Mixing Hough and Color Histogram Models for Accurate Real-Time Object Tracking

Antoine Tran and Antoine Manzanera

U2IS, ENSTA ParisTech
Université de Paris-Saclay
828, Bd des Maréchaux
91762 Palaiseau CEDEX - France
antoine.[tran,manzanera]@ensta-paristech.fr

Abstract. This paper presents a new object tracking algorithm, which does not rely on offline supervised learning. We propose a very fast and accurate tracker, exclusively based on two complementary low-level features: gradient-based and color-based features. On the first hand, we compute a Generalized Hough Transform, indexed by gradient orientation. On the second hand, a RGB color histogram is used as a global rotation-invariant model. These two parts are processed independently, then merged to estimate the object position. Then, two confidence maps are generated and combined to estimate the object size. Experiments made on VOT2014 and VOT2015 datasets show that our tracker is competitive among all competitors (in accuracy and robustness, ranked in the top 10 and top 15 respectively), and is one of the few trackers running at more than 100 fps on a laptop machine, with one thread. Thanks to its low memory footprint, it can also run on embedded systems.

1 Introduction

Object tracking is a very popular task in computer vision. Basically, the goal is to accurately localize one defined object (the target) in a video. Among difficulties, we can cite object deformation, motion change, rotation, scaling or those linked to the context (illumination change, occlusion, camouflage, camera motion). Applications such as human-computer interaction or augmented reality require reactive algorithms, so the computational cost may be a critical issue.

For efficiency, our tracker is based on very light methods, and combines low-level shape and color features. It is a model-free tracker, meaning that the offline training is done only at the first frame of the sequence. The gradient orientation is computed and used as an index of a Generalized Hough Transform (GHT) [2]. A RGB histogram is used to represent the color aspect of the target object, and distinguish it from the background. These two parts are processed independently, and merged to finally estimate the object location. They are complementary: the original GHT is robust to illumination, but weak against scale and rotation, unlike the color histogram model.

By testing and evaluating our tracker on VOT2014 and VOT2015 [15, 16], we show that this combination leads to high performance in terms of accuracy and robustness. Moreover, by associating low-level features and light algorithms, our tracker can run at more than 100 fps on a laptop machine, without explicit multithreading. It is one of the fastest among all competitors, while being ranked second among real-time trackers in terms of accuracy and robustness criteria from VOT2015. Its lightness also makes it suitable to embedded systems.

This paper is organized that way: after a short state-of-the-art of object tracking, we will explain our method. Finally, we will show some results obtained in the academic datasets.

2 Related Work

Given a sequence, object tracking consists in estimating the state of one target object at each frame. This state can be its center, its bounding box or its silhouette. Many works have addressed the tracking problem, and we refer to Yilmaz' survey for a coverage of the task [25].

Structurally, our tracker belongs to the class of trackers combining different methods [1, 3, 9, 18, 24]. Among recent works, STAPLE [3] is related to our technique: it combines correlation and color histogram model to provide an effective and fast tracker. As we are using the GHT in its original form, our template-part tracker is simpler and lighter than Bertinetto's, as it only requires gradient computation (instead of HOG features [5]). Duffner's PixelTrack [9] is also close to our tracker: it combines a GHT with a foreground/background color model into a very fast tracking algorithm (above 100 fps).

The VOT committee annually proposes a dataset to evaluate and rank trackers [14–16]. In this challenge, the most accurate trackers are based on Convolutional Neural Network [19, 20] and correlation filters [3, 7]. If we focus on the fastest algorithms of VOT15 challenge, Vojir's tracker [23] based on the Meanshift [4] proposes decent accuracy, with a speed far beyond real-time. Maresca [18] proposes a fast tracker based on estimation of object motion, obtained by the combination of several light trackers.

Our proposed tracker is based on very low-level features and lightweight operations: color histogram and GHT. Color histogram is popular in object tracking [4, 21, 23], since it is fast and robust to scale and in-plane rotation changes. The GHT is an extension of the Hough Transform [8], used to detect arbitrary shapes. It consists in considering some elements of this shape (pixels, patches) and, according to their local appearances, make them vote for potential positions of the shape center. The estimated center is then determined by the location that has accumulated the highest number of votes. The GHT is robust to illumination changes and to camouflage issue, compensating some weaknesses of the color-based model. In tracking context, the main issue of the GHT is the robustness to scale and rotation changes. However, several authors proposed Hough-based trackers [9–11, 17] by modifying the original GHT, or using complementary methods. Amongst Hough-based trackers, only Hua [13] outperforms our algorithm, but is 100 times slower. However, his algorithm essentially relies on a HOG-based detector, which estimates several candidates locations, and the Hough Transform is only used to discard wrong candidates. PixelTrack [9] is at a same order of magnitude in speed, but much less accurate and robust.

