT [36] and DSST [44], corresponding to image (a). High scores are in red, low scores in blue. The evolution of the score distribution (intensity of the red stain, size, local maxima, etc.) gives information about the tracker good functioning. Details are in Section 5.1.

good complementarity of trackers means a low incompleteness.

5. On-line Tracker Failure Prediction

Many existing trackers can be used in our fusion framework: it is only required that they output a bounding box and an associated confidence value at each frame. Here, we describe an important stage of the fusion of multiple tracker outputs, which is to select the correct outputs. Our approach is to predict tracking failures from a set of M parallel trackers $\mathbf{T} = \{T_1, T_2, ...T_M\}$, either individually or collectively. At each time t, for each tracker T_i, $i \in [1, M]$, a state $s_t^i = \{0, 1\}$ is estimated, 1 for a valid state and 0 for a drifting state. Three ways to estimate this state are proposed in this section.

5.1. Behavioral Indicators (BI)

Trackers may have various functional and algorithmic structures. We present in this section three types of behavioral indicators that can adapt generically to most of them: confidence score, Score Map and specific indicators.

Confidence Score. At each frame, a tracker outputs a bounding box corresponding generally to a maximum confidence or prediction score in a search window depending on the tracker (correlation, classification or detection score). This confidence score represents the discriminating capacity of the inner object model and also can be interpreted as a behavioral indicator of the correct or poor operation of the tracker: the higher the score, the more reliable the prediction. It is expected that drift can be anticipated when this score decreases over time or goes under a given threshold.

Score Map. Rather than exploiting only its maximum value as it has been done in [19, 29, 30], our idea is to exploit the spatial distribution scores over the whole image or at least over a local window. This distribution which we call Score Map, can be peaked or flat, peaked meaning that the localisation is expected to be more precise and more reliable. Examples of Score Maps computed by different trackers are shown in Figure 3. We assume that tracker behavior can be characterized by this Score Map. For that, we need to extract some behavioral indicators based on the intensity of the response and its spatial distribution. We determine empirically these indicators by a closer analysis of the map distribution, testing several ones and choosing those that describe best its evolution. The exact indicators are detailed in Section 7.

Specific Indicators. For more complex trackers or for trackers for which the computation of Score Maps is either unreliable or not feasible due to their algorithmic structure, we defined ad hoc indicators: this is the case of the DPM based tracker [38, 39] which generates only sparse detections, and the ASMS [43] which uses an iterative scale estimation. Their exact definition is detailed in Section 7.

Estimate a tracker state using Behavioral Indicators. For each tracker T_i, $i \in [1, M]$, we construct a function DP_i capable of predicting its state s_t^i depending on its behavioral indicators ϕ_t^i at each time t, $DP_i : \phi_t^i \rightarrow s_t^i$. It consists of separating the behavioral indicator space in two states, the valid and invalid state by thresholding. Thresholds are obtained by an off-line evaluation of the predictive ability of DP_i and are chosen so as not to exceed a certain number of false drift predictions, the chosen values are detailed in Section 7. The off-line evaluation of the predictive ability of DP_i consists of comparing the drift predictions output by DP_i over a database to the real drifts measured by the VOT protocol. A prediction is counted as correct (good detection) when it has taken place 1-15 frames before a real drift otherwise is counted as a false alarm. The chosen range of 1-15 frames is large due to the nature of the drifts, that can be fast or slow.

5.2. Box Filtering (BF)

The principle is to predict a drift when the current estimated location of the target $\hat{B}_t^i$ from tracker T_i is very far from the previous estimated location $\hat{B}_{t-1}^{fusion}$ output by fusion: $dist(\hat{B}_{t-1}^{fusion}, \hat{B}_t^i) > width(\hat{B}_{t-1}^{fusion})$ then $s_t^i = 0$, otherwise $s_t^i = 1$. The distance $dist$ used is the euclidean distance between the center of the boxes. Contrarily to the overlap distance, it does not penalize a well-centered $\hat{B}_t^i$ which is very different in size from $\hat{B}_{t-1}^{fusion}$.

5.3. Box Consensus (BC)

The principle is to use information from other trackers to estimate the quality of current predictions, assuming that only few trackers in a given collections are likely to drift. We use a simple decision rule by analyzing the distribution of bounding boxes output by all trackers at time t,

$\hat{\mathbf{B}}_\mathbf{t} = (\hat{B}_t^1, \hat{B}_t^2, ...\hat{B}_t^M)$. We first cluster the predicted bounding boxes $\hat{\mathbf{B}}_\mathbf{t}$ as connected components derived from a similarity threshold: bounding boxes having an overlap > 0.5 are put in a same cluster. We then select a reference cluster whose center is the nearest to the previous estimated location $\hat{B}_{t-1}^{fusion}$. The boxes that do not belong to the selected cluster are declared drifting. We found this simple outlier detection sufficient in our experiments given the size of the pool of trackers used.

6. Proposed Fusion Approach

Our system is composed of a set of M trackers $\mathbf{T} = \{T_1, T_2, \dots T_M\}$ running in parallel. At the starting frame, all trackers receive a bounding box B_0 as input, which is the known localization of the target in the first image I_0. At each image I_t, each tracker T_i, $i \in [1, M]$, outputs an estimated bounding box $\hat{B}_t^i$. Our fusion approach consists of fusing their outputs $\hat{\mathbf{B}}_\mathbf{t} = (\hat{B}_t^1, \hat{B}_t^2, ...\hat{B}_t^M)$ with a dynamic selection of the good ones, to give the output of the system $\hat{B}_t^{fusion}$. This selection goes through an on-line evaluation of the trackers as described previously in Section 5. Moreover, passive or active fusion is handled in our framework, enabling to choose between different ways to correct the trackers. The fusion scheme is divided into 4 stages as illustrated in Figure 4 and detailed below.

6.1. Tracker Parallel Running

At each time t, each tracker T_i outputs one bounding box $\hat{B}_t^i$ and behavioral indicators ϕ_t^i as described in Section 5.1.

6.2. Tracker Selection by On-line Failure Prediction

The dynamic tracker selection is performed by Drift Predictors (DP) that estimate the state of each tracker. Each tracker T_i, $i \in [1, M]$, is associated

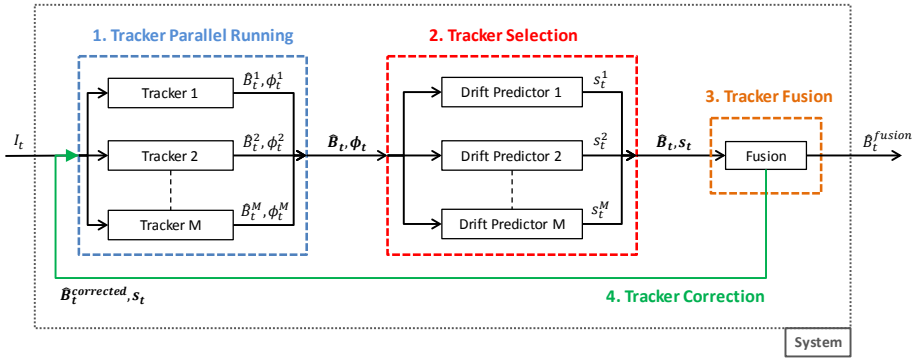


Figure 4: Our fusion approach divided into 4 principal stages (parallel running, selection, fusion and correction), refer to Section 6. For each new image I_t, each tracker T_i, $i \in [1, M]$, predicts the target location $\hat{B}_t^i$ and computes (optional) behavioral indicators ϕ_t^i that are indicators of the correct or poor functioning of the tracker. Vectors are in **bold.** $\hat{\mathbf{B}}_\mathbf{t} = (\hat{B}_t^1, ...\hat{B}_t^M)$ and/or $\boldsymbol{\phi_t} = (\phi_t^1, ...\phi_t^M)$ are used by Drift Predictors (DP) to predict the state of each tracker $s_t^i = \{0, 1\}$, 1 for a valid state and 0 for a drifting state. $\mathbf{s_t} = (s_t^1, ...s_t^M)$. Only bounding boxes provided by valid trackers are fused to give the output of the system $\hat{B}_t^{fusion}$. Drifting trackers can be corrected by updating or reinitializing their observation model with the correct location newly computed $\hat{B}_t^{i,corrected} = \hat{B}_t^{fusion}$. $\hat{\mathbf{B}}_\mathbf{t}^\mathbf{corrected} = (\hat{B}_t^{1,corrected}, ...\hat{B}_t^{M,corrected})$.

with a DP_i that estimates its state s_t^i at each time t using:

(i) Behavioral Indicators (BI) in Section 5.1,

(ii) or Box Filtering (BF) in Section 5.2,

(iii) or Box Consensus (BC) in Section 5.3,

(iv) or combined methods like BI+BF, which is an OR operation between their individual estimated state vector $\mathbf{s_t} = (s_t^1, s_t^2, ...s_t^M)$,

(v) or BI+BC, BI is first applied followed by BC: boxes are first eliminated by BI, then the remaining ones are clustered.

Simulation of an Ideal DP. We can simulate an *Ideal DP* for each tracker in order to measure the fusion performance in the best case scenario. For that, we use the overlap to the ground truth B_t and define a drift threshold fixed to 0.2. If $IoU(\hat{B}_t^i, B_t) < 0.2$, then DP_i outputs a drift: $s_t^i = 0$.

No DP. The selection step can be skipped.

6.3. Fusion Bounding Box Computation

The output of the system $\hat{B}_t^{fusion}$ is computed by merging the outputs of the K valid trackers, $K \leq M$. Recall that a valid tracker T_i is defined by a valid state $s_t^i = 1$. Two methods are used to compute $\hat{B}_t^{fusion}$:

(i) Average (Avg): by averaging the coordinates of the K valid boxes.

(ii) Center of gravity (Grav): by weighting the K valid boxes in relation to the euclidean distance to the other boxes. Define d_{ij} the distance between the centers of box i and box j. Weight w_i of box i is:

$$w_i = \frac{\frac{1}{\sum_{j \neq i} d_{ij}}}{\sum_{k=1}^{K} \frac{1}{\sum_{j \neq k} d_{kj}}}$$

If there is no correct box left after the selection step, then $\hat{B}_t^{fusion} = \hat{B}_{t-1}^{fusion}$.

6.4. Tracker Correction

After calculating $\hat{B}_t^{fusion}$, a tracker correction step is possible. In a passive approach, this step is skipped (P). In the active one, the invalid trackers can be corrected using the outputs of the valid ones. Indeed, drifting trackers are more likely to fail in the next frames since their object models are polluted by bad updates. Thus, two types of correction can be applied to the object model: update or reinitialization. Reinitialization is useful when the tracker has definitely drifted out of the target area. However, reinitialization means restarting the tracker from scratch, resetting the past knowledge of the target appearance and variability. A less drastic way is to update the object model by providing the right location and features to track when the tracker is drifting. Therefore, we present three exclusive ways to correct the object model:

(i) Update the model of drifting trackers only (UD) with the right location output by fusion $\hat{B}_t^{fusion}$,

(ii) Update the model of all trackers (UA) with $\hat{B}_t^{fusion}$,

(iii) Reinitialize the model of drifting trackers only (RD) with $\hat{B}_t^{fusion}$.

Define the *corrected bounding boxes* used to update or reinitialize the trackers: $\mathbf{\hat{B}_t^{corrected}} = \left\{\hat{B}_t^{i,corrected}, i \in [1, M]\right\}$. $\hat{B}_t^{i,corrected}$ corresponds to $\hat{B}_t^{fusion}$ for an update or a reinitialization, $\hat{B}_t^i$ otherwise (no correction). $\mathbf{\hat{B}_t^{corrected}}$ are looped back to the trackers as input as illustrated in Figure 4.

6.5. Fusion Configurations

One fusion configuration of the system consists of choosing one method for each step of the fusion (selection, fusion, correction). In total, 46 possible fusion configurations can be experimented.

7. Material and Implementation

Databases. We evaluate our fusion approach on 3 databases with varied objects and scenes in challenging conditions (camera moving and zoom, brightness changes, occlusions, deformable objects, fast appearance changes and object movements, etc.):

(a) **VOT2013**+ contains 12 videos from VOT2013 [34] completed with 1 video from the KITTI Vision Benchmark Suite [46], and 5 other videos we have taken from a GoPro camera embarked on a vehicle, available on request. VOT2013+ contains a total of 25 objects (6525 frames).

(b) **VOT2015** [41] contains 60 videos (21455 frames).

(c) **VOT-TIR2015** [42] contains 20 thermal infra-red videos (11269 frames).

(d) **OTB-100** (`http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html`) contains 98 videos of various length (58264 frames).

Trackers. The considered trackers in this work consist of a mix of standard and state of art trackers based on different target features and object model:

(a) NCC (Normalized Cross-Correlation) [35], uses a gray-scale target patch taken from the first frame and a normalized cross-correlation.

(b) KLT (Kanade Lucas Tomasi Tracker) [29], is the optical flow tracker from the TLD [29] but without scale estimation.

(c) CT (Compressive Tracking) [36], uses an object model composed of multiple object-background classifiers, based on Haar-like features.

(d) STRUCK (Structured Output Tracking with Kernels) [37], is a structured output SVM framework using Haar-like features and gaussian kernels.

(e) DPM based tracker, combines an object detector based on trained deformable part models [47] using Dubout and Fleuret implementation [38] and a Kalman filter [39].

(f) DSST (Discriminative Scale Space Tracker) [44], learns a multi-scale correlation filter based on PCA-HOG features.

(g) MS (Meanshift) [40], uses an HSV color histogram from the initial target patch and a mean-shift on the back projection image of the histogram.

(h) ASMS (Adaptive Scale mean-shift) [43], uses a RGB color histogram weighting that exploits the target neighborhood and estimates the target position and scale using mean-shift and a backward consistency check.

(i) CCOT-HOG (Continuous Convolution Operator Tracker)[48], learns a multi-scale correlation filter on HOG features and searches the best matching pattern through a continuous interpolation in spatial domain.

We collected source codes of open source trackers[2] and integrated them into C++ multi-threaded fusion framework running on an Intel Xeon 4 core 2.80 GHz CPU with 8 GB RAM. Their individual performances (robustness and accuracy defined in VOT, and speed, the number of frames proceeded per second) on the 3 databases can be found in Table 2.

[2] `https://github.com/votchallenge/vot-toolkit` for NCC, `https://github.com/gnebehay/OpenTLD` for KLT, `http://www4.comp.polyu.edu.hk/~cslzhang/CT/CT.htm` for CT, `https://github.com/samhare/struck` for STRUCK, `https://github.com/fanxu/ffld` for DPM, `https://github.com/klahaag/cf_tracking` for DSST, `http://docs.opencv.org/2.4/modules/video/doc/motion_analysis_and_object_tracking.html` for MS, `https://github.com/vojirt/asms` for ASMS. `https://github.com/martin-danelljan/Continuous-ConvOp` provides a Matlab implementation of CCOT that has been integrated in the C++ framework.

Table 2: Tracker individual performances on VOT2013+, VOT2015, VOT-TIR2015 and OTB-100. The number of sequences and total number of frames for each base are indicated in parenthesis. *rob* is the robustness (total number of drifts), *acc* is the accuracy (average overlap) and *speed*, the number of frames proceeded per second. Best robustness, accuracy and speed per base are in **red**, the second best results in blue. See details in Section 7.

	VOT2013+ (25, 6525)			**VOT2015** (60, 21455)			**VOT-TIR2015** (20, 11269)			**OTB-100** (98, 58264)		
	rob	acc	speed	rob	acc	speed	rob	acc	speed	rob	acc	speed
NCC [35]	131	0.59	**612**	447	**0.54**	**1046**	149	**0.66**	**1310**	602	0.61	**1201**
KLT [29]	69	0.44	58	253	0.41	57	123	0.34	103	349	0.44	73
CT [36]	36	0.44	28	221	0.42	22	144	0.51	31	354	0.48	26
STRUCK [37]	38	0.49	23	156	0.46	19	140	0.54	20	182	0.56	22
DPM [38, 39]	57	0.47	24	525	0.4	49	186	0.49	48	740	0.41	46
DSST [44]	18	**0.61**	190	170	**0.54**	177	47	0.64	209	120	**0.67**	325
MS [40]	194	0.27	606	654	0.32	434	297	0.24	422	1205	0.36	568
ASMS [43]	30	0.44	290	112	0.5	236	89	0.53	610	197	0.56	355
CCOT-HOG [48]	**12**	0.56	8	**102**	0.51	7	**26**	0.64	9	**68**	0.66	10

Behavioral Indicators. There are 2 groups of trackers: ones with a Score Map (NCC, KLT, CT, STRUCK, DSST, MS, CCOT) and ones with specific indicators (DPM, ASMS). DP based on BI are built by fixing empirically the thresholds so that the number of generated false alarms is low.

For those that use the Score Map, most of the maps are rectangular local maps of fixed size or depending on the target size (w, h).

NCC uses three indicators computed on the local correlation map: $dx_{max} = (x_{max}(t) - x_{max}(t-1))/w$, $dy_{max} = (y_{max}(t) - y_{max}(t-1))/h$, $dmax =$

$(max(t-1) - max(t))/max(t-1)$. $x_{max}(t)$ and $y_{max}(t)$ are the x-y coordinates of the maximum score of the map $max(t)$ at time t. The BI thresholds are respectively 0.2, 0.2 and 0.2.

KLT computes a local map, each location (x, y) of the map stores the number of matched points cumulated in a window (w, h) centered on (x, y). Three indicators are used: $dx_c = (x_c(t) - x_c(t-1))/w$, $dy_c = (y_c(t) - y_c(t-1))/h$, $d_{spotsize} = spotsize(t) - spotsize(t_0)$. $x_c(t)$ and $y_c(t)$ are the x-y coordinates of the center of the spot in the map, defined by values > 200 at time t. The number $spotsize(t) = \#(pixels > 200)/\#(pixels > 0)$ counts the number of values > 200 normalized to the number of positive values at time t. The BI thresholds are respectively 0.35, 0.35 and 1.5.

CT computes a map of sums of likelihood ratios and uses one indicator: $d_{area} = (area(t) - area(t-1))/area(t-1)$. The area $area(t)$ counts the number of scores higher than $thr(t) = max - 0.1 * (max - min)$ with max and min being the maximum and minimum values of the map at time t. The BI threshold used is 0.7.

STRUCK computes a map of SVM classification scores and uses two indicators: $varxy_{10max} = \sqrt{varx_{10max}{}^2 + vary_{10max}{}^2}/min(w, h)$ and $dxy_{max} = \sqrt{dx_{max}{}^2 + dy_{max}{}^2}/min(w, h)$. $varxy_{10max}$ is the variance (in pixels) of the location of the first 10 maxima of the map at time t. The number dxy_{max} is the distance between the location of the maximum value in the map at time t and $t-1$. The BI thresholds used are respectively 0.5 and 0.5.

DSST computes a correlation map and uses one indicator: the Peak to Sidelobe Ratio PSR [49]. The BI threshold is 8.

MS computes one indicator on the back projection map: $fbratio = (max_F - \mu_B)/\sigma_B$ where max_F is the maximum score of the foreground (box), the mean μ_B and the standard deviation σ_B are computed on the neighbor back-

ground of size $3 * (w, h)$. The BI threshold is 8.
CCOT-HOG computes a correlation map at an optimal scale and exploits a single indicator: the relative ratio between the current max value and the maximal score on the first frame which is expected to be between 0.25 and 1.25.

For the trackers that do not use the Score Map, we compute specific indicators. DPM uses two indicators: $dbest = score(best1) - score(best2)$ and $obest = IoU(best1, best2)$. $dbest$ is the difference of scores between $best1$ and $best2$. $best1$ is the detection with the highest score and has an overlap > 0.3 with the previous target location. $best2$ is the detection with the best score below $best1$. The BI thresholds are respectively 0.05 and 1.
ASMS uses two indicators: bhattacharyya coefficients of the foreground $bhatta_f$ and background $bhatta_b$ at the target predicted position. The BI thresholds are respectively 0.4 and 0.3.

The threshold values have been settled empirically on the VOT2013+ dataset by collecting the indicator values and computing corresponding precision/recall curves for drift prediction. The target operating point was a recall (i.e. a good drift prediction rate) of approximately 0.75. We have checked that the threshold values generated similar precision/recall operating points on the VOT2015 database. The same values have been used for all the experiments.

8. Experimental Results

We conducted thorough experiments from our available data and software to answer two different questions: the impact of the selection and the correction step in our fusion approach and what is the best fusion configuration

in Section 8.1; the best combination of trackers achieving high robustness at low cost in Section 8.2.