In the last VOT challenges [14,15], few trackers belong to the category of real-time trackers (21 competitors among 132), with diverse performances in accuracy and robustness. In this category, our method is one of the fastest (more than 100 fps), but also one of the most accurate and robust (only beaten by [23]). Compared to slower tracker, ours is still competitive: ranked in the top 15 for both criteria in VOT2015 [15] and ranked in the top 10 in VOT2014 [16].

Our state representation is based on a bounding box. Given a frame t, the aim is to estimate the bounding box $B_t = (c_t, w_t, h_t)$ of an object $\mathcal{O}$, where each parameter is respectively: bounding box's center, width and height.

3 Our contribution

First, we explain how to initialize our tracker. Second, we deal with state estimation. As position is estimated before scale, we will explain these two steps in different sections. Third, we explain the model updating process.

3.1 Initialization

Our tracker is initialized at the first frame $\mathbf{I}_0$, with a manually set bounding box B_0. Let $\mathbf{I}(B)$ be the restriction of an image $\mathbf{I}$ to any subset of pixels B.

First, the target RGB histogram H_0 of $\mathbf{I}_0(B_0)$, with $n_c \times n_c \times n_c$ bins, is generated ($n_c = 12$). H_t will denote the target RGB histogram at the frame t.

Second, for the geometrical model, let $\mathbf{M}_0$ and $\mathbf{\Phi}_0$ be the gradient magnitude and orientation of $\mathbf{I}_0(B_0)$. The goal is to initialize the *R-Table* R (which will be updated over the time), indexed by $n_o = 16$ orientations. It consists in considering all pixels $p \in \mathbf{I}_0(B_0)$, for which $\mathbf{M}_0(p) > \epsilon_M$ ($\epsilon_M = 70.0$) and whose quantified gradient orientation is θ_p, then to store in $R(\theta_p)$ the couple $(\overrightarrow{u} = \overrightarrow{pc_0}, \omega_{\overrightarrow{u}})$, which is the displacement from p to the bounding box center c_0, and $\omega_{\overrightarrow{u}} = 1.0$ a default weight value.

3.2 Estimation of position

Given an image $\mathbf{I}_t$ from a sequence, our tracker first estimates the object center, then its scale. We will explain these two steps independently.

On the one hand, we perform a basic GHT. As during initialization, it requires to evaluate gradient image from $\mathbf{I}_t$, keep pixels p whose gradient magnitude is above ϵ_M, and quantify its orientation as θ_p. Then, for each couple $(\overrightarrow{u}, \omega_{\overrightarrow{u}}) \in R(\theta_p)$, p votes for all displacements $p + \overrightarrow{u}$ with the weight $\omega_{\overrightarrow{u}}$. More formally:

$$\mathbf{HT}_t(p) = \sum_q \sum_{(\overrightarrow{u}, \omega) \in R(\theta_q)} \omega \cdot \delta(p, q + \overrightarrow{u}) \tag{1}$$

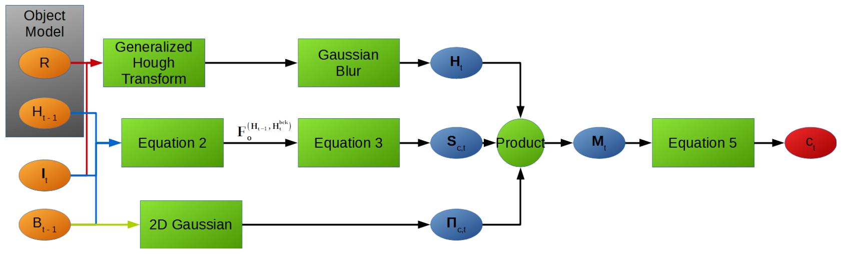


Fig. 1: Position tracking diagram.

with δ the Kronecker function. Finally, when all pixels have voted, the created map (the Hough Transform) $\mathbf{HT}_t$ emphasizes the most probable locations of the object center, with respect to the geometrical model. The GHT is performed under its simplest form, unlike [9] who indexes their R-Table using gradient and color features and [11, 17] who use machine learning classifiers.

We then blur $\mathbf{HT}_t$ (3×3 discrete Gaussian filter), in order to add robustness to deformation. The GHT is intrinsically robust to illumination change (conservation of the orientation), but not to scaling (pixels votes spread away from the center) nor to rotation (pixels vote according to the wrong list in the R-Table).