8.1. What is the best fusion configuration?

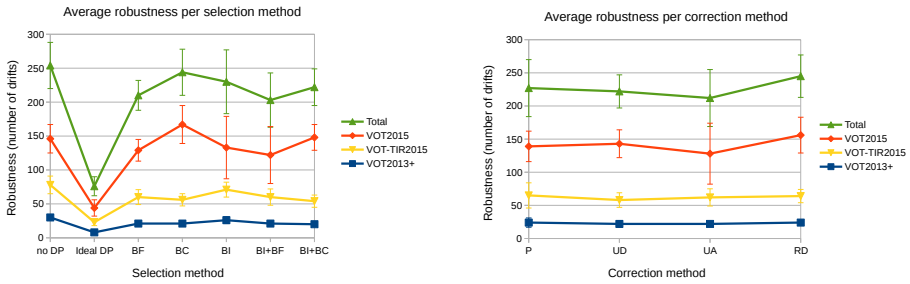
In order to circumscribe the number of combinations evaluated and avoid useless computations, we first selected the 4 trackers among the 9, with comparable individual performances: CT, STRUCK, DSST and ASMS. Preliminary experiments with low performing trackers (NCC, KLT, DPM and MS) showed no improvement when fusing them with the other trackers.

For this section, we experimented the fusion of 3-tracker combinations using the 4 selected trackers over the 46 fusion configurations, some fusion configurations only make sense when the number of combined trackers is higher than 2: CT+STRUCK+DSST, CT+STRUCK+ASMS, CT+DSST+ASMS and STRUCK+DSST+ASMS.

We eliminated the 23 fusion configurations from the 46 that use the center of gravity fusion (*Grav*). Experiments showed that an average fusion (*Avg*) gives fewer drifts than a center of gravity fusion (*Grav*).

The following fusion robustness results in Figure 5 and Table 3 are computed by averaging the robustness of the 4 selected tracker combinations over all the fusion configurations using the specified selection and/or correction method.

Impact of the Selection Step in Fusion. Figure 5a shows the average fusion robustness obtained using each of the selection method (*no DP, Ideal DP, BF, BC, BI, BI+BF, BI+BC*) on the 3 databases. Designing a good DP associated with a simple fusion strategy greatly improves the tracking robustness as the *Ideal DP* results show. Using one of our 5 designed selection



(a) Selection (b) Correction

Figure 5: Average fusion robustness obtained for the 3 databases individually and summed (Total). (a) shows the performances by using different tracker selection methods (*no DP, Ideal DP, BF, BC, BI, BI+BF, BI+BC*). (b) shows the performances by using different tracker correction methods (*P, UD, UA, RD*). For each method, the robustness is averaged over the 4 combinations (CT+STR+DSST, CT+STR+ASMS, CT+DSST+ASMS, STR+DSST+ASMS) and the fusion configurations of the corresponding selection method in (a) and correction method in (b). The standard deviation is provided.

methods (*BF, BC, BI, BI+BF, BI+BC*) globally outperforms fusion without the selection step (*no DP*) as the figures show in *Total* (total of the 3 databases). The best method is *BI+BF* followed by *BF*. There is a slight advantage in mixing selection methods. Bad results of *BI* are partly due to the limited capacity of the trackers to provide sufficiently reliable scores that leave no doubt about the tracker behavior. Indeed, the capacity of the appearance model to absorb the target appearance variations remains unclear and differs from one tracker to another. It is more efficient to combine *BI* with other methods.

Impact of the Correction Step in Fusion. Figure 5b shows the average fusion robustness using each of the correction methods (*P, UD, UA, RD*) on the 3 databases whatever the chosen selection method except the *Ideal DP* config-

urations that are removed from the calculation. In *Total* of the 3 databases, the best correction method whatever the utilized selection method is UA, followed by UD, then P. The worse is RD. One of the reasons of the bad impact of reinitialization is that the appearance used to setup a new model occurs simultaneously with the drifting phenomenon (occlusion, illumination, aspect change), which is often a transient event in the sequence. Thus, model update is preferable to reinitialization.

Determining the best fusion configuration. Table 3 gives the average fusion robustness over the total of the 3 databases (one single big database) with the corresponding standard deviation for the different selection+correction methods. When there is *no DP*, the only correction that can be made is *UA*. Passive fusion *P* is considered as invalid when using *BI*, *BI+BF* or *BI+BC* because DP based on *BI* are incapable of detecting a drift after it has occurred (linked to the inner working of the model), and thus need an active approach to be operational. The results show that with a perfect command on the selection step (*Ideal DP*), active fusion (*UD, UA, RD*) would outperform passive fusion (*P*). In that case, *UD* would be the best active fusion method. In practice, the best configurations are 2 active configurations (*BI+BF+UA, BI+UA*) followed by 2 passive ones (*BF+P, BC+P*). Active fusion gives generally better results but is also more fluctuating (important standard deviation).

8.2. What is the best combination of trackers?

Even if we have full control on the selection and correction step, i.e. capable of selecting properly the trackers for fusion and correcting the drifting ones, there is one parameter left that influences on the fusion performance: the choice of the trackers. Thus, the question is to know how to determine

Table 3: Average fusion robustness per selection (*no DP, Ideal DP, BF, BC, BI, BI+BF, BI+BC*) and correction method (*P, UD, UA, RD*) on the total of the 3 databases. For each selection+correction method, the fusion robustness is averaged over the 4 combinations (CT+STR+DSST, CT+STR+ASMS, CT+DSST+ASMS, STR+DSST+ASMS) and the fusion configurations of the corresponding selection+correction method. The standard deviation is provided. The best robustness performances (except the *Ideal DP*) are in **red**, and the best one is in addition **underlined**. See details in Section 8.1.

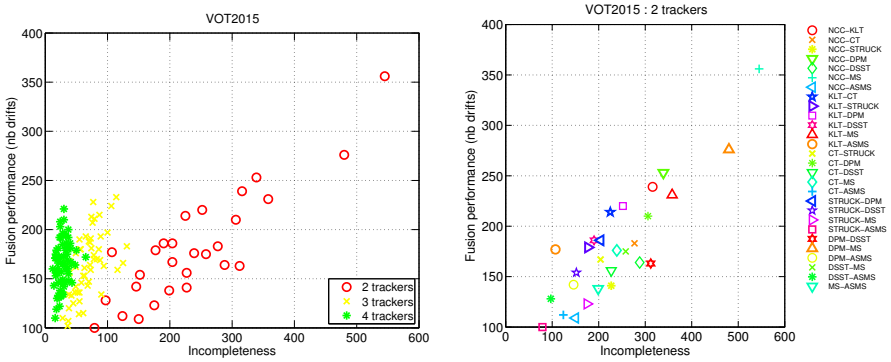
	no DP	Ideal DP	BF	BC	BI	BI+BF	BI+BC
P	282±28	86±20	**200±20**	**201±17**			
UD		67±12	212±18	243±26	241±24	208±26	210±20
UA	227±14	78±14	216±35	249±17	**191±62**	**180±62**	213±35
RD		71±9	213±21	283±16	259±32	223±25	244±19

the best group of trackers to fuse for a given database. One research area to explore is the complementarity of the trackers. Our hypothesis is that, the more complementary the trackers, the better the fusion.

For the following experiments, we retain one of the best fusion configurations from Section 8.1: *BC+P*. The reason why we choose a passive configuration is that the active ones give more fluctuating performance from one combination to another.

Complementarity of trackers. Figure 6 plots the fusion performance w.r.t. the incompleteness on VOT2015 of all the possible combinations of 2-4 trackers from the 8 available in (a) and details the combinations of 2 trackers in (b). The fusion performance seems to be correlated to the incompleteness: the lower the incompleteness, the better the robustness. Thus, the incompleteness can be used as an empirical way to determine the most efficient

combinations of trackers on a database for a small number of fused trackers. We also observe that the variations of the fusion performance decrease with the number of fused trackers. However, the maximum performance is reached for 2 trackers on VOT2015. Thus, adding more trackers does not necessarily improves robustness. The combination of very heterogeneous trackers with very different individual performance can affect significantly the fusion results. It requires the capacity to manage the numerous drifts of the bad trackers.



(a) 2-4 tracker combinations (b) 2-tracker combinations

Figure 6: Fusion robustness w.r.t. the incompleteness of tracker combinations (NCC [35], KLT [29], CT [36], STRUCK [37], DPM [38, 39], DSST [44], MS [40], ASMS [43]) using the fusion configuration *BC+P* on VOT2015. (a) shows all the combinations of 2-4 trackers among the 8. (b) shows all the combinations of 2 trackers. See details in Section 8.2.

Best tracker combinations for each database. Table 4 shows the combination with the highest robustness for each database using the selected fusion configuration (*Ours*). We do the same with another fusion approach quite similar to our work: Bailer's passive fusion [4] (*Bailer*), by executing its on-

line available code[3] using the *basic approach*. We computed as a percentage the *Gain* obtained relatively to the best individual robustness of the combination, tagged as *Indiv* in the table. Our fusion approach outperforms the passive approach of Bailer et al. for the 3 databases. One has to notice that the best combination is not the same for all databases; however the performance *Gain* using the best fusion combination is always positive, justifying the usefulness of a fusion approach.

We provide further discussion of these results and extra experimentation in the following section.

Table 4: Best robustness (*Robust*) from *Bailer*'s fusion and from *Ours* using the configuration *BC+P* on the 3 databases. *Best* indicates the corresponding combination of trackers. The percentages indicate the *Gain* of the fusion compared to the best individual tracker of the combination *Indiv*. D: DSST, A: ASMS, M: MS, S: STRUCK, C: CT, P: DPM. Best results are in **red**. See details in Section 8.2.

	VOT2013+		**VOT2015**		**VOT-TIR2015**	
	Best	Robust	Best	Robust	Best	Robust
Indiv.	D	18	A	112	D	47
Bailer	D+M	16 (11%)	S+D+A	108 (4%)	S+D+A	45 (4%)
Ours	D+A	**13 (28%)**	S+A	**100 (11%)**	C+P+D+A	**39 (17%)**

8.3. Discussion and further directions of work

The fundamental idea of this study was to exploit on-line tracker drift prediction (DP) in an active fusion process. This was motivated by the fact

[3]https://sites.google.com/site/alainpagani/

(BF+P)

(BI+BF+UA)

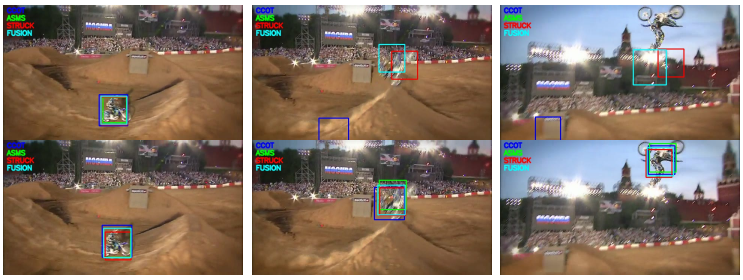


Figure 7: Example of the impact of target confusion on two fusion approaches. *Motor-Rolling* sequence from OTB-100 at times 61, 71 and 85.

(BF+P)

(BI+BF+UA)

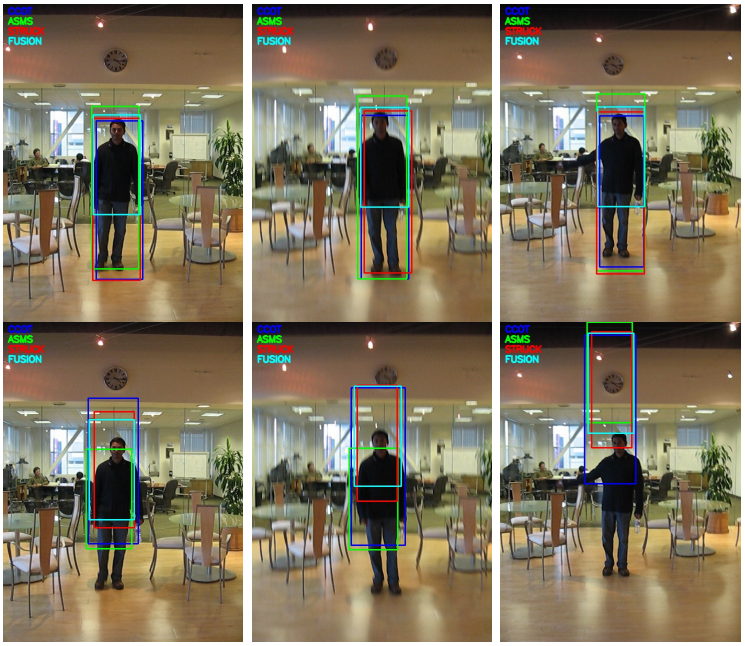


Figure 8: Example of the impact of badly localized bounding boxes over targets on two fusion approaches. *Human2* sequence from OTB-100 at times 760, 790 and 828.

Table 5: Conditional performance on OTB-100 dataset for 4 different fusion configurations of three different trackers (ASMS, STRUCK and DSST). IV = Illumination Variation, SV = Scale Variation, OCC = Occlusion, DEF = Deformation, MB = Motion Blur, FM = Fast Motion, IPR = In-Plane Rotation, OPR = Out-of-Plane Rotation, OV = Out-of-View, BC = Background Clutters, LR = Low Resolution.

	OPR	BC	OCC	MB	SV	IV	IPR	LR	OV	FM	DEF
BF+P	80	54	76	58	94	72	73	15	25	57	45
BC+P	**55**	**32**	**53**	**52**	75	**45**	**51**	9	**21**	60	44
BI+UA	73	34	62	73	**71**	49	73	**6**	25	87	48
BI+BF+UA	82	54	82	54	99	71	79	20	25	**54**	**40**

that having access to an Ideal DP greatly improved the overall performance of the fused solution. Designing such a DP and controlling its impact appeared to be quite tricky in practice. This section discusses some of the issues that have been raised during this study.

Influence of video features. Table 5 analyzes the robustness of a three-tracker configuration according to specific video attributes as proposed in (`http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html`). The figures show that the more complex fusion strategies (BI+) seem to be useful only on few videos from the OTB-100 dataset. We couldn't find however any clear and meaningful correlation between the fusion strategies and the video features which does not depend on the nature of the trackers involved.

Repertoire of trackers. Table 6 shows the influence of performance heterogeneity when fusing several trackers. The results illustrate the fact that fusion strategies seem mostly interesting when trackers are of comparable performance but rely on different design principles. The CCOT and DSST

Table 6: Influence of the quality of individual trackers: STRUCK(S), ASMS (A), DSST(D), CCOT(C) in the fusion process of two configurations on OTB-100 dataset.

Fusion strategy	Number of drifts	
	S+A+D	**S+A+C**
Indiv.	182/197/120	182/197/**68**
Bailer Basic	105	88
Bailer Weighted	116	75
BC+P	119	102
BF+P	**93**	79
BI+UA	128	102
BI+BF+UA	129	99

trackers both exploit a multi scale discriminant pattern correlation approach but exhibit different performances: when DSST is fused with STRUCK and ASMS, which are trackers of comparable performance regarding robustness, three fusion strategies are able to increase the overall system performance. This is not true for CCOT, where all fusion strategies give lower performance than the best individual tracker. Table 6 shows a comparison with two other fusion schemes proposed by Bailer et al. [4]: the *basic approach* and the *weighted approach* exploiting explicitly an estimate of the average tracking accuracy. This last strategy is able to control the performance loss, but do not provide any improvement when compared to the best tracker (CCOT).

Fusion vs. single. To further evaluate the impact of fusion, we have identified the sequences where fusion appeared to be profitable vs. damageable (Table 7), with the hypothesis than each sequence in the OTB-100 bench-

Table 7: Number of sequences from OTB-100 dataset where fusion has a positive/negative impact when compared to the best individual tracker. Only sequences with a difference of more than 2 drifts are reported. The names of the best/worse sequences are provided with the drift number difference.

Fusion strategy	Number of sequences better/worse	
	S+A+D vs. D	**S+A+C vs. C**
Bailer Basic	11/9 Skating2(-5) / Tiger1(+3)	3/8 David(-4) / Human4(+7)
Bailer Weighted	8/7 David(-5) / BlurFace(+11)	6/11 MotorRolling(-2) / Panda(+3)
BC+P	5/2 Skating2(-7) / Suv(+4)	1/1 MotorRolling(-4) / BlurFace(+8)
BF+P	11/1 David(-6) / Human9(+6)	9/8 David(-5) / BlurOwl(+7)
BI+UA	12/14 BlurOwl(-12) / Car1(+7)	7/13 MotorRolling(-5) / Car1(+11)
BI+BF+UA	12/14 BlurOwl(-9) / Car1(+7)	10/11 MotorRolling(-5) / Human2(+11)

mark is a typical example of video feature mixture (for example the *BlurOwl* sequence is marked with SV-MB-IPR-FM features, whereas *Car1* is marked with LR-IV-SV-BC). This table shows that fusion improves performance on sequences that are difficult for the best individual tracker (*Skating2* and *BlurOwl* for DSST, *MotorRolling* for CCOT), but may also degrade it for several sequences. The degradation is however less consistent over sequences, and does not seem to be correlated with the video features — performance on *BlurOwl* for instance can be either increased or lowered depending on the tracker pool.

Influence of tracker behavior on fusion. To further understand the reasons of performance increase or degradation of the fusion strategies, one has to take into account the individual tracker behaviors. We propose to classify individual tracker abnormal behavior in two different categories: target confusion and bad localization. The first category happens when the target model is attracted by a pattern with an appearance similar to the target and gets stuck on it. The model adaptation mechanism used to control target appearance variability, often a first order dynamical model, gets trapped on a wrong pattern. Figure 7 shows an example of such an attraction (see CCOT box in the first row). BF+P is a simple strategy that does not exploit any correction nor selection, but a box averaging (see 6.3): when tracker outputs are far apart, fusion generates a wrong bounding box. BI+UA+BF, however, contains a correction mechanism able to reposition trackers: the second row of Figure 7 shows that this fusion strategy is able to counter CCOT target confusion.

The second category describes trackers that provide bounding boxes with translation, scale or orientation errors. Figure 8 shows the behavior of the same strategies on trackers that provide in-homogeneously localized bound-

ing boxes: the correction mechanism in this case has a negative impact, and progressively drives trackers to be collectively attracted on a wrong pattern. These two examples show how fusion strategies may be able to correct tracker behavior, but also that none — at least those evaluated in this paper — can be considered universally better.

Role of selection. On-line tracker selection is a very useful step in the fusion strategy but heavily depends on a high level of drift prediction accuracy. We proposed two different approaches to do so: an analysis of predicted box distribution (BF and BC), and a self-diagnosis indicator of the appearance model quality (BI). The first one appeared to be simple and rather effective, but failed in case of large camera motion; the second was difficult to calibrate directly from the inner data accessible in each tracker and did not show substantial improvements in the fusion process when used alone. Both selection strategies exploit low dimensional tracker features: other decision schemes exploiting more directly image features and their dynamics could improve drift prediction accuracy, but would probably necessitate a larger set of training data.

Role of correction. One limitation of our approach is the lack of adaptability of the reinitialization and updating schemes: all the trackers are either reinitializing or updating their model from the fused prediction depending on the chosen strategy. Although UA and UD appeared globally better than RD (UA $>$ UD $>$ RD), a better flexibility of the correction scheme could improve their robustness (see the difference of correction impact on the two examples of Figure 7 and Figure 8).

Practical limitations. The main practical limitation to make use of the various fusion approaches described in this paper on a given set of trackers is the implementation of the BI based selection part (indicator design and

drift detection threshold). All the other fusion configuration parameters (P,BF,BC,UA,UD,RD) are tracker independent.

A basic approach to design a drift predictor is to use the confidence coefficient or the likelihood often computed in the processing chain to locate the target position from a local score map (see example in Figure 3). Rather generic indicators can be computed from the score distribution in the map, as has been done for DSST or STRUCK, for instance (see sections 5 and 7). However, as Figure 3 also illustrates, the score map shape is highly tracker dependent, and sometimes not that easy to exhibit given a tracker code — not speaking of the software interoperability issues to integrate the code in a common environment.

9. Conclusion

The work described in this paper focused on the design of good strategies for the on-line fusion of trackers. The emphasis was on controlling the overall robustness of tracking measured as a number of drifting events, i.e. the number of times the target is lost when applied on a given database. Trackers deal with critical situations differently (illumination, occlusion, appearance changes, camera motion); the idea was to exploit their complementarity on various fusion strategies.

Fusion can operate at two levels: by selecting the appropriate set of good trackers and/or by correcting either their output or their inner state. Drift prediction based on various features has been proposed and more specifically studied as a key component of the selecting step. The overall fusion strategies resulted in 46 different schemes that have been extensively evaluated on 3 databases (VOT2013+, VOT2015, VOT-TIR2015 and OTB-100)

and a repertoire of 8 trackers with available source code (NCC, KLT, CT, STRUCK, DPM, DSST, MS, ASMS, CCOT).

The results of the experiments can be summarized as a series of recommendations (What trackers use? What to fuse and how?) when trying to apply on-line fusion given a target database or application context and a set of trackers with their individual robustness evaluation on the database.

(1) Fusion is helpful when fusing trackers with comparable individual performance (robustness) and gives an important gain. By contrast, fusing very heterogeneous trackers can be harmful when noisy outputs contaminate the other trackers and degrade their behavior.

(2) A selection step is useful, the simplest methods based on bounding box reasoning — temporal filtering and consensus — leading to comparable results to more specific methods trying to give independently a hint of each individual tracker behavior (score or likelihood maps).

(3) The correction step is sensitive to individual tracker behaviors: passive fusion cannot recover from target confusion, and active fusion may be contaminated by bad target localization.

Fusion performance also depends on tracker complementarity besides their individual performance. To quantify the complementarity of a set of trackers, we defined an incompleteness measure based on off-line individual drifts that is predictive (with a certain variance) of the fusion performance of 2 to 4 trackers. This measure can be used to choose the best combination of trackers for a given database.

References

[1] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, M. Shah, Visual tracking: an experimental survey, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (7) (2014) 1442–1468.

[2] Y. Wu, J. Lim, M.-H. Yang, Online object tracking: A benchmark, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2013, pp. 2411–2418.

[3] M. Kristan, J. Matas, A. Leonardis, T. Vojíř, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, L. Čehovin, A novel performance evaluation methodology for single-target trackers, IEEE transactions on pattern analysis and machine intelligence 38 (11) (2016) 2137–2155.

[4] C. Bailer, A. Pagani, D. Stricker, A superior tracking approach: Building a strong tracker through fusion, in: European Conference on Computer Vision, Springer, 2014, pp. 170–185.

[5] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, A. V. D. Hengel, A survey of appearance models in visual object tracking, ACM Trans. Intell. Syst. Technol. 4 (4) (2013) 58:1–58:48. doi:10.1145/2508037.2508039.
URL http://doi.acm.org/10.1145/2508037.2508039

[6] N. Xiong, P. Svensson, Multi-sensor management for information fusion: issues and approaches, Information fusion 3 (2) (2002) 163–186.

[7] Y. Bar-Shalom, P. K. Willett, X. Tian, Tracking and data fusion: A Handbook of Algorithms., Yaakov Bar-Shalom, 2011.

[8] M. H. Khan, M. F. Valstar, T. P. Pridmore, A generalized search method

for multiple competing hypotheses in visual tracking, in: International Conference on Pattern Recognition, IEEE, 2014, pp. 2245–2250.

[9] J. Kwon, K. M. Lee, Tracking by sampling trackers, in: International Conference on Computer Vision, IEEE, 2011, pp. 1195–1202.

[10] O. Khalid, J. C. SanMiguel, A. Cavallaro, Multi-tracker partition fusion, IEEE Transactions on Circuits and Systems for Video Technology 27 (7) (2017) 1527–1539. doi:10.1109/TCSVT.2016.2542699.

[11] J. Zhang, S. Ma, S. Sclaroff, Meem: Robust tracking via multiple experts using entropy minimization, in: European Conference on Computer Vision, Springer, 2014, pp. 188–203.

[12] T. Penne, C. Tilmant, T. Chateau, V. Barra, Markov chain monte carlo modular ensemble tracking, Image and Vision Computing 31 (6) (2013) 434–447.

[13] J. H. Yoon, D. Y. Kim, K.-J. Yoon, Visual tracking via adaptive tracker selection with multiple features, in: European Conference on Computer Vision, Springer, 2012, pp. 28–41.

[14] E. Erdem, S. Dubuisson, I. Bloch, Visual tracking by fusing multiple cues with context-sensitive reliabilities, Pattern Recognition 45 (5) (2012) 1948 – 1959.

[15] H. Nam, B. Han, Learning multi-domain convolutional neural networks for visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4293–4302.

[16] Y. Zhou, X. Bai, W. Liu, L. J. Latecki, Similarity fusion for visual

tracking, International Journal of Computer Vision 118 (3) (2016) 337–363.

[17] A. Yilmaz, O. Javed, M. Shah, Object tracking: A survey, Acm computing surveys (CSUR) 38 (4) (2006) 13.

[18] N. T. Siebel, S. Maybank, Fusion of multiple tracking algorithms for robust people tracking, in: European Conference on Computer Vision, Springer, 2002, pp. 373–387.

[19] B. Stenger, T. Woodley, R. Cipolla, Learning to track with multiple observers, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 2647–2654.

[20] J. Santner, C. Leistner, A. Saffari, T. Pock, H. Bischof, Prost: Parallel robust online simple tracking, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 723–730.

[21] B. Zhong, H. Yao, S. Chen, R. Ji, T.-J. Chin, H. Wang, Visual tracking via weakly supervised learning from multiple imperfect oracles, Pattern Recognition 47 (3) (2014) 1395 – 1410.

[22] S. Moujtahid, S. Duffner, A. Baskurt, Coherent selection of independent trackers for real-time object tracking, in: International Conference on Computer Vision Theory and Applications (VISAPP), 2015, pp. 584–592.

[23] H. Grabner, H. Bischof, On-line boosting and vision, in: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, Vol. 1, IEEE, 2006, pp. 260–267.

[24] S. Moujtahid, S. Duffner, A. Baskurt, Classifying global scene context for on-line multiple tracker selection, in: British Machine Vision Conference (BMVC), 2015.

[25] P. Zhang, J. Wang, A. Farhadi, M. Hebert, D. Parikh, Predicting failures of vision systems, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 3566–3573.

[26] S. Daftry, S. Zeng, J. A. Bagnell, M. Hebert, Introspective perception: Learning to predict failures in vision systems, in: Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, IEEE, 2016, pp. 1743–1750.

[27] H. Grimmett, R. Paul, R. Triebel, I. Posner, Knowing when we don't know: Introspective classification for mission-critical decision making, in: Robotics and Automation (ICRA), 2013 IEEE International Conference on, IEEE, 2013, pp. 4531–4538.

[28] H. Grimmett, R. Triebel, R. Paul, I. Posner, Introspective classification for robot perception, The International Journal of Robotics Research 35 (7) (2016) 743–762.

[29] Z. Kalal, K. Mikolajczyk, J. Matas, Tracking-learning-detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (7) (2012) 1409–1422.

[30] T. Vojir, J. Matas, J. Noskova, Online adaptive hidden markov model for multi-tracker fusion, Computer Vision and Image Understanding 153 (2016) 109–119.

[31] T. Vojir, J. Matas, J. Noskova, Online adaptive hidden markov model for multi-tracker fusion, CoRR abs/1504.06103.

[32] T. A. Biresaw, A. Cavallaro, C. S. Regazzoni, Tracker-level fusion for robust bayesian visual tracking, IEEE Transactions on Circuits and Systems for Video Technology 25 (5) (2015) 776–789.

[33] W. Zhong, H. Lu, M.-H. Yang, Robust object tracking via sparsity-based collaborative model, in: Computer vision and pattern recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 1838–1845.

[34] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Cehovin, G. Nebehay, G. Fernandez, T. Vojir, A. Gatt, et al., The visual object tracking vot2013 challenge results, in: International Conference on Computer Vision Workshop, IEEE, 2013, pp. 98–111.

[35] J. Lewis, Fast normalized cross-correlation, in: Vision interface, Vol. 10, 1995, pp. 120–123.

[36] K. Zhang, L. Zhang, M.-H. Yang, Real-time compressive tracking, in: European Conference on Computer Vision, Springer, 2012, pp. 864–877.

[37] S. Hare, A. Saffari, P. H. Torr, Struck: Structured output tracking with kernels, in: International Conference on Computer Vision, IEEE, 2011, pp. 263–270.

[38] C. Dubout, F. Fleuret, Exact acceleration of linear object detectors, in: European Conference on Computer Vision, Springer, 2012, pp. 301–311.

[39] R. E. Kalman, A new approach to linear filtering and prediction problems, Journal of Fluids Engineering 82 (1) (1960) 35–45.

[40] G. R. Bradski, Real time face and object tracking as a component of a perceptual user interface, in: Workshop on Applications of Computer Vision, IEEE, 1998, pp. 214–219.

[41] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, R. Pflugfelder, A. Gupta, A. Bibi, A. Lukezic, A. Garcia-Martin, A. Saffari, A. Petrosino, A. S. Montero, The visual object tracking vot2015 challenge results, in: 2015 IEEE International Conference on Computer Vision Workshop (ICCVW), 2015, pp. 564–586. doi:10.1109/ICCVW.2015.79.

[42] K. Alahari, A. Berg, G. Hager, J. Ahlberg, M. Kristan, J. Matas, A. Leonardis, L. Cehovin, G. Fernandez, T. Vojir, G. Nebehay, R. Pflugfelder, A. Lukezic, A. Garcia-Martin, A. Saffari, A. Li, A. S. Montero, B. Zhao, C. Schmid, D. Chen, D. Du, F. S. Khan, F. Porikli, G. Zhu, G. Zhu, H. Lu, H. Kieritz, H. Li, H. Qi, J. chan Jeong, J. il Cho, J.-Y. Lee, J. Zhu, J. Li, J. Feng, J. Wang, J.-W. Kim, J. Lang, J. M. Martinez, K. Xue, L. Ma, L. Ke, L. Wen, L. Bertinetto, M. Danelljan, M. Arens, M. Tang, M.-C. Chang, O. Miksik, P. H. S. Torr, R. Martin-Nieto, R. Laganière, S. Hare, S. Lyu, S.-C. Zhu, S. Becker, S. L. Hicks, S. Golodetz, S. Choi, T. Wu, W. Hübner, X. Zhao, Y. Hua, Y. Li, Y. Lu, Y. Li, Z. Yuan, Z. Hong, The thermal infrared visual object tracking vot-tir2015 challenge results, in: 2015 IEEE International Conference on Computer Vision Workshop (ICCVW), 2015, pp. 639–651. doi:10.1109/ICCVW.2015.86.

[43] T. Vojir, J. Noskova, J. Matas, Robust scale-adaptive mean-shift for tracking, Pattern Recognition Letters 49 (2014) 250 – 258.

[44] M. Danelljan, G. Häger, F. Khan, M. Felsberg, Accurate scale estimation for robust visual tracking, in: British Machine Vision Conference, BMVA Press, 2014.

[45] C. Bailer, D. Stricker, Tracker fusion on vot challenge: How does it perform and what can we learn about single trackers?, in: 2015 IEEE International Conference on Computer Vision Workshop (ICCVW), 2015, pp. 630–638. doi:10.1109/ICCVW.2015.85.

[46] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? the kitti vision benchmark suite, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2012, pp. 3354–3361.

[47] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part-based models, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (9) (2010) 1627–1645.

[48] M. Danelljan, A. Robinson, F. S. Khan, M. Felsberg, Beyond correlation filters: Learning continuous convolution operators for visual tracking, in: European Conference on Computer Vision, Springer, 2016, pp. 472–488.

[49] D. S. Bolme, J. R. Beveridge, B. Draper, Y. M. Lui, et al., Visual object tracking using adaptive correlation filters, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 2544–2550.

*Author Biography

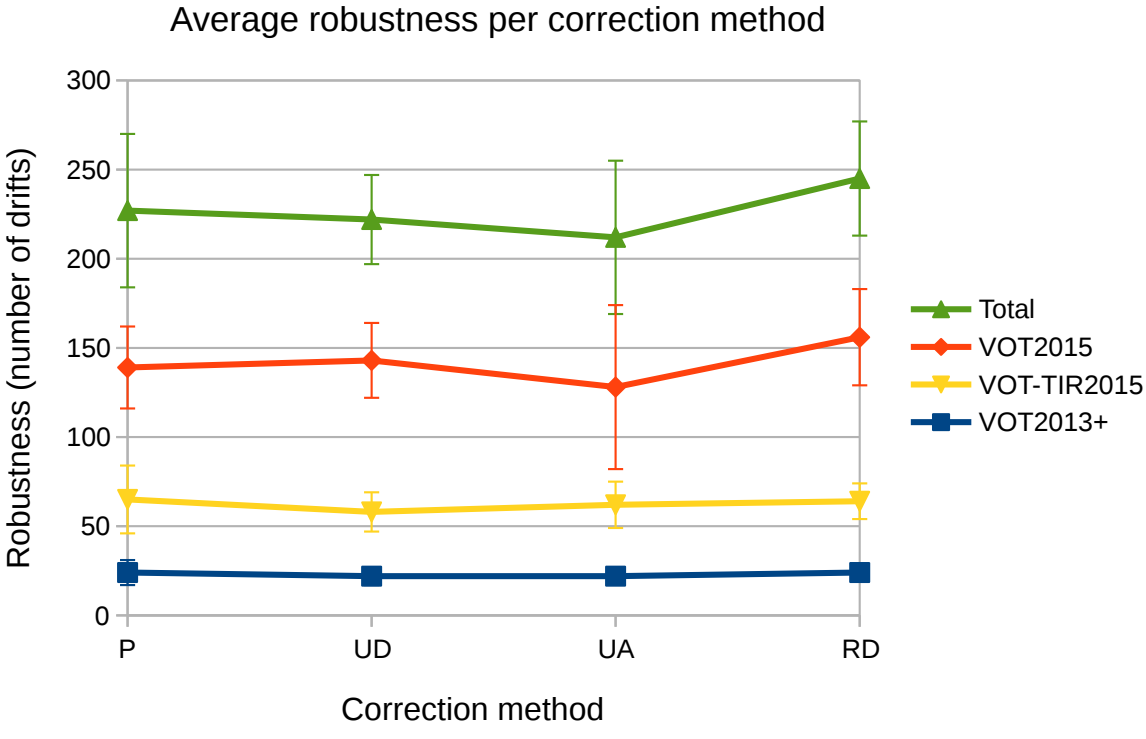
Isabelle Leang graduated from the Ecole Nationale Supérieure de l'Electronique et de ses Applications (France), received a Master degree in Computer Sciences from the Université de Cergy-Pontoise (France) in 2012. She is actually preparing a Ph.D. degree in the Information Processing and Modeling Departement at ONERA (France).

Stéphane Herbin received an engineering degree from the Ecole Supérieure d'Electricité (Supélec), the M.Sc. degree in Electrical Engineering from the University of Illinois at Urbana-Champaign, and the Ph.D. degree in applied mathematics from the Ecole Normale Supérieure de Cachan. Employed by ONERA since 2000, he works in the Information Processing and Modeling Department. His main research interests are stochastic modeling and analysis for object recognition and scene interpretation in images and videos.

Benoît Girard received a Ph.D. degree in Computer Science (2003) from the Université Pierre et Marie Curie (Paris, France). He currently work as a Research Director at the Centre National de la Recherche Scientifique. His main research interests are action selection, reinforcement learning and decision making in animals and robots.

Jacques Droulez received a mathematical training at Ecole Polytechnique (Paris, France) and a MD (1982) from the University Paris 6. He is currently Research Director at the Centre National de la Recherche Scientifique. His main research interests are motion and object perception, sensori-motor control and Bayesian modeling of biological systems.