On the other hand, to exploit the color model, we define foreground and background areas. We use the color model histogram H_{t-1}, and build a background histogram H_t^{bck}. These features have the advantage to compensate for the GHT weaknesses, and vice-versa. Given the last estimated bounding box B_{t-1}, let S_t be the background area, defined by all pixels inside the bounding box of center c_{t-1}, and dimension $(\alpha \cdot w_{t-1}, \alpha \cdot h_{t-1})$ ($\alpha = 2.0$), excluding B_{t-1}. From S_t, we build H_t^{bck}. Then, for every pixel $p \in (B_{t-1} \cup S_t)$, let q_t^p be its quantified color in $\mathbf{I}_t$. As proposed in [21], we define the foregroundness with respect to the object $\mathcal{O}$, knowing the object color histogram H_{t-1} and the background color model:

$$\mathbf{F}_{\mathcal{O}}^{(H_{t-1},H_t^{bck})}(p) = \begin{cases} \frac{H_{t-1}(q_t^p)}{H_{t-1}(q_t^p)+H_t^{bck}(q_t^p)} & \text{if } p \in (B_{t-1} \cup S_t) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$\mathbf{F}_{\mathcal{O}}^{(H_{t-1},H_t^{bck})}(p)$ indicates how probably p belongs to the target. Unlike [21], we do not combine Eq. 2 with a distractor-aware model, as we aim to remain as simple as possible. Compared to PixelTrack [9], this method only needs a model histogram and no prior information about background. Then, given a bounding box B of size (w_{t-1}, h_{t-1}) inside $(B_{t-1} \cup S_t)$, let $\mathbf{S}_{c,t}(B)$ be the normalized score evaluating whether the target is inside B:

$$\mathbf{S}_{c,t}(B) = \frac{\sum_{p \in B} F_{\mathcal{O}}^{(H_{t-1},H_t^{bck})}(p)}{w_{t-1} \cdot h_{t-1}} \tag{3}$$

This formulation is simpler than Possegger's [21], who proposed a method to discard distractors. We also use a prediction map, that indicates the likelihood to find c_t at x, given B_{t-1}:

$$\mathbf{\Pi}_t^{B_{t-1}}(x) = \exp\left(\frac{-(x - c_{t-1})^2}{2 \cdot \min(w_{t-1}, h_{t-1})^2}\right) \tag{4}$$

Finally, for all pixels x, we define $\mathbf{M}_t$ such as $\mathbf{M}_t(x) = \mathbf{HT}_t(x) \cdot \mathbf{S}_{c,t}(B_x) \cdot \mathbf{\Pi}_t^{B_{t-1}}(x)$ with B_x the rectangle centered in x of size (w_{t-1}, h_{t-1}). From $\mathbf{M}_t$, we finally estimate the object position c_t as follows:

$$c_t = \begin{cases} \underset{x}{\text{argmax}}(\mathbf{M}_t(x)) & \text{if } \max_x(\mathbf{M}_t(x)) \neq 0 \\ c_{t-1} + \overrightarrow{c_{t-2}c_{t-1}} & \text{otherwise} \end{cases} \tag{5}$$

The second case of Eq. 5 assumes that, when the support of $\mathbf{HT}_t$ and $\mathbf{S}_{c,t}$ are disjoint, the target is translating with a vector $\overrightarrow{c_{t-2}c_{t-1}}$. We choose a pixel-wise multiplication to merge our two trackers, unlike STAPLE [3] who used a weighted average. In that way, we do not have to deal with the difference of magnitude of $\mathbf{HT}_t$ and $\mathbf{S}_{c,t}$, and to adjust a weight. We also differ from PixelTrack [9], where the final position is obtained by linear combination of the centers estimated by the GHT on one hand and the color segmentation map on the other hand. Fig. 1 describes all steps for our position estimation.

3.3 Estimation of scale

The second step consists in estimating object scale.

On the one hand, from the GHT, let us define the backprojection map $\mathbf{BP}_t$, for all pixels p fulfilling conditions to be stored in the R-Table:

$$\mathbf{BP}_t(p) = \frac{\sum_{(\overrightarrow{u},\omega) \in R(\theta_p)} \mathbf{M}_t(p + \overrightarrow{u})}{|R(\theta_p)|} \tag{6}$$

with $|R(\theta_p)|$ the cardinality of $R(\theta_p)$. The approach is similar to Duffner's one [9]. However, Duffner only backprojects the peak of the GHT, while we consider the sum of the voted positions for all pixels. In both cases, the made assumption is that the higher the backprojection is, the more likely it belongs to the target.