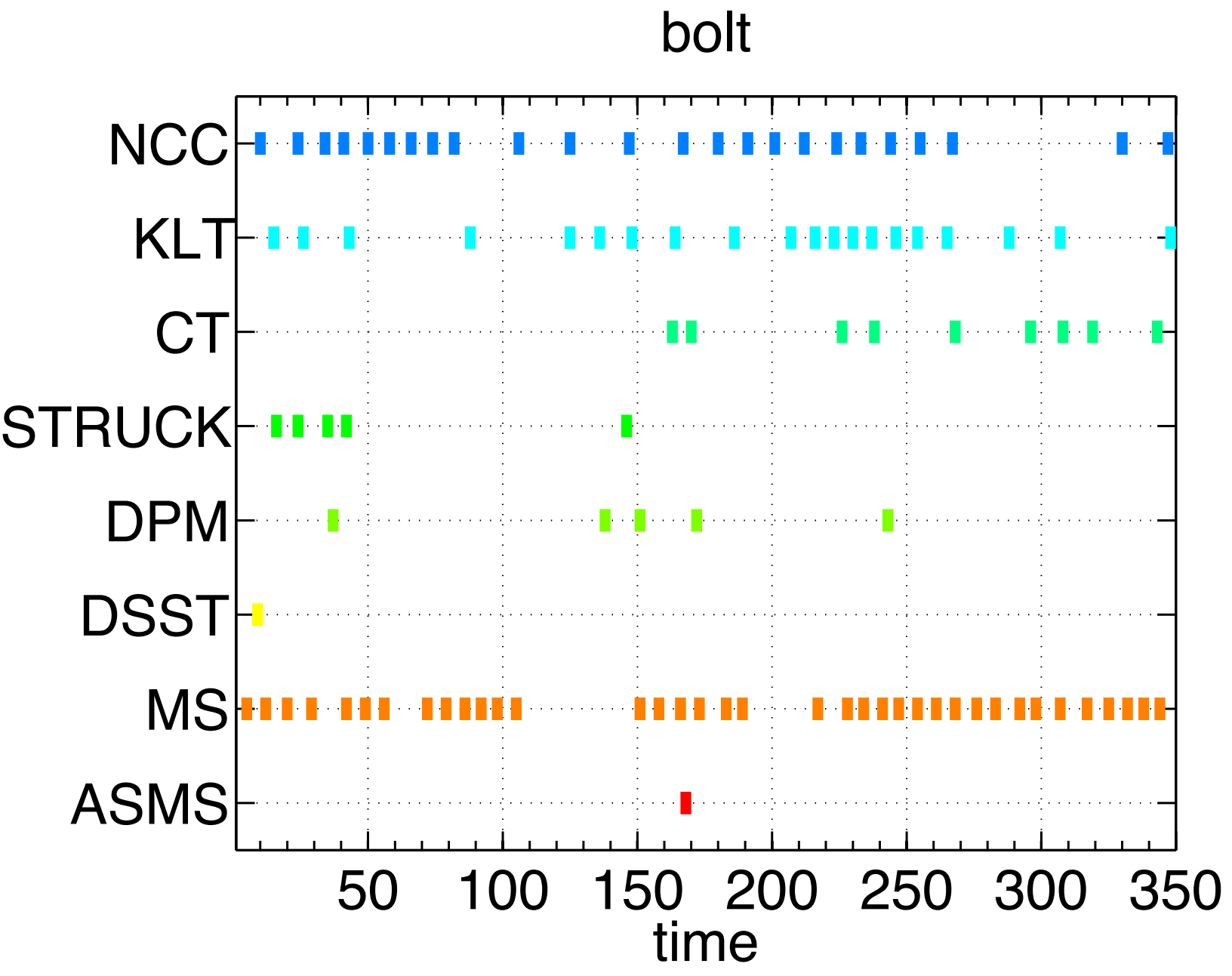
correctionstep.pdf



fish1_img_0110.eps

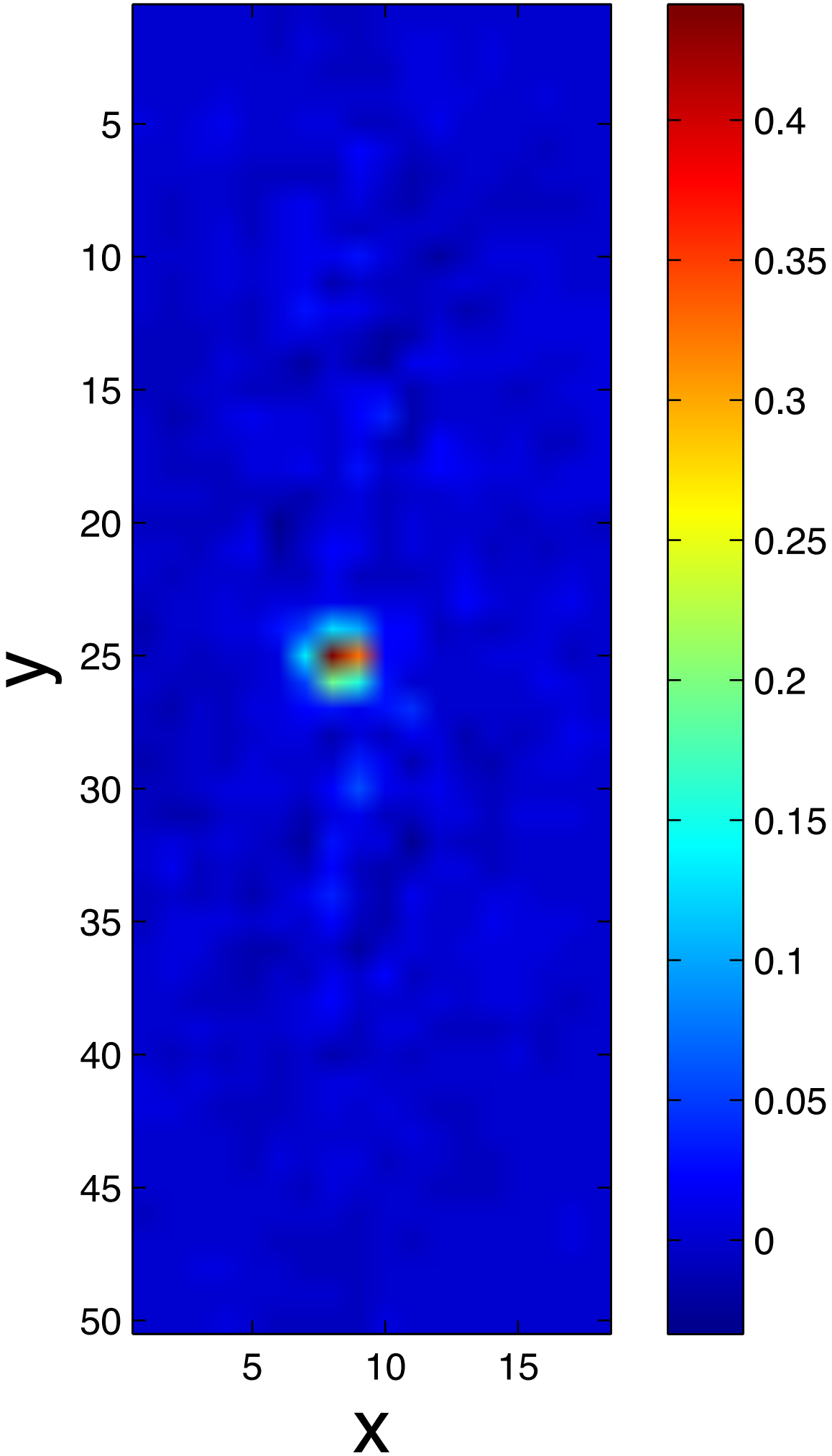
quadrocopter_img_0160.eps

tracklets_vot2013_bolt.eps

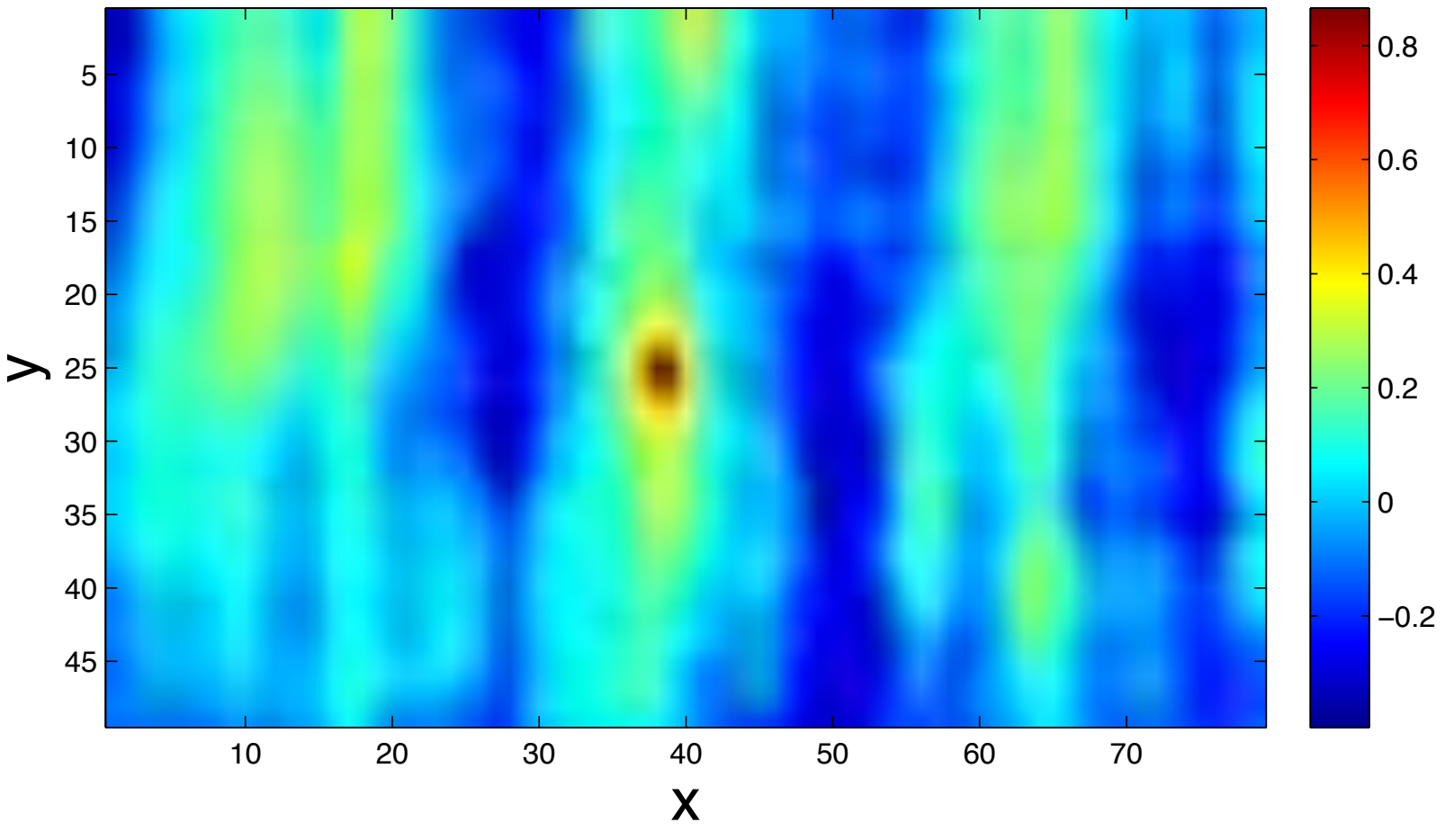


fish1_img_0150.eps

map_dsst_0002.eps

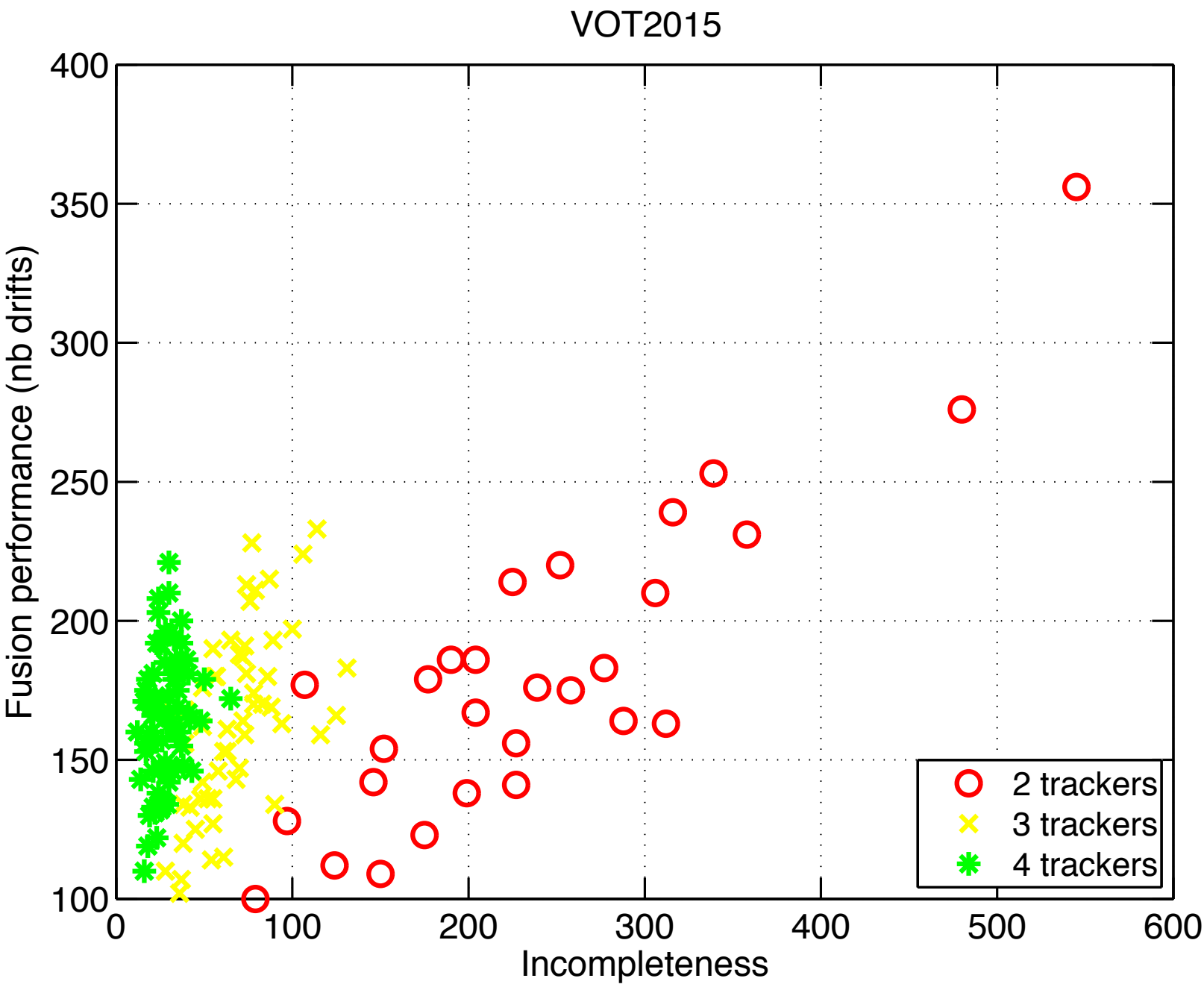


map_ncc_0002.eps

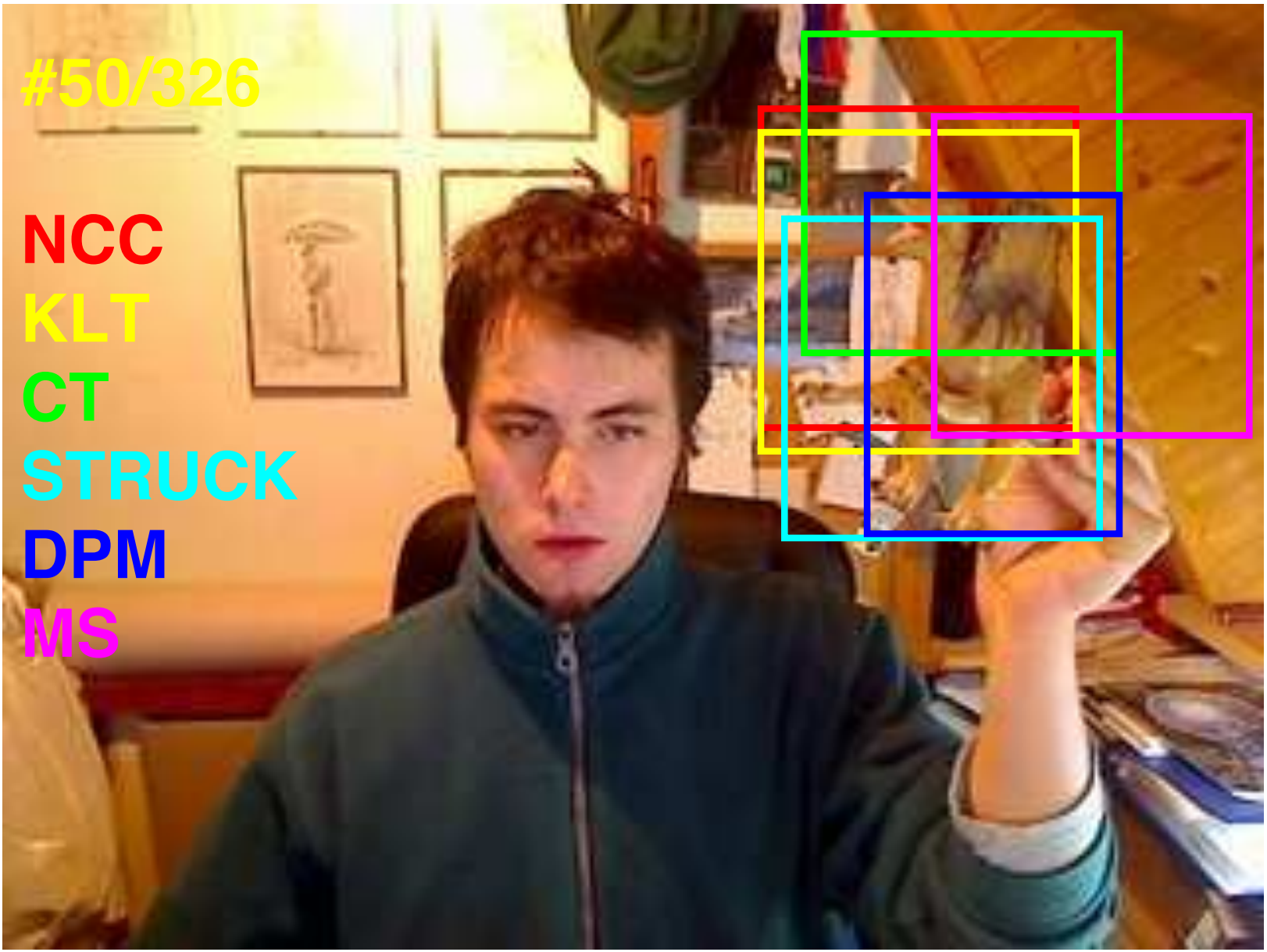


fish1_img_0145.eps

vot2015_fusion99.eps

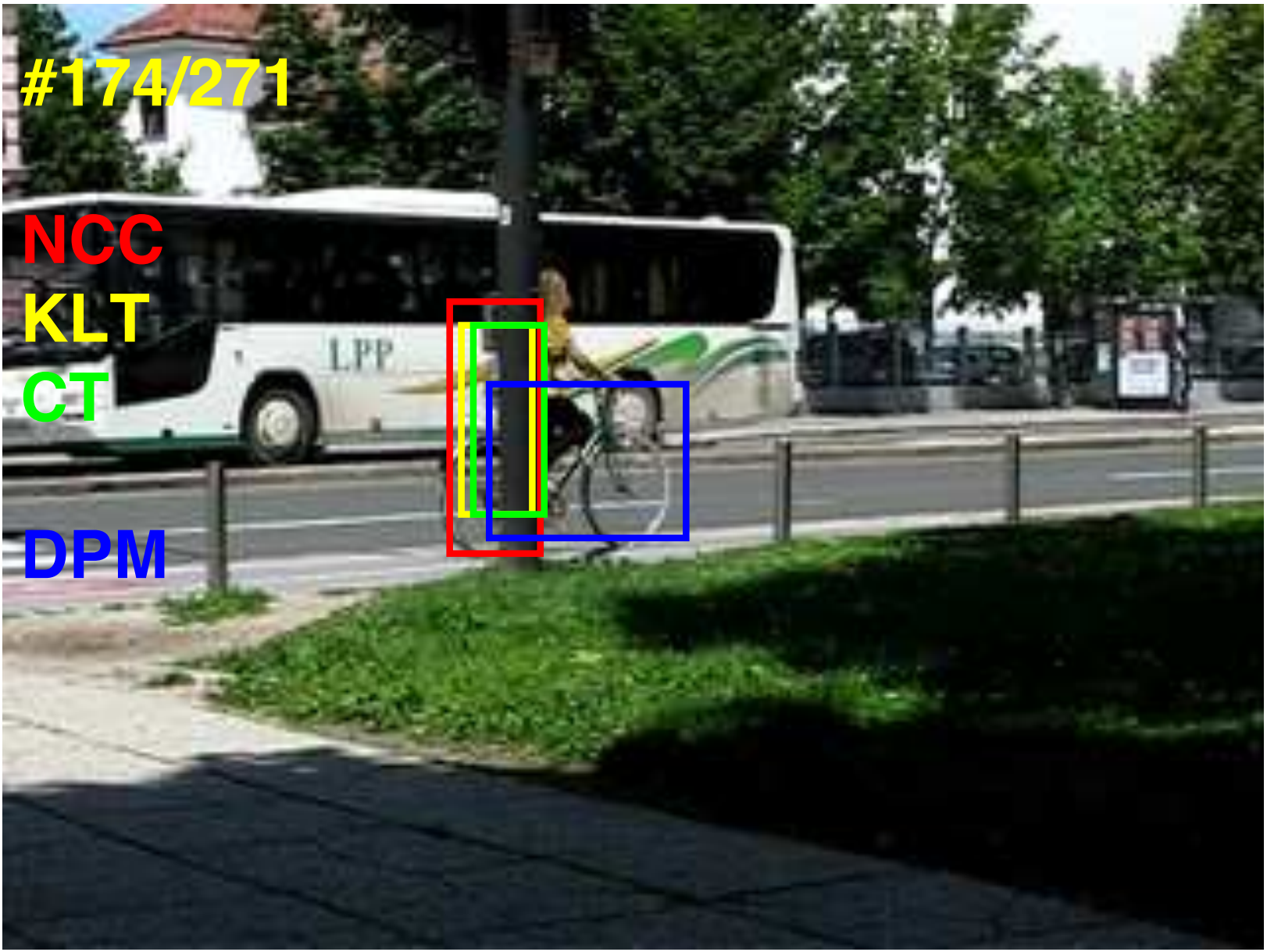


dinosaur_img_0050.eps

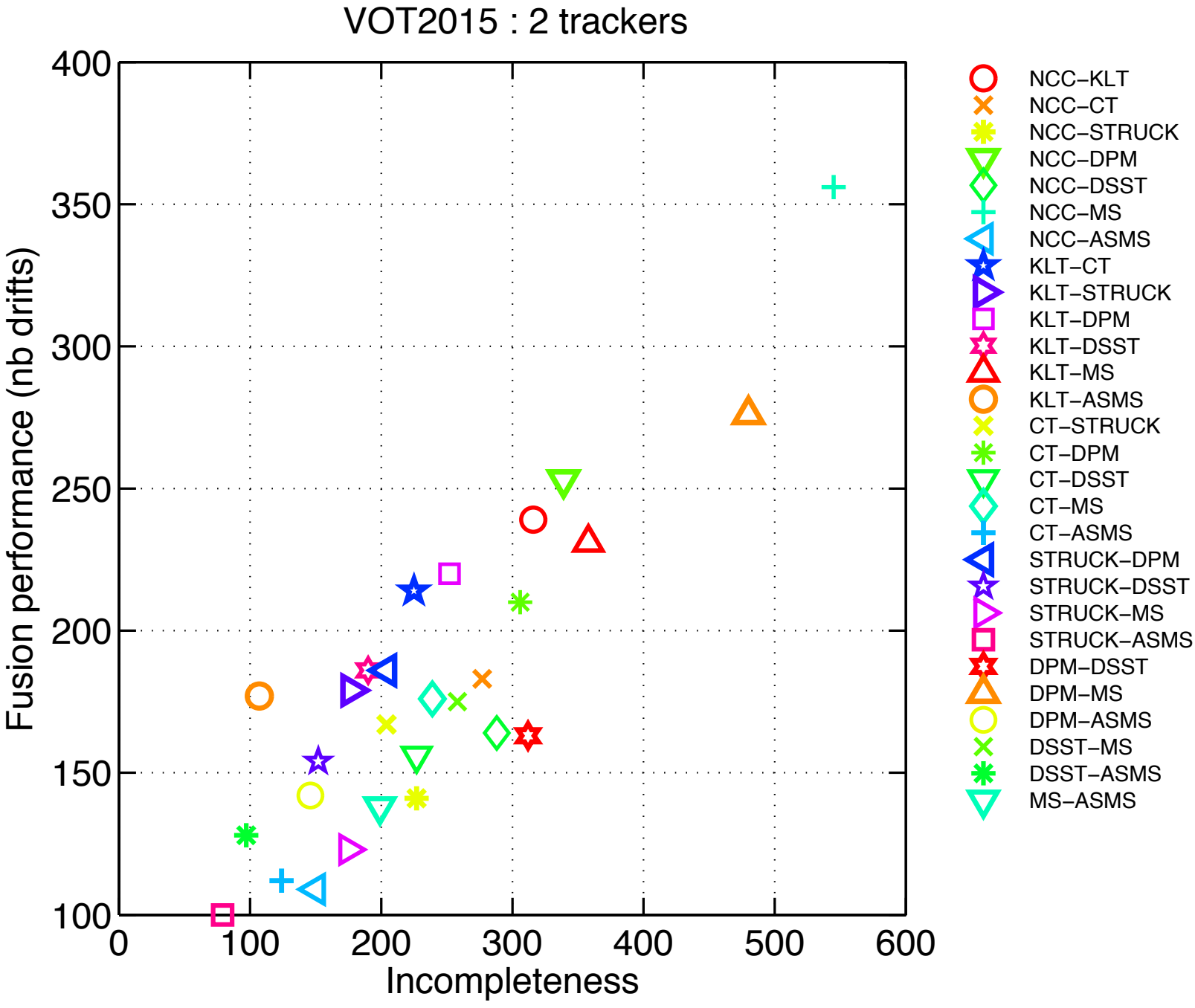


quadrocopter_img_0173.eps

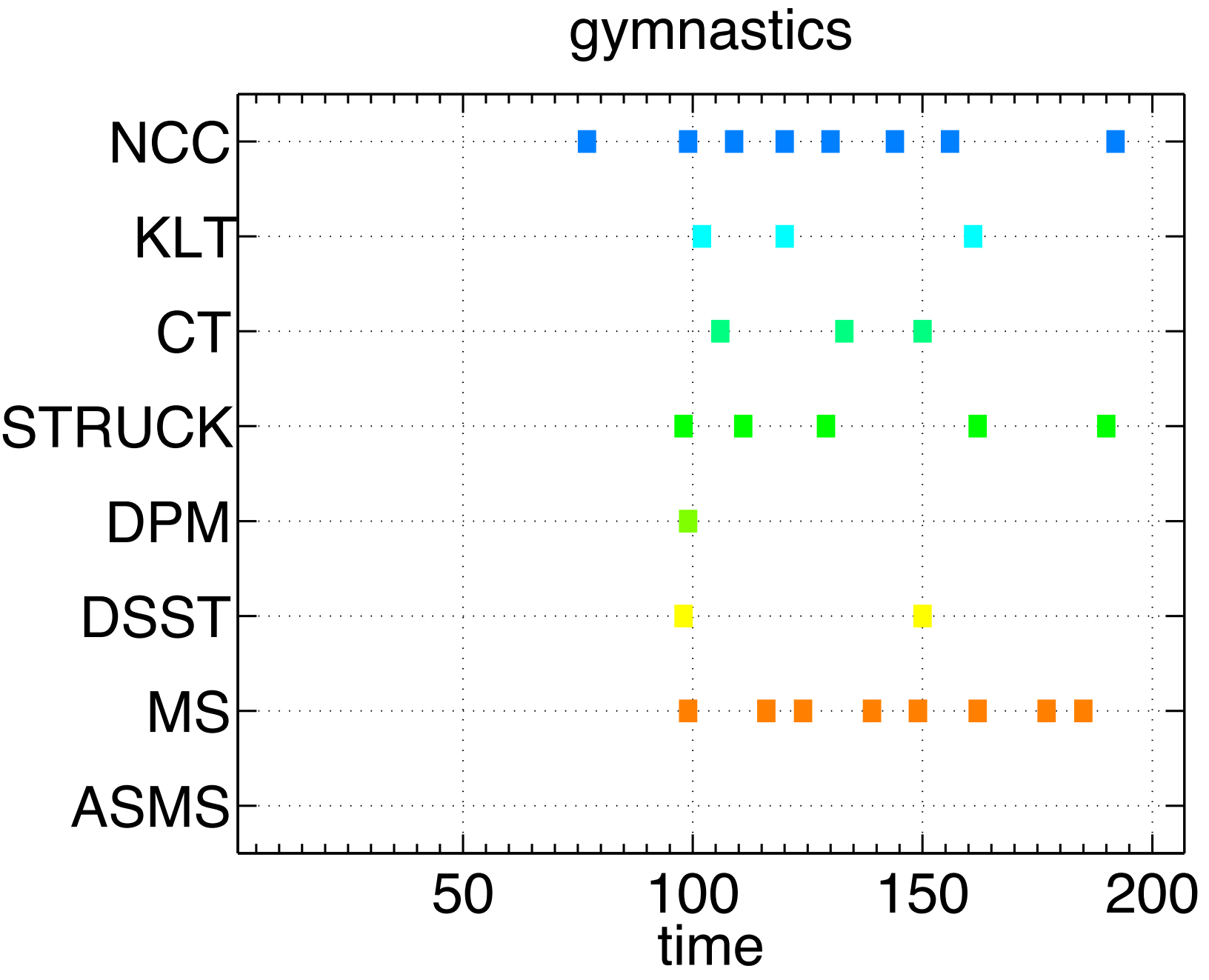
bicycle_img_0174.eps



vot2015_2trackers.eps

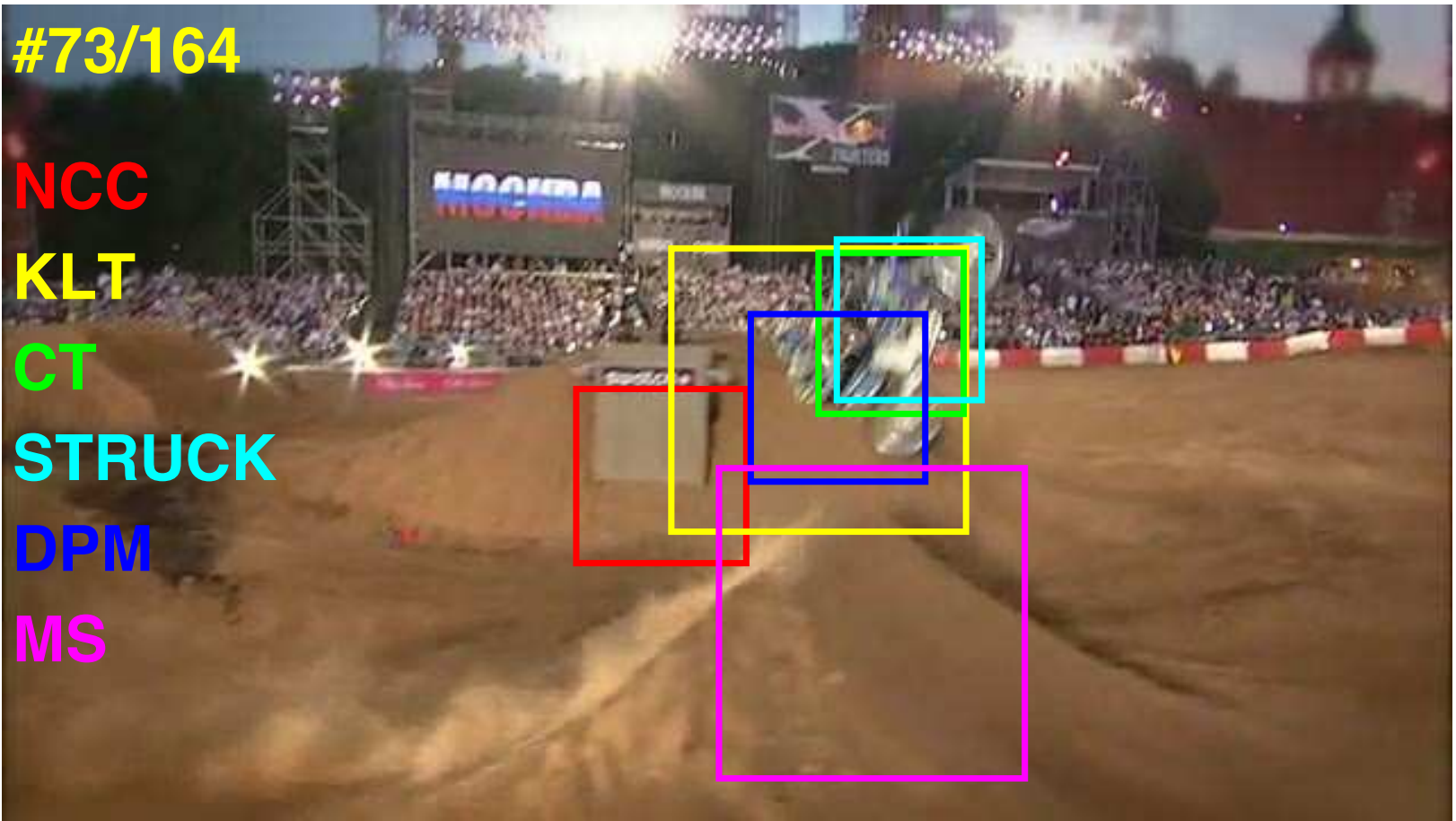


tracklets_vot2013_gymnastics.eps



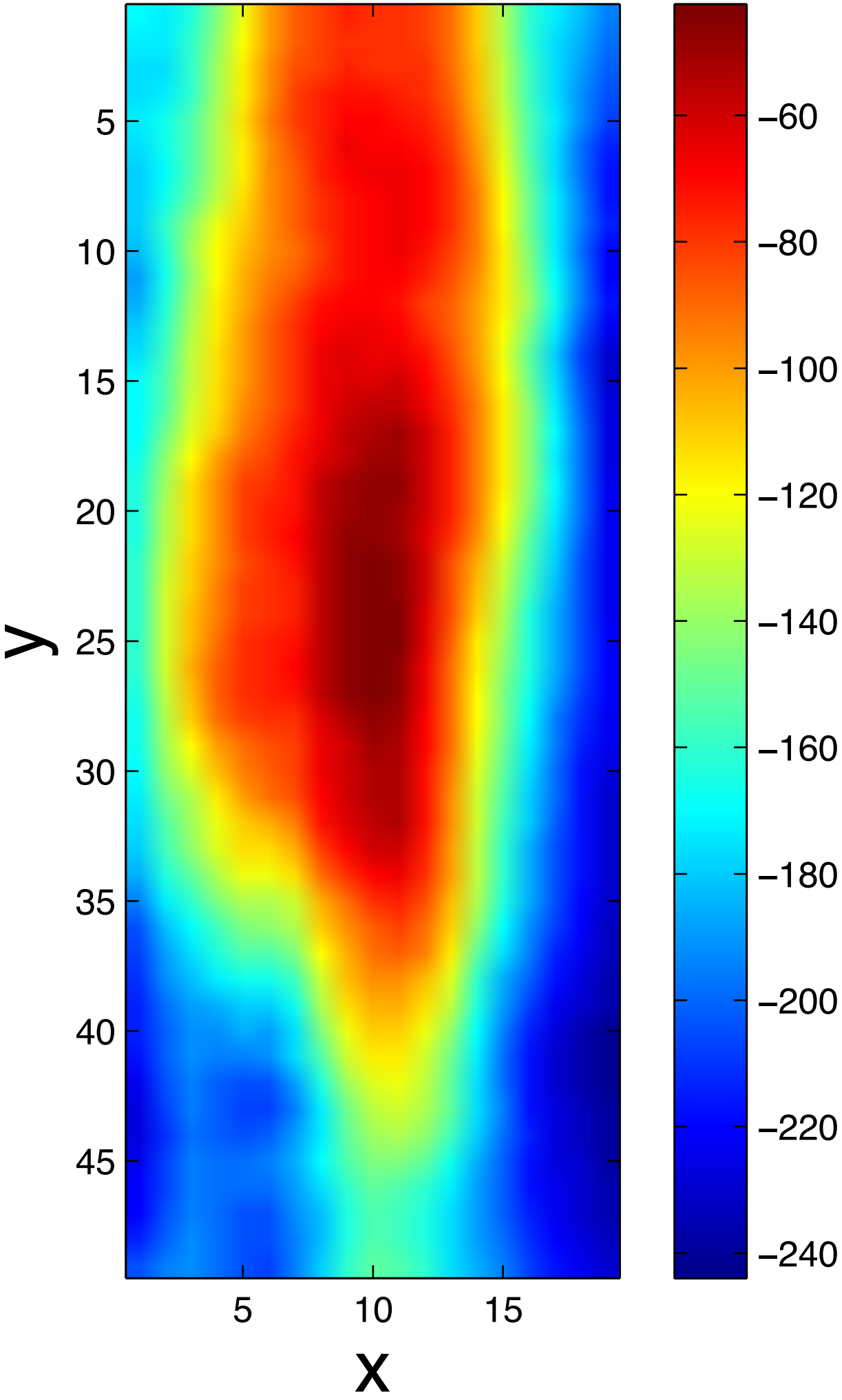
motocross1_img_0055.eps

motocross1_img_0073.eps

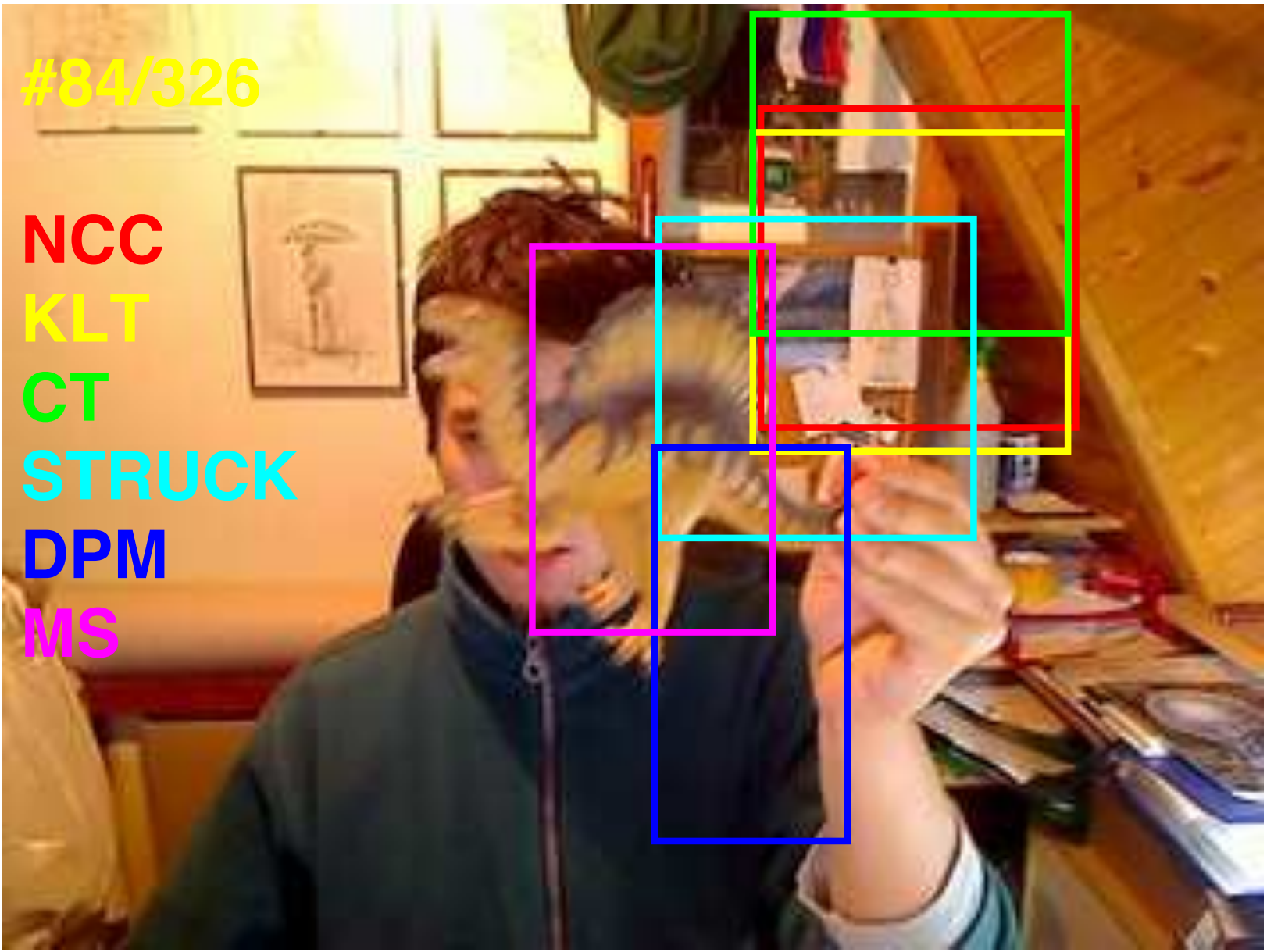


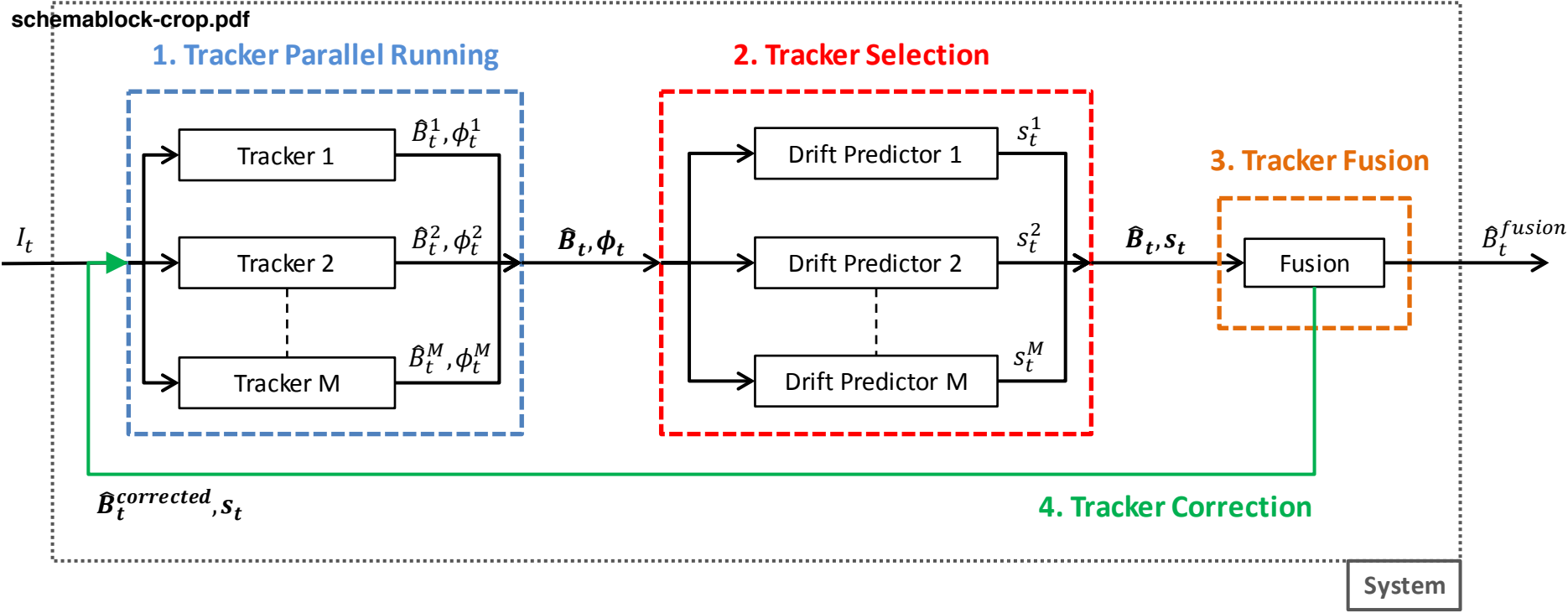
motocross1_img_0079.eps

map_ct_0002.eps



dinosaur_img_0084.eps

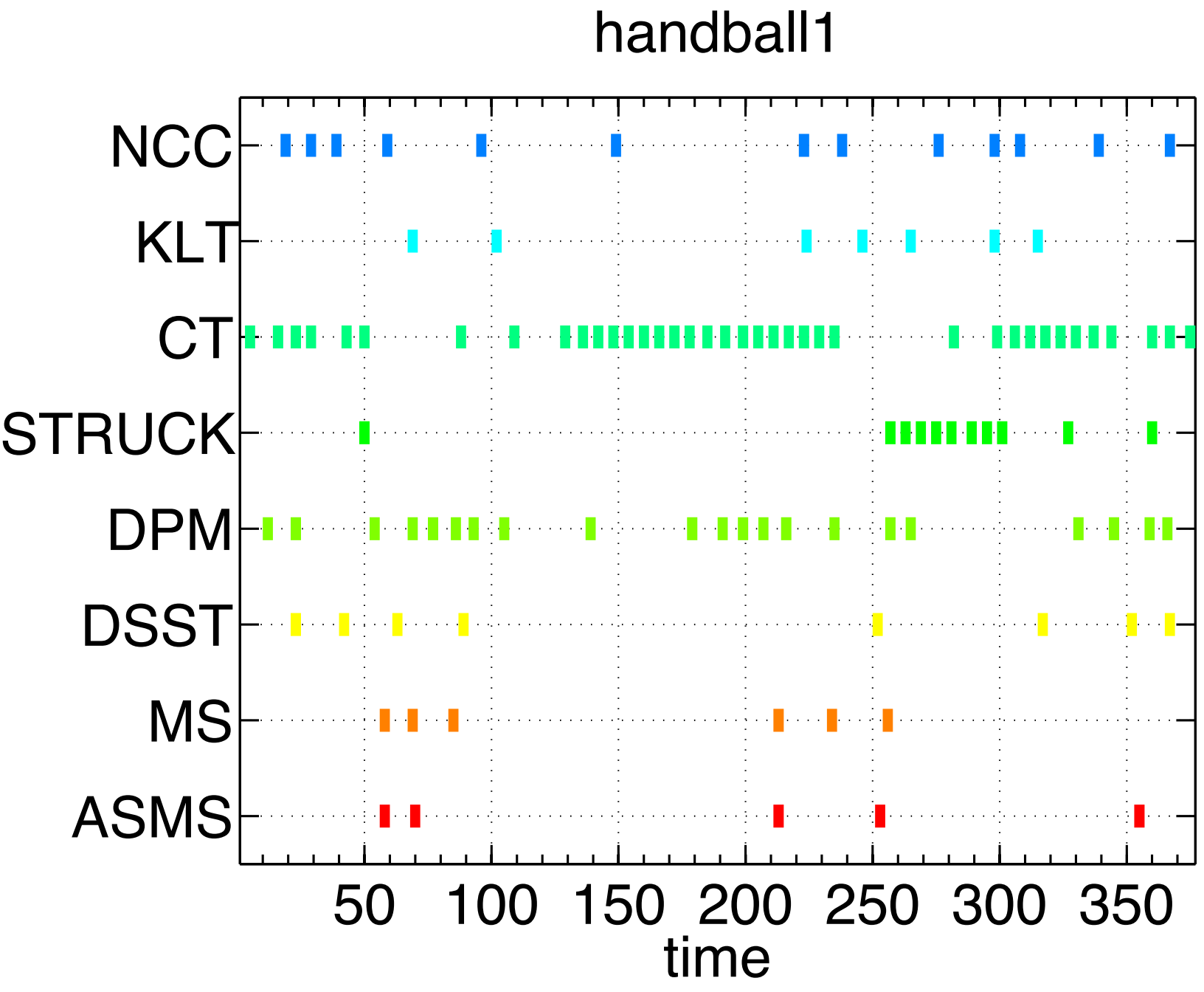



schemablock-crop.pdf
1. Tracker Parallel Running
2. Tracker Selection
3. Tracker Fusion
4. Tracker Correction
Tracker 1
Tracker 2
Tracker M
Drift Predictor 1
Drift Predictor 2
Drift Predictor M
Fusion
I_t
$\hat{B}_t^1, \phi_t^1$
$\hat{B}_t^2, \phi_t^2$
$\hat{B}_t^M, \phi_t^M$
$\hat{B}_t, \phi_t$
s_t^1
s_t^2
s_t^M
$\hat{B}_t, s_t$
$\hat{B}_t^{fusion}$
$\hat{B}_t^{corrected}, s_t$
System