On the other hand, we consider $\mathbf{F}_{\mathcal{O}}^{(H_{t-1}, {H_t}^{bck})}$, defined Eq. 2, as a color confidence map. Then, let $\mathbf{BF}_t$ be the final confidence map:

$$\mathbf{BF}_t = 0.5 \cdot (\mathbf{BP}_t + \mathbf{F}_{\mathcal{O}}^{(H_{t-1}, {H_t}^{bck})}) \tag{7}$$

Fig. 2 illustrates these maps. Then, inspired by Posseger [21], we consider the set of object pixels $\text{OP}_t = \{p | \mathbf{BF}_t(p) > 0.5\} \cup R_t$, with R_t a safe foreground area defined as the rectangle centered on c_t, of size $(\beta \cdot w_{t-1}, \beta \cdot h_{t-1})$ ($\beta = 0.20$). From OP_t, we only retain the connected component that contains c_t, to discard isolated pixels that could generate scale overestimation. Finally, we estimate a potential bounding box by computing the bounding box $\bar{B}_t$ of this connected

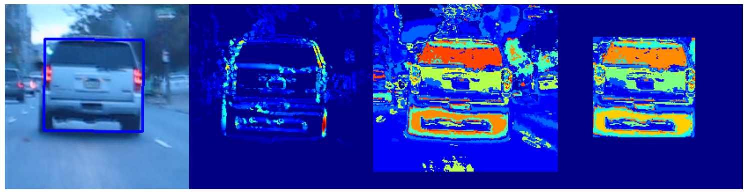

(a) $\mathbf{I}_t$ (b) $\mathbf{BP}_t$ (c) $\mathbf{F}_{\mathcal{O}}^{(H_{t-1},H_t^{bck})}$ (d) $\mathbf{BF}_t$

Fig. 2: Cropped frame from *car1* from VOT2015, with the ground truth in blue. Backprojection map $\mathbf{BP}_t$ and pixel color likelihood map $\mathbf{F}_{\mathcal{O}}^{(H_{t-1},H_t^{bck})}$ are complementary: while the first one indicates which border pixels are more likely to belong to the target, the second one gives high results for pixels of car's back.

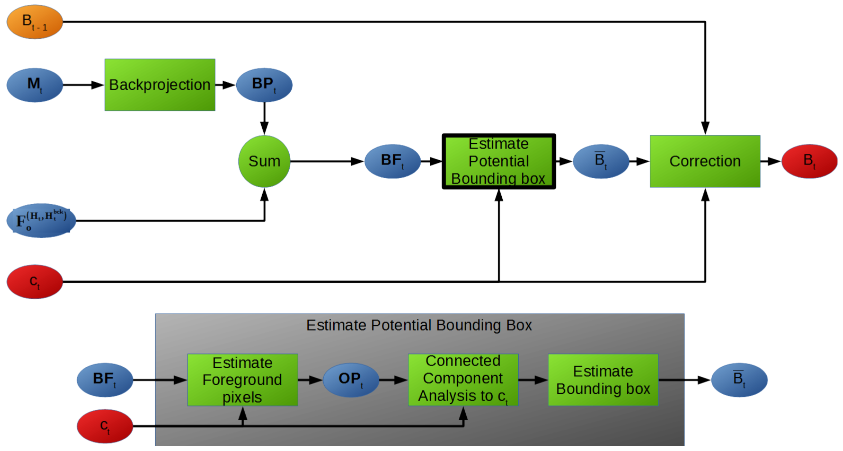

B_{t-1}
M_t
Backprojection
BP_t
Sum
BF_t
Estimate Potential Bounding box
B̄_t
Correction
B_t
F_o^{(H_t,H_t^{bck})}
c_t
Estimate Potential Bounding Box
BF_t
Estimate Foreground pixels
OP_t
Connected Component Analysis to c_t
Estimate Bounding box
B̄_t
c_t

Fig. 3: Scale estimation diagram.