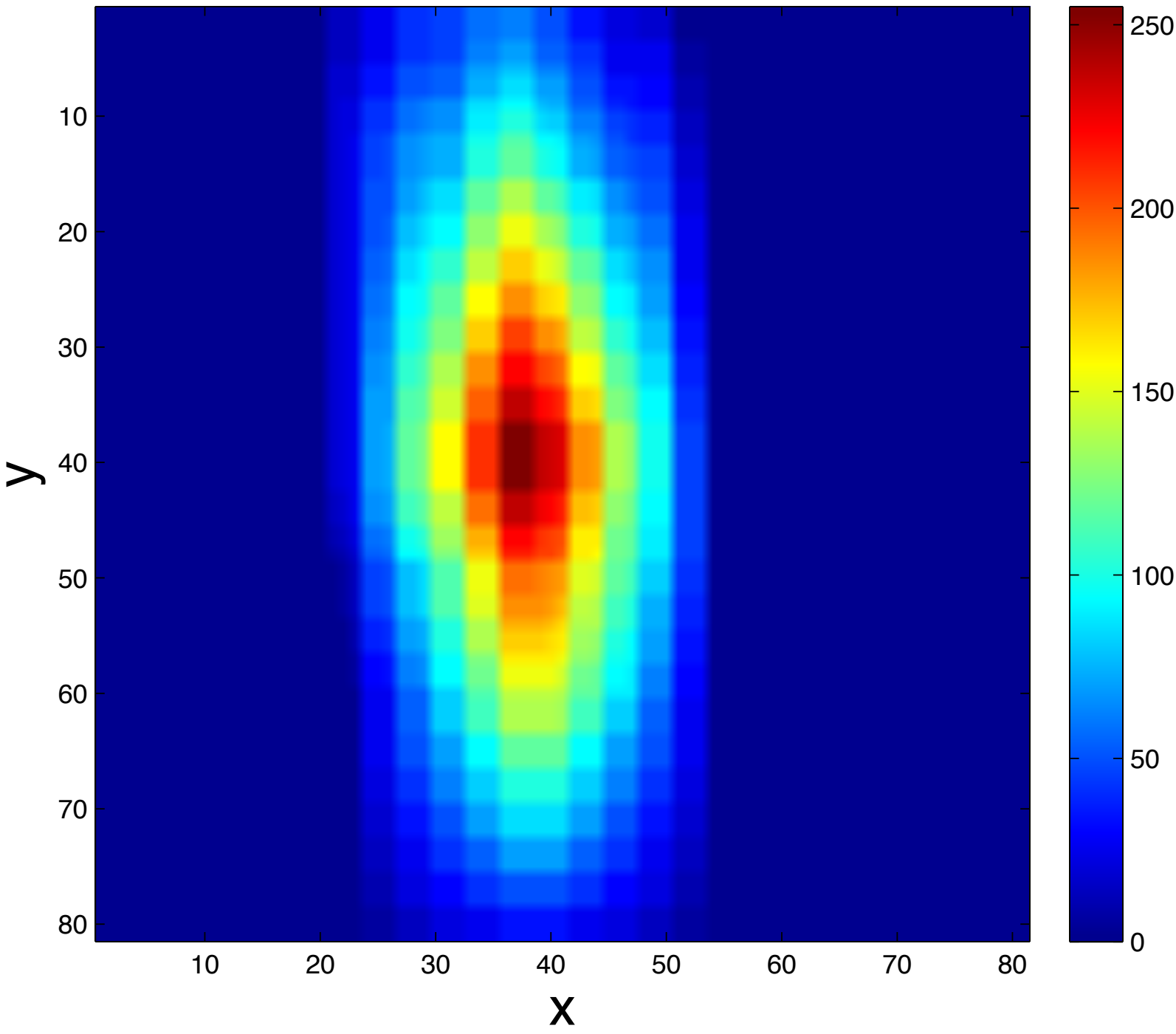
quadrocopter_img_0177.eps

map_img_0002.eps

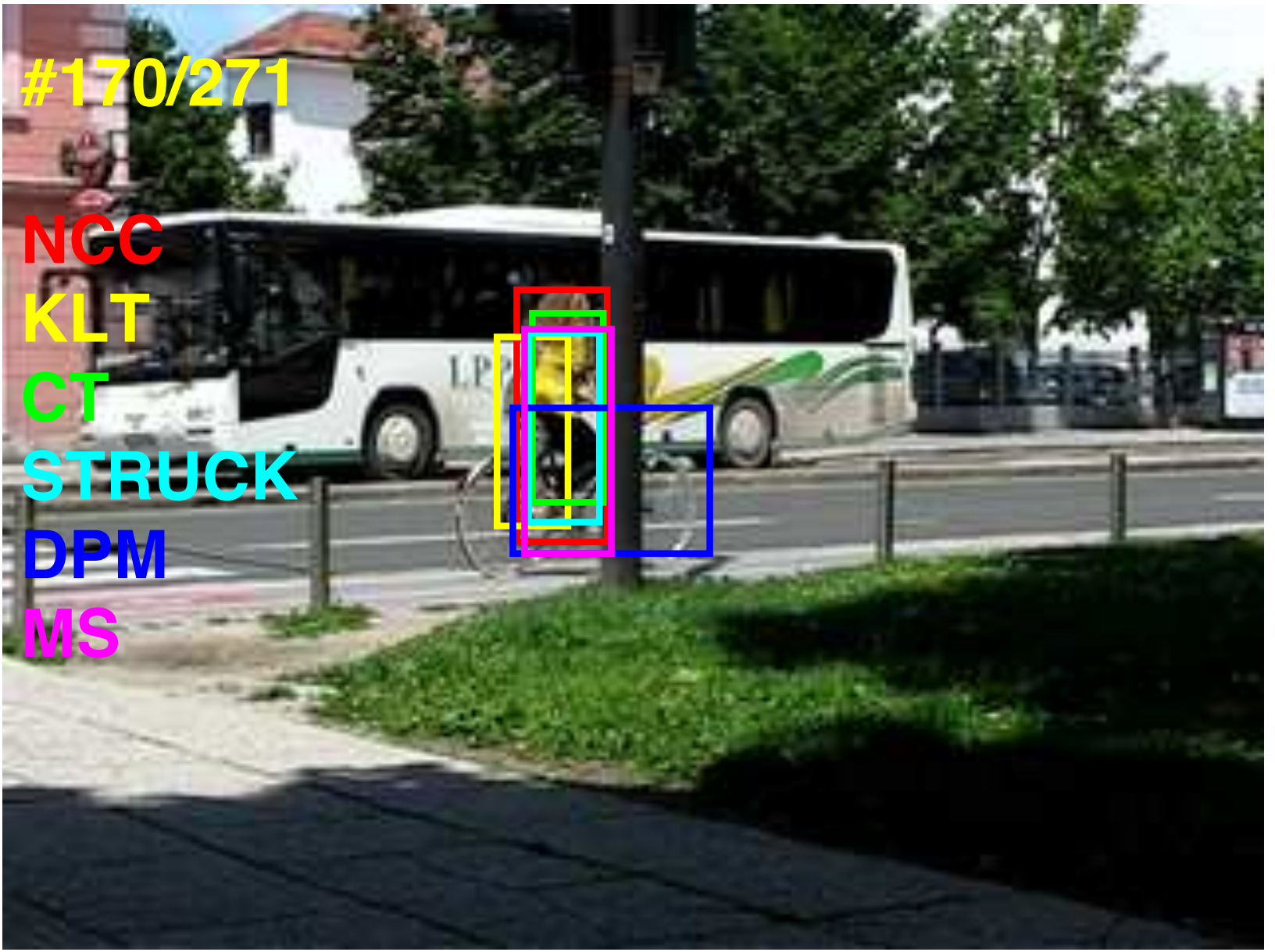
tracklets_vot2015_handball1.eps



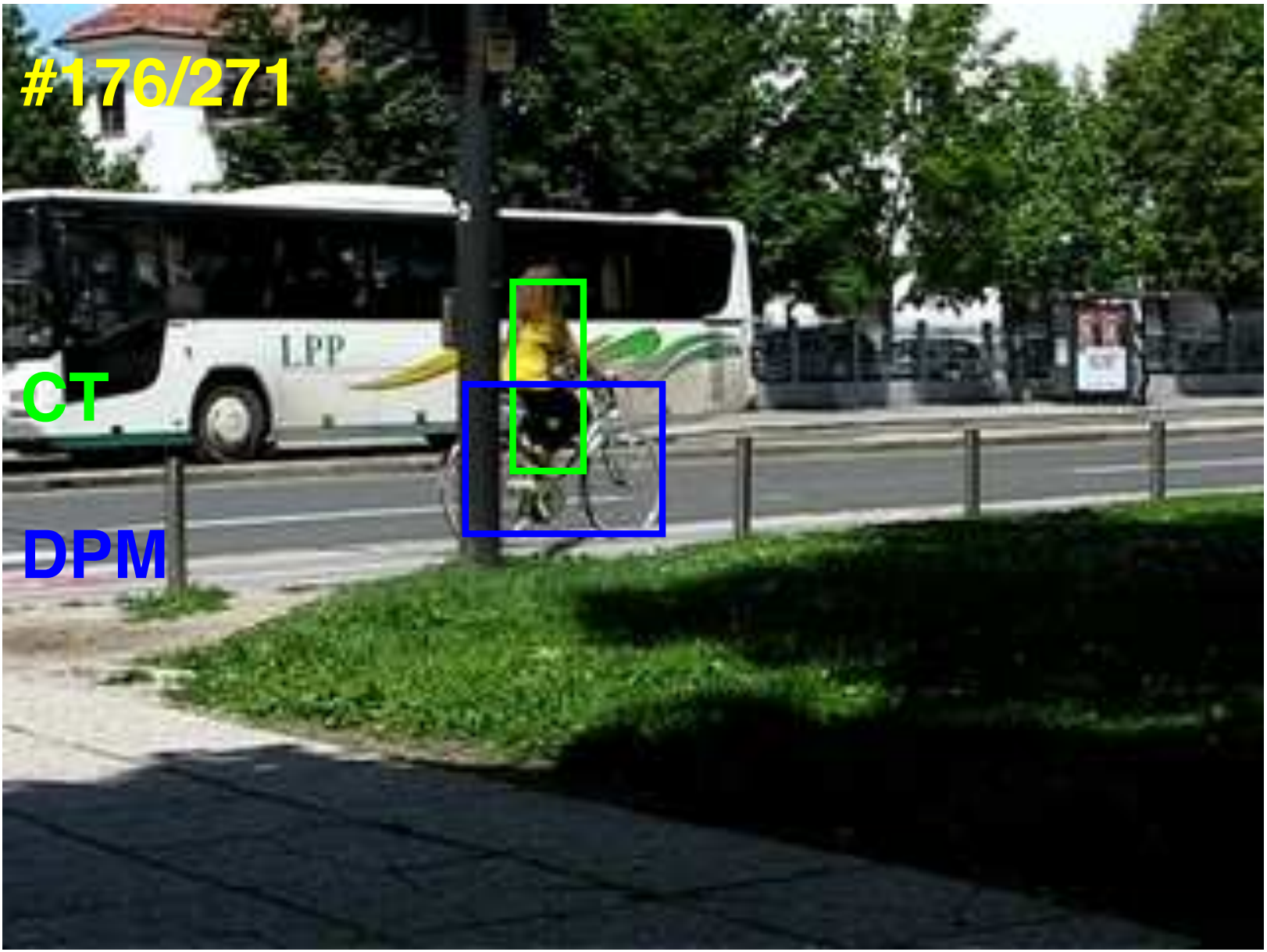
map_klt_0002.eps



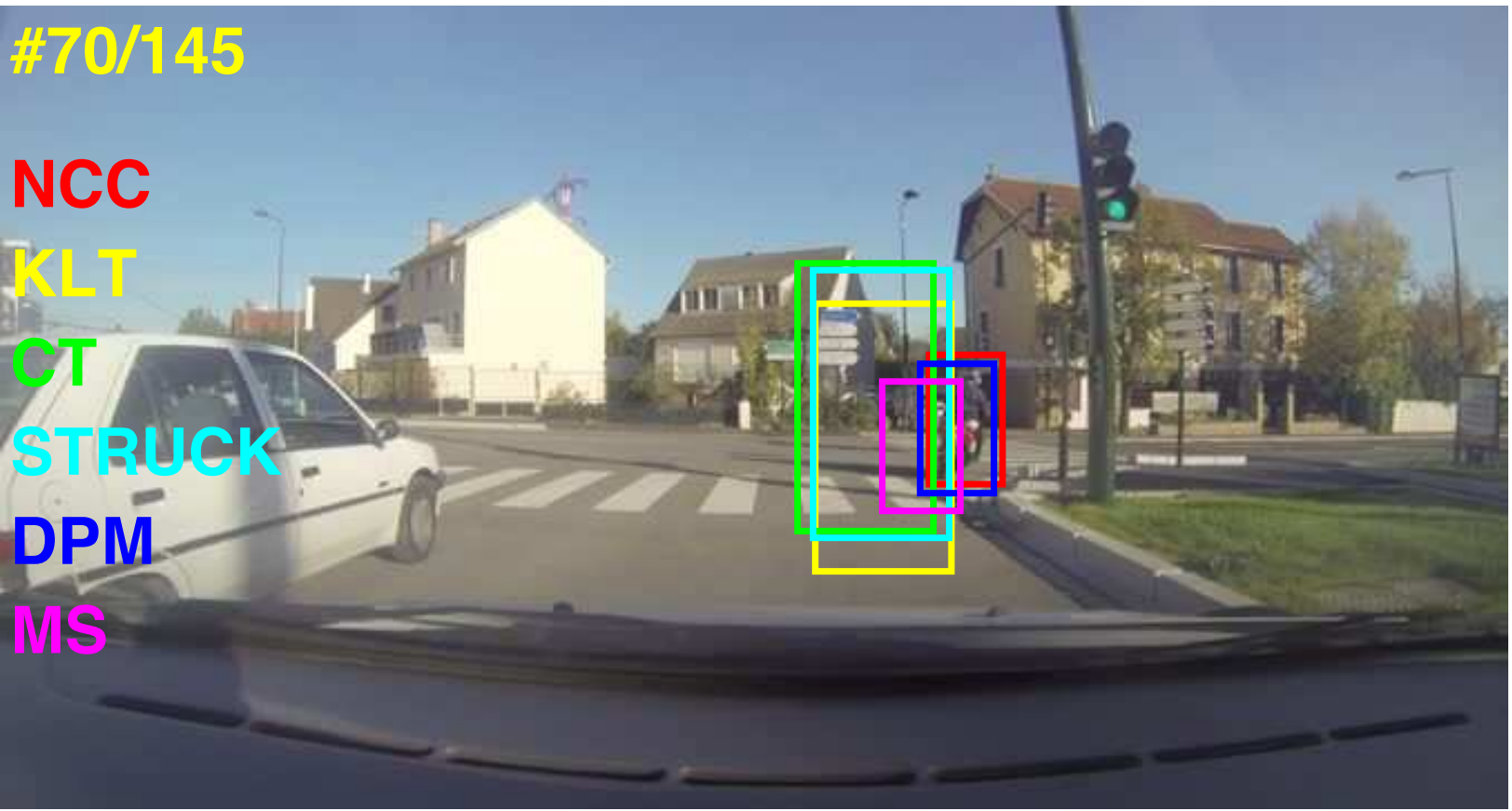
bicycle_img_0170.eps



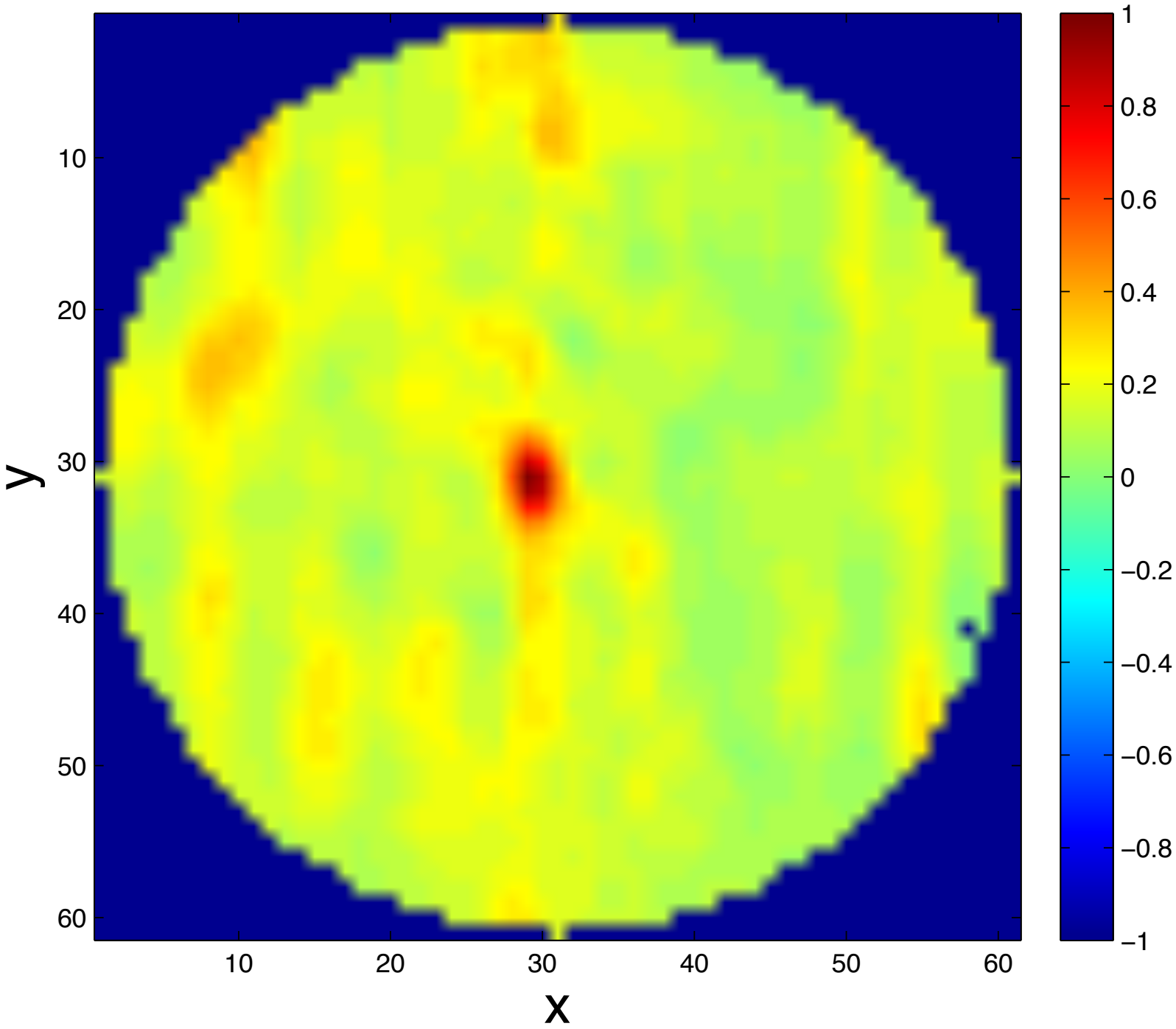
bicycle_img_0176.eps



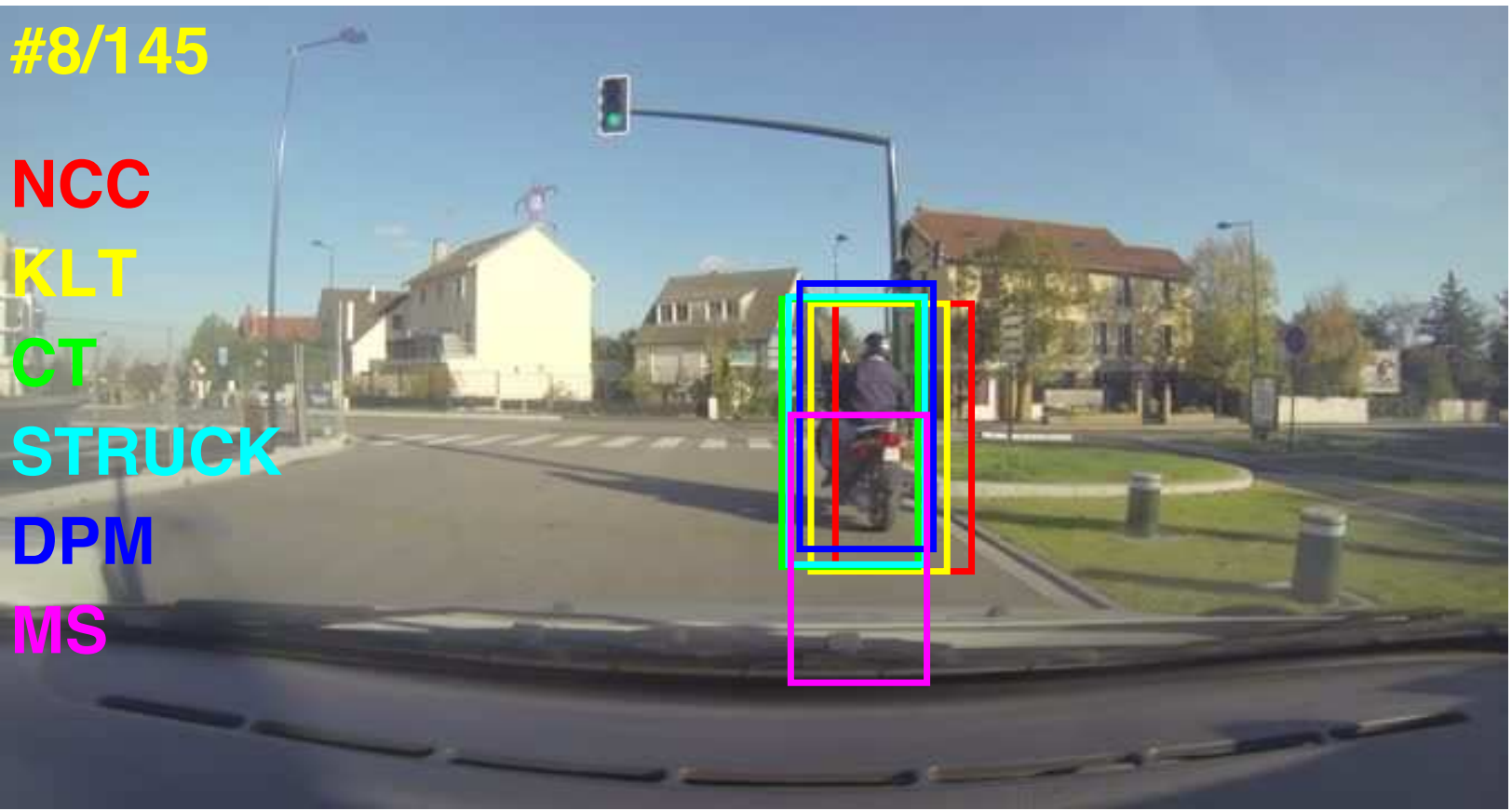
gopr0020_moto_img_0070.eps



map_struck_0002.eps

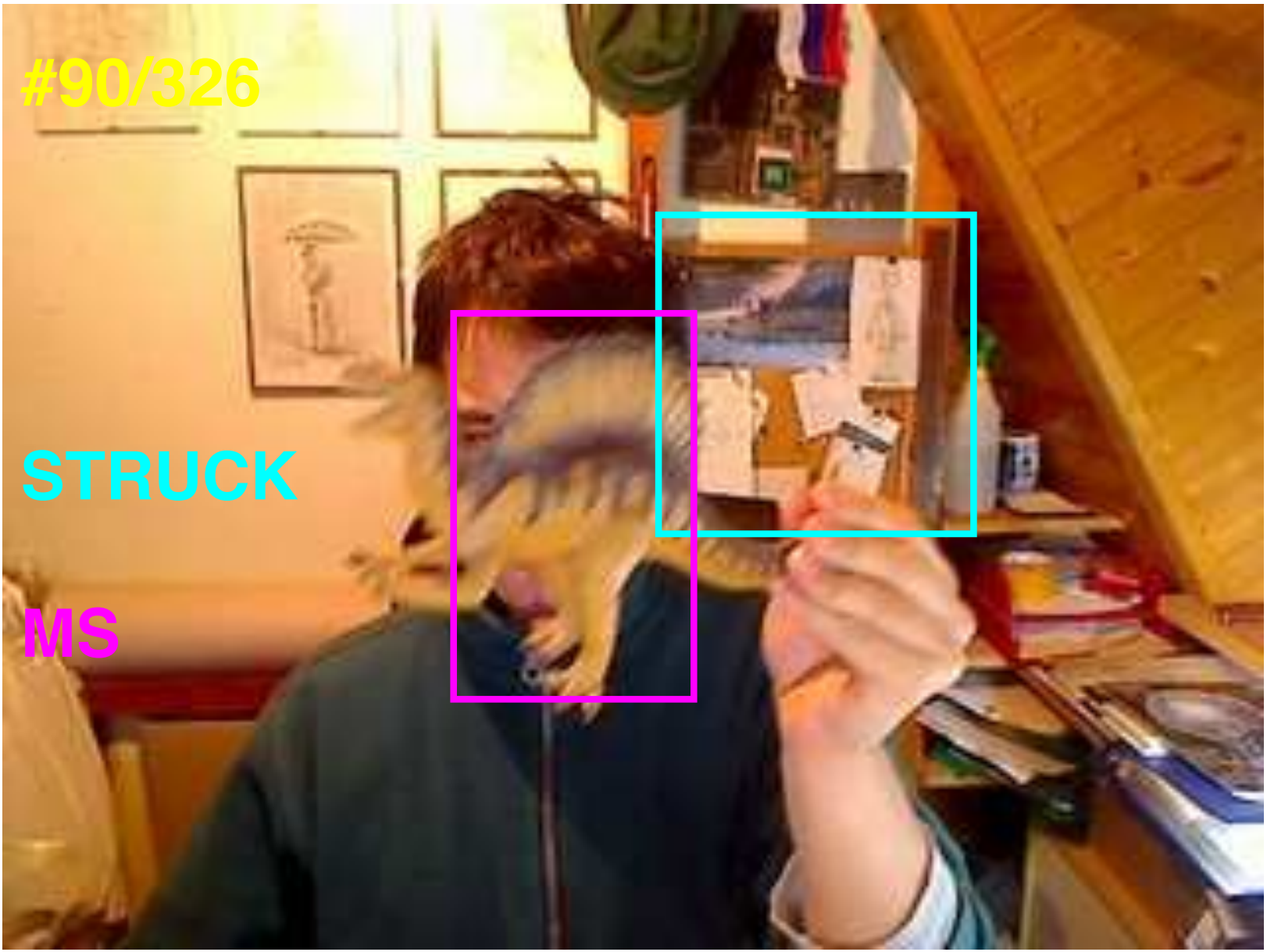


gopr0020_moto_img_0008.eps

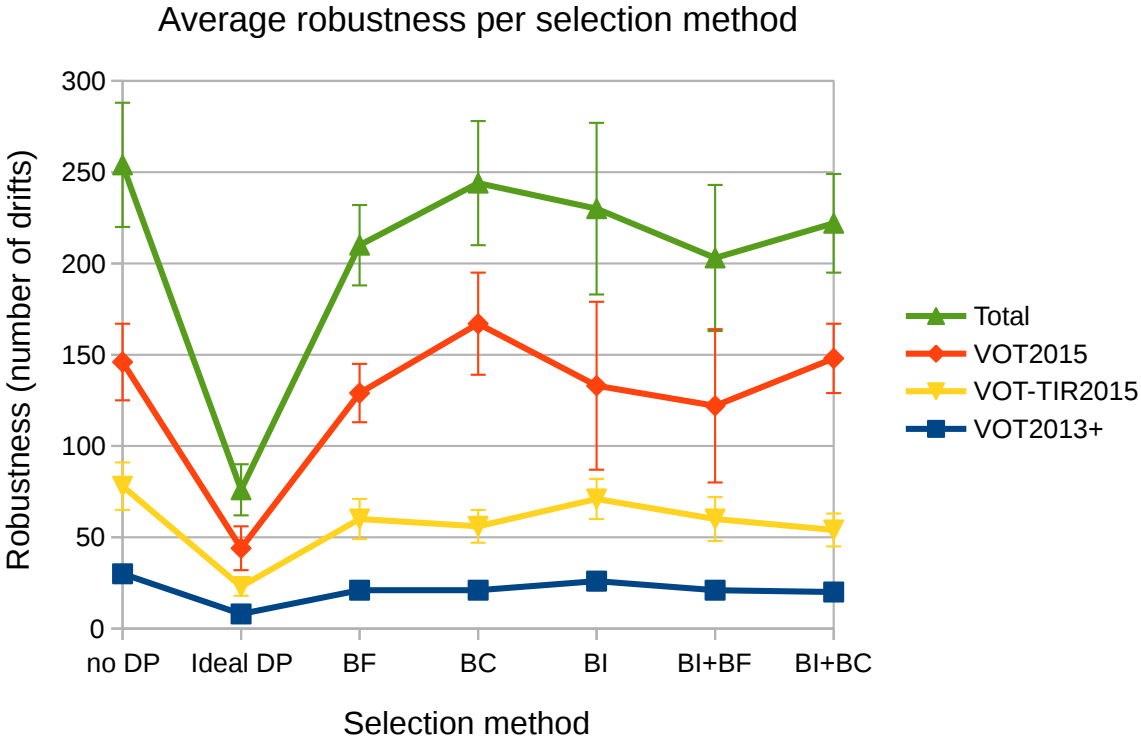


gopr0020_moto_img_0075.eps

dinosaur_img_0090.eps

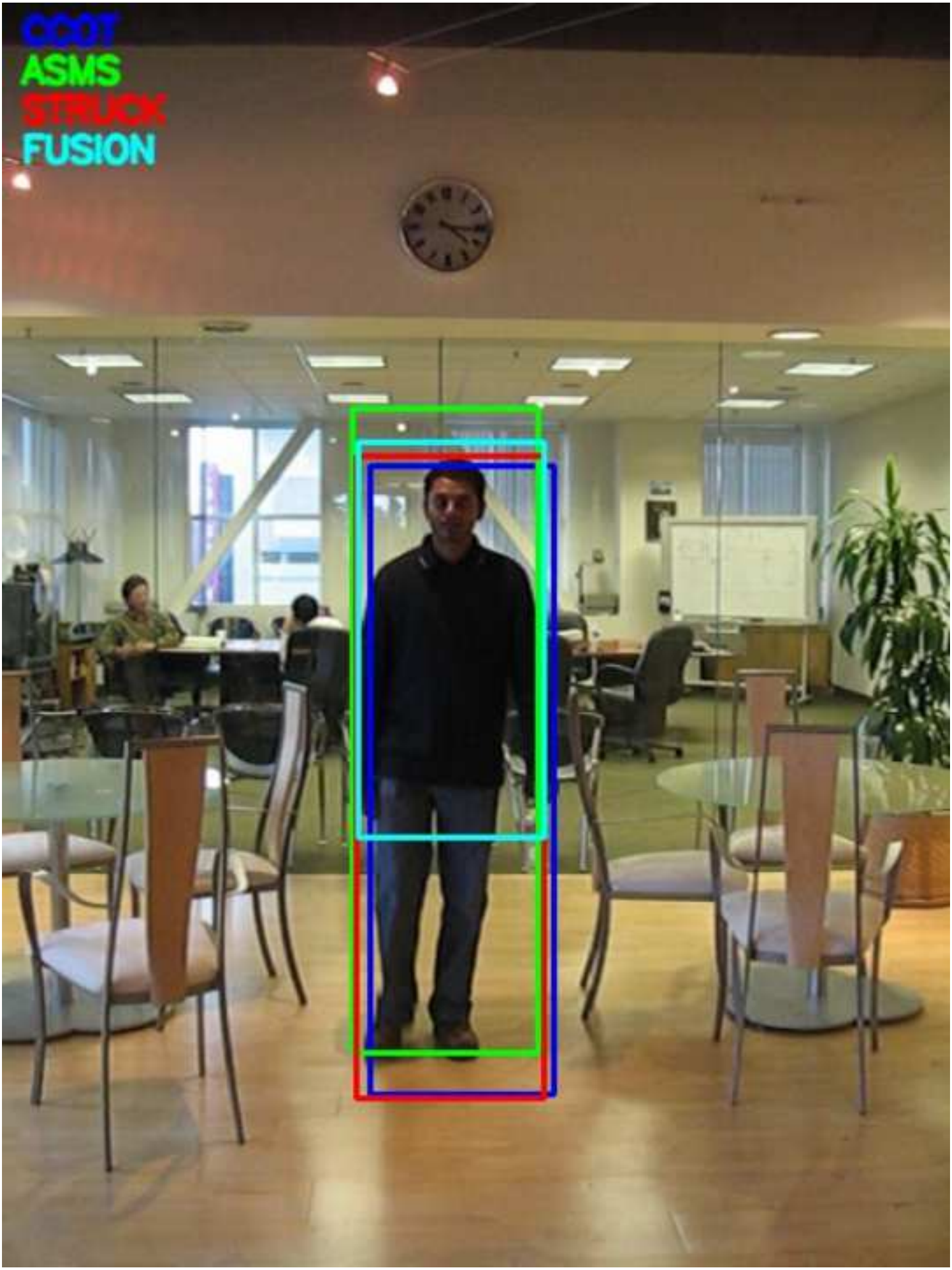


selectionstep.pdf



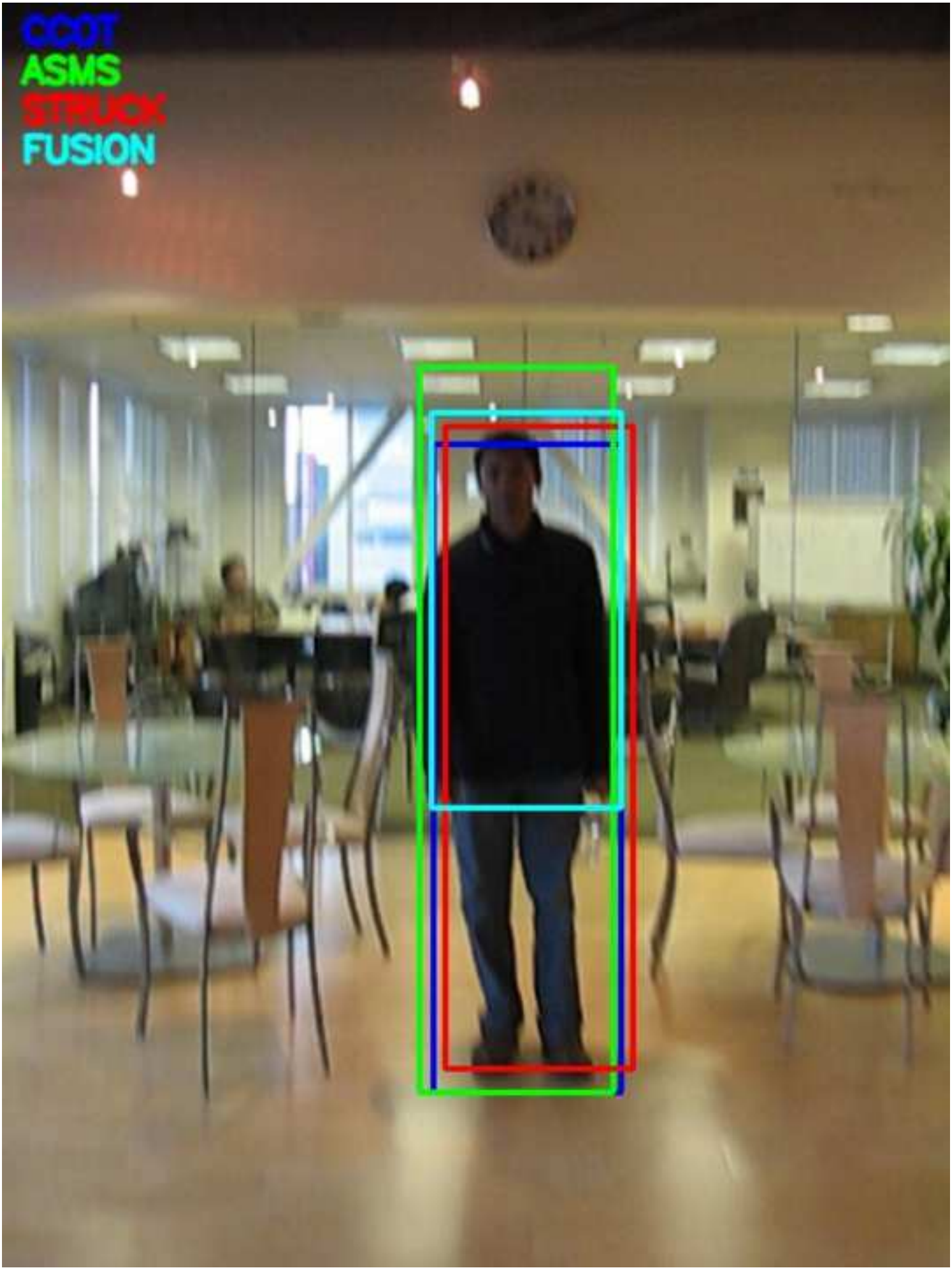
Figure

Click here to download high resolution image



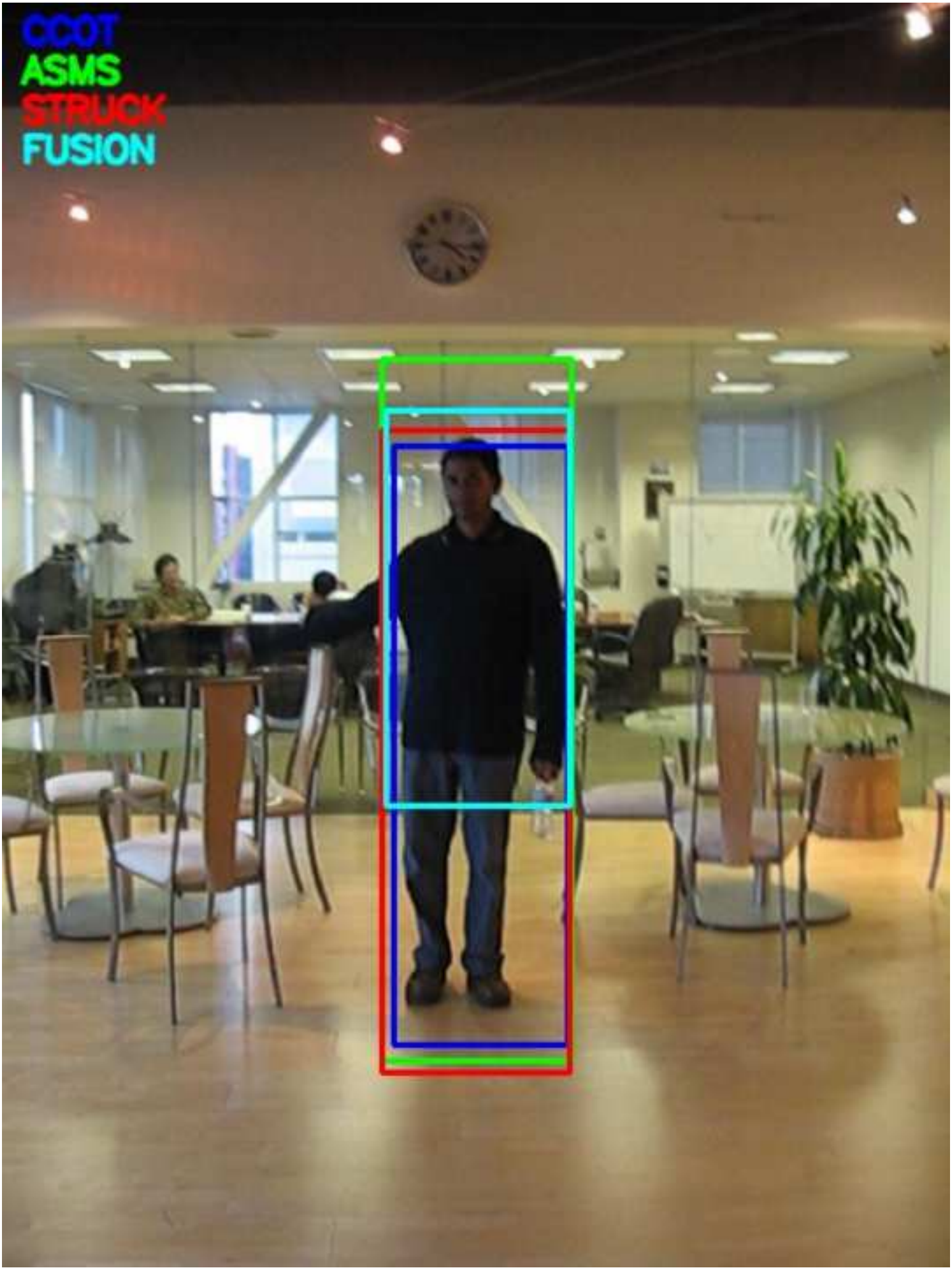
Figure

Click here to download high resolution image



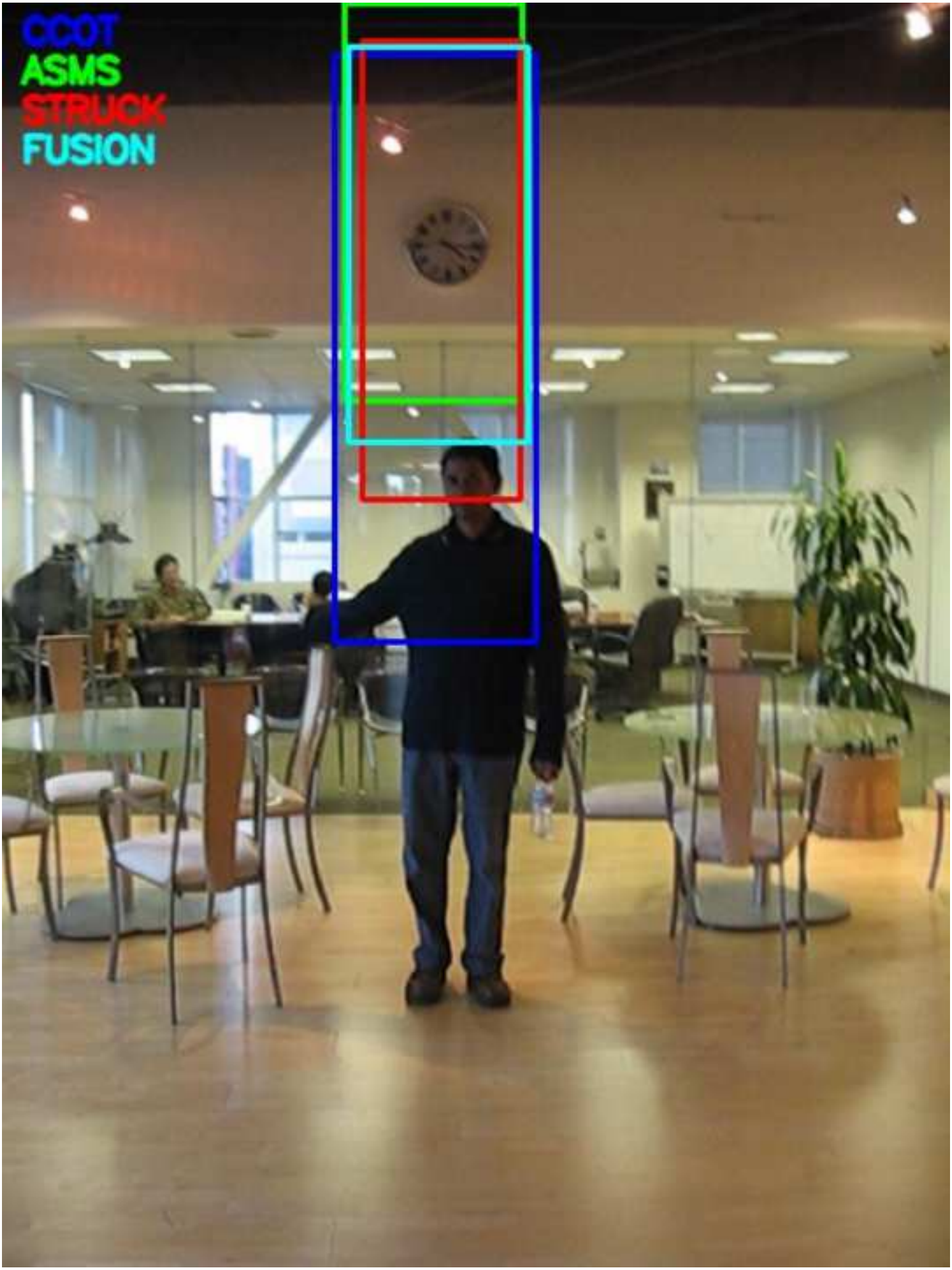
Figure

Click here to download high resolution image



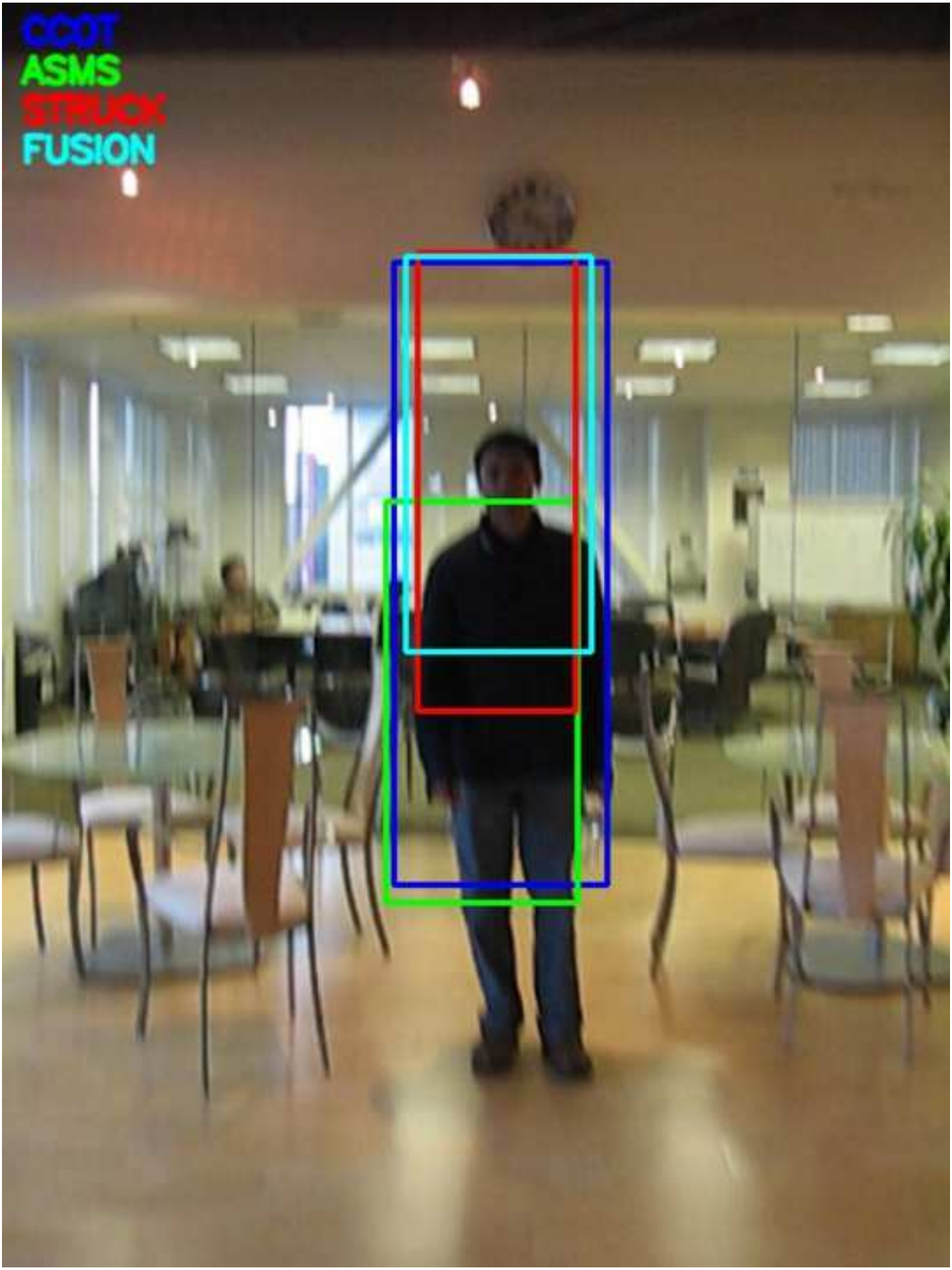
Figure

Click here to download high resolution image