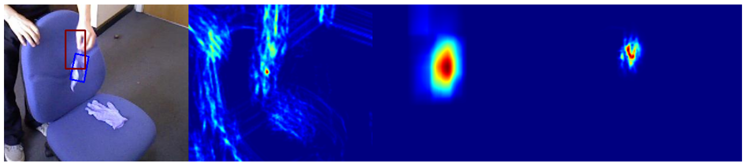
component. Then, we reject bounding box sizes whose relative area variation with respect to B_{t-1} is above 5%. Otherwise, we update object's size using the same aspect ratio, as follows:

$$X_t = \lambda_t \cdot X_{t-1} \tag{8}$$

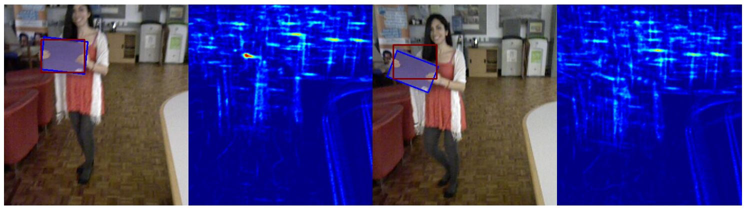
with $X \in \{w, h\}$ and $\lambda_t = \min(1.05, \max(0.95, \frac{\mathcal{A}(\bar{B}_t)}{\mathcal{A}(B_{t-1})}))$ ($\mathcal{A}$ being the area operator). Fig. 3 summarizes scale estimation operations. Finally, to prepare the updating process, let $\mathbf{SG}_t$ be the shape and color-based confidence map such that $\mathbf{SG}_t(p) = \mathbf{BF}_t(p)$ if $p \in B_t$, and 0 otherwise.

3.4 Model update

The last step consists in updating the model, knowing the estimated bounding box B_t. To update the color model, let H_t^o be the color histogram of $\mathbf{I}_t(B_t)$ and



(a) Cropped frame (*glove*) (b) $\mathbf{HT}_t$ (c) $\mathbf{S}_c, t$ (d) $\mathbf{M}_t$



(e) Cropped frame 1 (*book*) (f) $\mathbf{HT}_1$ (g) Cropped frame 2 (h) $\mathbf{HT}_2$

Fig. 4: The first line illustrates failures due to the transparency of the glove (ground truth in blue, tracker output in red), making its color similar to the chair. The GHT still performs correctly (sharp peak on the second image).
On the second line the GHT fails due to the rotation of the book: the peak disappears from $\mathbf{HT}_1$ to $\mathbf{HT}_2$.

$\mu_c = 0.05$ the color updating rate:

$$H_t = (1 - \mu_c) \cdot H_{t-1} + \mu_c \cdot H_t^o. \tag{9}$$

To update the R-Table R, we start by reducing all displacement weights:

$$\forall \theta, \forall (\overrightarrow{u}, \omega_{\overrightarrow{u}}) \in R(\theta), \omega_{\overrightarrow{u}} \leftarrow (1 - \mu_g) \cdot \omega_{\overrightarrow{u}} \tag{10}$$

Then, considering the confidence map $\mathbf{SG}_t$, and the object center c_t, for all pixels $p \in B_t$ with gradient orientation θ_p, we consider the displacement $\overrightarrow{v} = \overrightarrow{pc_t}$, with a weight equal to $\mu_g \cdot \mathbf{SG}_t(p)$. Then, we consider two cases:

- if $\overrightarrow{v}$ is in $R(\theta_p)$, we increment its weight by $\mu_g \cdot \mathbf{SG}_t(p)$ ($\mu_g = 0.05$), in order to reinforce the most relevant elements of the R-Table
- otherwise we add an entry into $R(\theta_p)$ with the weight $\mu_g \cdot \mathbf{SG}_t(p)$

Finally, for each index of the R-Table, we keep only the $N_R = 200$ displacements with the strongest weights, to limit computational and memory cost.

4 Experiments

Before dealing with experiments on academic datasets, we will detail our implementation setup.

4.1 Implementations details

Our algorithm is developed using C++ and the OpenCV 2.4.9 library, and tested on a laptop at 2.4 GHz, without explicit multithreading. In terms of implementation, iterative pixel access is done using image pointers, and image histogram computation is done using Look Up Table. At frame t, our tracker only processes area centered in c_{t-1}, and of size two times the last bounding box area. To evaluate the map $\mathbf{S}_{c,t}$, we use integral images. Otherwise, no major algorithmic optimization has been done.