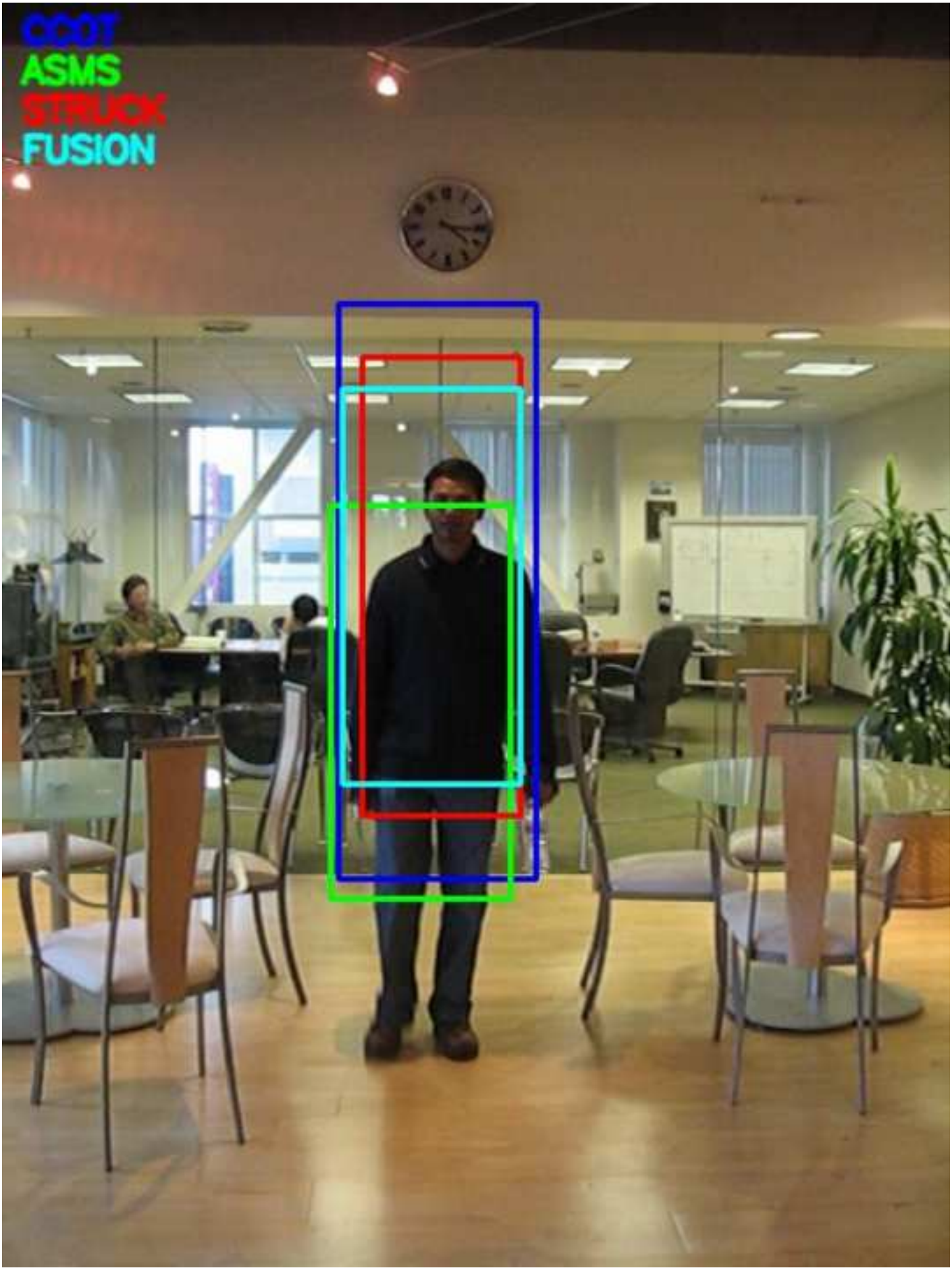
Figure

Click here to download high resolution image



Figure

Click here to download high resolution image



Figure

Click here to download high resolution image

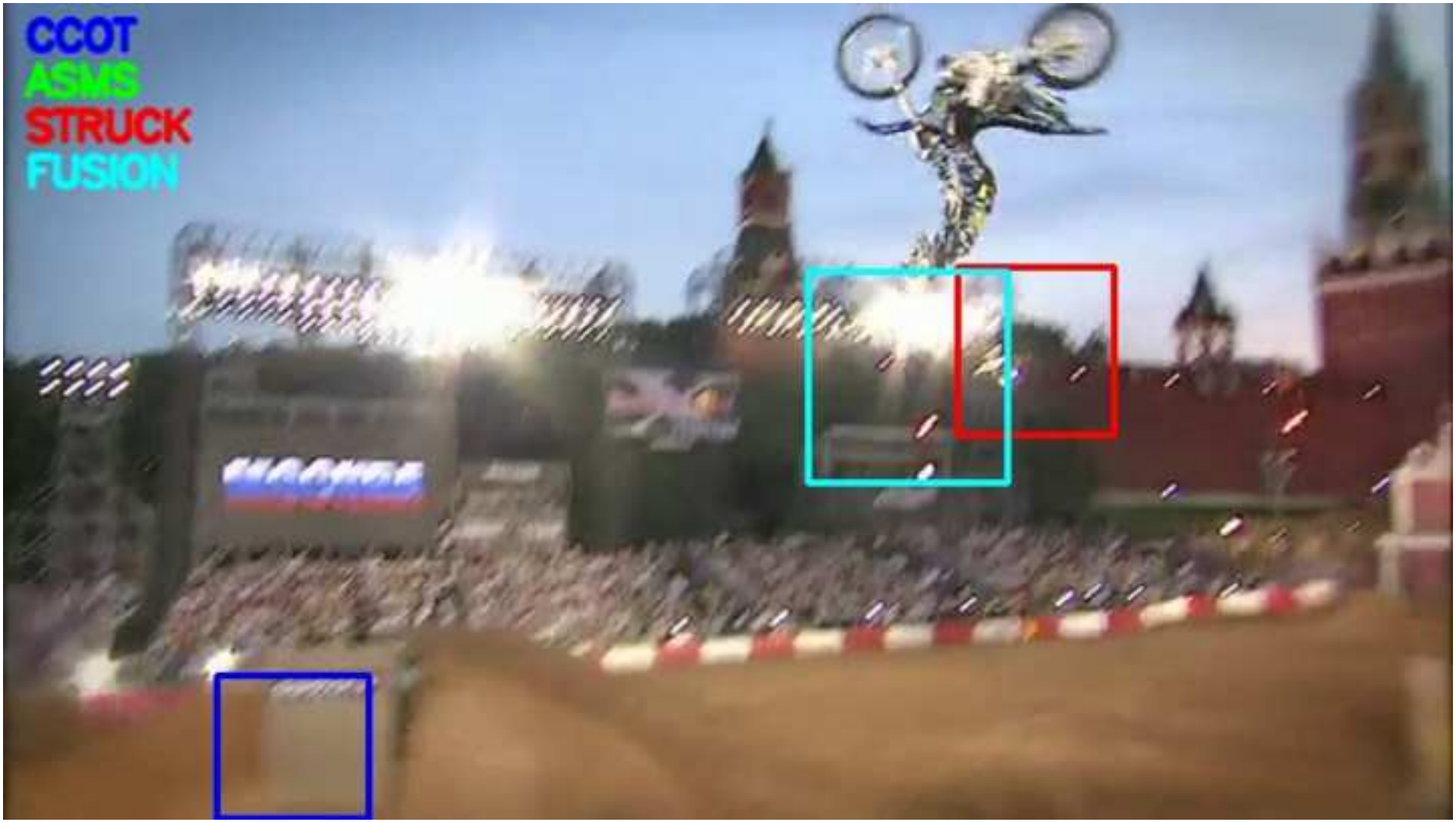
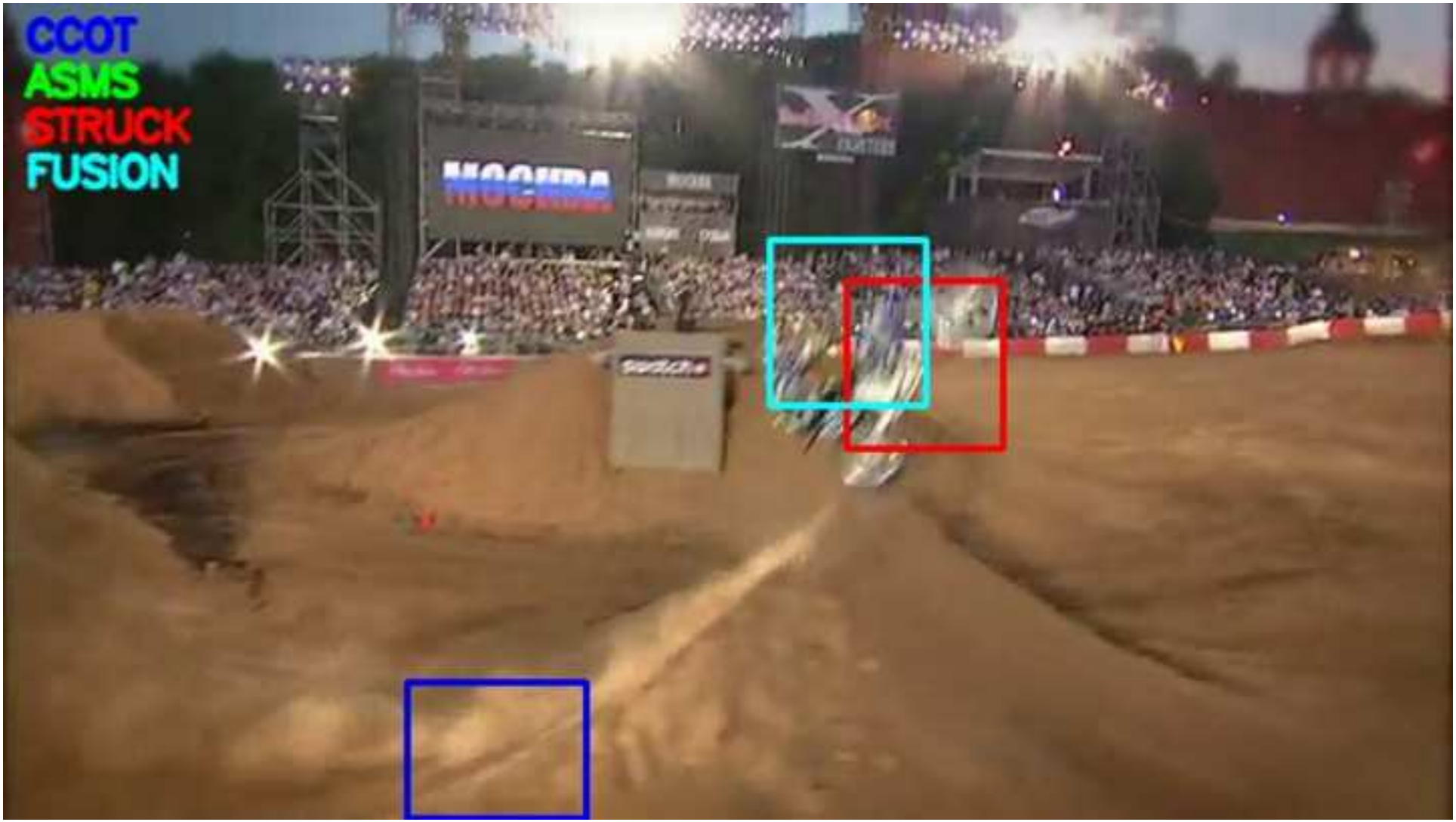


Figure
Click here to download high resolution image



Figure

Click here to download high resolution image

Figure

Click here to download high resolution image

Figure

Click here to download high resolution image

Figure

Click here to download high resolution image