In terms of memory footprint, target's informations, composed of the color histogram (${n_c}^3 = 12^3$ floating point numbers), the R-Table ($n_0 \cdot N_R \cdot 2 \cdot 4$ integers for displacements and $n_0 \cdot N_R \cdot 2$ floats for weights) and the two last states (4 integers for coordinates and scales), resulting, with our set of parameters, in a memory footprint of about 45 ko. This quantity is independent of target' size. It is however negligible compared to the number of temporary images: 2 8-bit images (gradient orientation and magnitude maps), 1 RGB image (the sub-image in which we are tracking the target) and 5 32-bit images (float) ($\mathbf{HT}_t$, $\mathbf{BP}_t$, $\mathbf{S}_{c,t}$, $\mathbf{F}_{\mathcal{O}}^{(H_{t-1},H_t^{bck})}$ and $\mathbf{BF}_t$), which depends on object's size (and its associated search window): for a 100×100 object's size, the footprint will reach 2 Mo. Considering the speed obtained experimentally, we are convinced that our method is suitable embedded systems. The code of our implementation will be made available soon. All parameters have been tuned for the best trade-off between performance and speed on VOT2015 [15]. For experiments case, we will denote as CHT the position tracker only, and CHTs our complete tracker.

4.2 Results on VOT 2014 and VOT 2015

Each year, the VOT committee proposes a dataset to test and evaluate trackers. Each frame from each sequence is labeled according to its difficulty (occlusion, camera motion, size, motion or illumination change). Evaluation criteria are:

- Accuracy: based on overlap measure $O(GT_t, B_t) = \frac{GT_t \cap B_t}{GT_t \cup B_t}$, with GT_t the ground truth at the frame t
- Robustness: given by the number of failures (frames where $O(GT_t, B_t) = 0$)
- Speed: based on a normalized speed (EFO units, see [15])

The VOT committee provides results of all competitors, and a toolkit to evaluate and rank trackers. For all experiments, we use the function *report_challenge* to get weighted mean rank (based on ranks for all difficulties), pooled rank (based on all sequences), expected overlap, and speed. We also compute ranking with the whole set of results, but only display those of relevant trackers. Results for VOT2014 and VOT2015 are summarized Tab. 1.

VOT2014 [16] is a dataset composed of 25 sequences, and results for 40 trackers are available. We choose to show our results compared to DSST [6] (VOT2014 winner), Hough-based trackers [9,17,18] and real-time trackers [9,12, 18,22]. Amongst Hough-based trackers, our method is as well ranked as Matflow,

Tracker	Weighted mean rank		Pooled rank		Expected	Speed	
	Accuracy	Robustness	Accuracy	Robustness	overlap	(EFO)	(fps)
VOT2014							
CHTs	8.83	4.33	6	9	0.2960	129.77	159.21
CHT	8.42	4.08	6	9	0.2916	109.75	134.65
DSST [6]	1.83	4.33	1	3	0.3693	5.80	13.07
Matrioska [17]	9.83	12	6	9	0.2671	10.20	21.88
bdf [18]	10.67	7.50	10	9	0.3097	46.82	100.45
Matflow [17, 18]	8.67	3.17	6	4	0.3120	19.08	40.94
FoT [22]	7	18.17	6	22	0.2859	114.64	306.52
PTp [9]	25.33	11.17	30	9	0.2519	49.89	127.87
KCF [12]	2	4.67	1	5	0.3641	24.23	63.42
VOT2015							
CHTs	12.67	13.67	13	17	0.2606	103.89	111.22
CHT	13.83	15.67	13	20	0.2615	101.91	109.10
Staple [3]	1	4.33	1	5	0.345		
ASMS [23]	7.50	11	2	13	0.2353	115.09	142.26
bdf [18]	29.33	32	27	43	0.2054	200.24	78.43
FoT [22]	19.50	42.50	16	53	0.1934	143.62	177.53
DSST [6]	4.0	23.67	1	38	0.2707	3.29	4.47
DAT [21]	13.73	17.33	6	20	0.2428	9.82	14.87
HT [11]	20	28.50	13	43	0.2045	0.91	0.56
Matflow [17, 18]	22.17	27.33	23	43	0.2098	81.34	31.86
MDNET [20]	1	1.33	1	1	0.3789	0.87	0.97
sPST [13]	1.67	4.50	1	5	0.3134	1.03	1.16

Table 1: Results in VOT2014 and VOT2015 for different trackers. Real-time trackers appear in bold.

combining Matrioska [17] and bdf [18], but is 6 times faster (in EFO units). In the real-time trackers category, ours is the second fastest one, beaten by FoT [22] but our method is much more robust. KCF [12] is more accurate and robust than our method, but is slower. Globally, our tracker is well ranked (top 10 among 40 competitors) but is one of the tracker proposing the best trade-off between effectiveness and speed. Surprisingly, our position-only tracker (CHT) and our complete one have similar performance in terms of accuracy and robustness, but the complete one is faster (probably due to object size reduction).

VOT2015 challenge is composed of 60 sequences and results for 62 competitors are available. Compared to MDNET [20], VOT2015 winner based on CNNs, and STAPLE [3] (results from author's website, without speed results) our tracker is less effective, but faster (being lighter, and knowing that the author mentioned 80 fps on a 4.0GHz CPU, we expect our method to be faster than STAPLE). Compared to STAPLE, our shape-based tracker relies on the GHT, with gradient computation only, while STAPLE requires a more complex algorithm (correlation-based tracker using HOG features). Amongst real-time trackers [18, 22, 23], our tracker is one of the few above 100 EFO, and is only beaten by Vojir's extension of Meanshift [23] in rankings and speed, but with

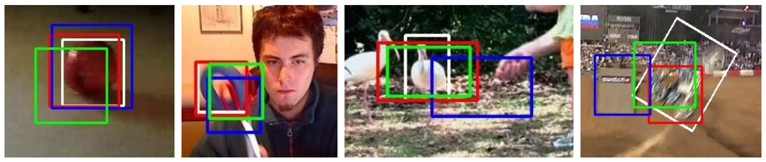


Fig. 5: Results from VOT2014 and 2015.
White bounding boxes are ground truth, red ones are obtained with our tracker. The two first images are cropped from *sphere* and *torus* sequences from VOT2014 (PixelTrack in blue, and MatFlow in green).
The two last are cropped from *birds2* and *motocross* sequences from VOT2015 (DAT in blue, and STAPLE in green).

slightly higher expected overlap. Otherwise, we are better ranked than other real-time trackers. We perform better than other Hough-based trackers [11, 17], including Matflow, which was on par with our tracker in VOT2014, but excluding sPST [13], which is 100 times slower and relies on an object detector. Compared to Possegger [21], from which our color part tracker is inspired, we demonstrate the usefulness of the Hough part, since, in weighted mean ranking, our method is slightly more robust. Some results are shown Fig 4 and Fig 5. We also used VOT2015 to see performances of each part of the tracker (Hough and color ones). In both cases, the loss of performance is dramatic, as we can see on Fig. 6, where we show the results of different versions in terms of expected overlap (the higher, the better) and failures (the lower, the better) for several difficulties, both obtained on the set of frames concerned by the difficulties.

5 Conclusion

In this paper, we proposed a tracker working without offline training and tracking arbitrary objects. Unlike most state-of-the-art trackers, our method is based on a very low level representation. First, our geometrical model is only based on gradient, through a GHT. Then, an object color histogram is used to generate a map indicating the likeliness of the pixel to belong to the object. These two operations, done independently, are combined to estimate object center. Second, from the two features, we generate two confidence maps, and merge them in order to estimate the object size. The final confidence map is built from the fusion of these two maps and used to update the object geometrical and color models. Our whole tracker relies on computationally efficient operations, and performs tracking task beyond real-time (about 100 fps) with a low memory footprint. Experiments were done on recent academic dataset [15, 16], for which our tracker is ranked in the first third for accuracy and robustness. It is also one of the fastest from VOT2014 and VOT2015 challenges. Thanks to its speed and low memory footprint, our algorithm can be implemented on embedded systems, combined with other trackers to improve accuracy and robustness or with object detection or background subtraction to automatically initialize tracked regions.

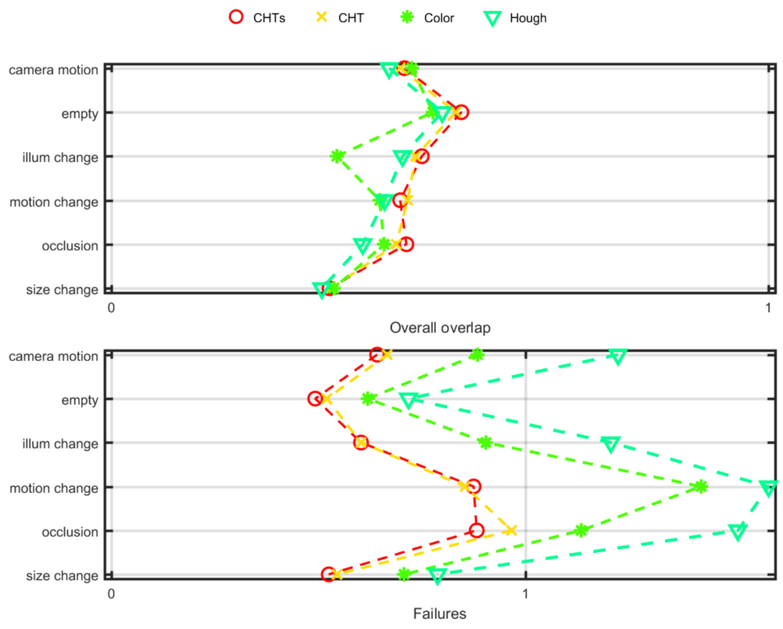


Fig. 6: Expected overlap and number of failures for our complete tracker, and for partial versions (position tracking only, Hough only and Color only)

Acknowledgments

We gratefully acknowledge financial support from the French Government Defense procurement and technology agency (DGA/MRIS).

References

1. Babenko, B., Yang, M.H., Belongie, S.: Robust object tracking with online multiple instance learning. IEEE transactions on pattern analysis and machine intelligence 33(8), 1619–1632 (2011)
2. Ballard, D.H.: Generalizing the Hough transform to detect arbitrary shapes. Pattern recognition 13(2), 111–122 (1981)
3. Bertinetto, L., Valmadre, J., Golodetz, S., Miksik, O., Torr, P.: Staple: Complementary learners for real-time tracking. arXiv preprint arXiv:1512.01355 (2015)
4. Comaniciu, D., Ramesh, V., Meer, P.: Real-time tracking of non-rigid objects using mean shift. In: Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on. vol. 2, pp. 142–149. IEEE (2000)
5. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). vol. 1, pp. 886–893. IEEE (2005)
6. Danelljan, M., Häger, G., Khan, F., Felsberg, M.: Accurate scale estimation for robust visual tracking. In: British Machine Vision Conference, Nottingham, September 1-5, 2014. BMVA Press (2014)

7. Danelljan, M., Robinson, A., Khan, F.S., Felsberg, M.: Beyond correlation filters: Learning continuous convolution operators for visual tracking. In: European Conference on Computer Vision. pp. 472–488. Springer (2016)
8. Duda, R.O., Hart, P.E.: Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM 15(1), 11–15 (1972)
9. Duffner, S., Garcia, C.: Pixeltrack: a fast adaptive algorithm for tracking non-rigid objects. In: Computer Vision (ICCV), 2013 IEEE International Conference on. pp. 2480–2487. IEEE (2013)
10. Gall, J., Yao, A., Razavi, N., Van Gool, L., Lempitsky, V.: Hough forests for object detection, tracking, and action recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33(11), 2188–2202 (2011)
11. Godec, M., Roth, P.M., Bischof, H.: Hough-based tracking of non-rigid objects. Computer Vision and Image Understanding 117(10), 1245–1256 (2013)
12. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE Transactions on Pattern Analysis and Machine Intelligence 37(3), 583–596 (2015)
13. Hua, Y., Alahari, K., Schmid, C.: Online object tracking with proposal selection. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3092–3100 (2015)
14. Kristan, M., Leonardis, A., Matas, J., al.: The visual object tracking VOT2016 challenge results (2016)
15. Kristan, M., Matas, J., Leonardis, A., al.: The visual object tracking VOT2015 challenge results. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. pp. 1–23 (2015)
16. Kristan, M., Pflugfelder, R., Leonardis, A., Matas, J., al.: The visual object tracking vot2014 challenge results (2014), `http://www.votchallenge.net/vot2014/program.html`
17. Maresca, M.E., Petrosino, A.: Matrioska: A multi-level approach to fast tracking by learning. In: Image Analysis and Processing–ICIAP 2013, pp. 419–428. Springer (2013)
18. Maresca, M.E., Petrosino, A.: Clustering local motion estimates for robust and efficient object tracking. In: European Conference on Computer Vision. pp. 244–253. Springer (2014)
19. Nam, H., Baek, M., Han, B.: Modeling and propagating cnns in a tree structure for visual tracking. arXiv preprint arXiv:1608.07242 (2016)
20. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4293–4302 (2016)
21. Possegger, H., Mauthner, T., Bischof, H.: In defense of color-based model-free tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2113–2120 (2015)
22. Vojíř, T., Matas, J.: The enhanced flock of trackers. In: Registration and Recognition in Images and Videos, pp. 113–136. Springer (2014)
23. Vojir, T., Noskova, J., Matas, J.: Robust scale-adaptive mean-shift for tracking. In: Scandinavian Conference on Image Analysis. pp. 652–663. Springer (2013)
24. Yang, F., Lu, H., Yang, M.H.: Robust visual tracking via multiple kernel boosting with affinity constraints. IEEE Transactions on Circuits and Systems for Video Technology 24(2), 242–254 (2014)
25. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM computing surveys (CSUR) 38(4), 13 (2006)