How to Make an RGBD Tracker ?

Uğur Kart[1][0000−0002−3728−2473] Joni-Kristian Kämäräinen[1][0000−0002−5801−4371]
Jiří Matas[2][0000−0003−0863−4844]

[1] Laboratory of Signal Processing
Tampere University of Technology, Tampere, Finland
{ugur.kart,joni.kamarainen}@tut.fi
[2] The Center for Machine Perception
Czech Technical University, Prague, Czech Republic
matas@cmp.felk.cvut.cz

Abstract. We propose a generic framework for converting an arbitrary short-term RGB tracker into an RGBD tracker. The proposed framework has two mild requirements – the short-term tracker provides a bounding box and its object model update can be stopped and resumed. The core of the framework is a depth augmented foreground segmentation which is formulated as an energy minimization problem solved by graph cuts. The proposed framework offers two levels of integration. The first requires that the RGB tracker can be stopped and resumed according to the decision on target visibility. The level-two integration requires that the tracker accept an external mask (foreground region) in the target update. We integrate in the proposed framework the Discriminative Correlation Filter (DCF), and three state-of-the-art trackers – Efficient Convolution Operators for Tracking (ECOhc, ECOgpu) and Discriminative Correlation Filter with Channel and Spatial Reliability (CSR-DCF). Comprehensive experiments on Princeton Tracking Benchmark (PTB) show that level-one integration provides significant improvements for all trackers: DCF average rank improves from 18th to 17th, ECOgpu from 16th to 10th, ECOhc from 15th to 5th and CSR-DCF from 19th to 14th. CSR-DCF with level-two integration achieves the top rank by a clear margin on PTB. Our framework is particularly powerful in occlusion scenarios where it provides 13.5% average improvement and 26% for the best tracker (CSR-DCF).

Keywords: Visual Object Tracking · RGBD Tracking

1 Introduction

Short-term visual object tracking has been an active research topic in computer vision due to its widespread application areas. In recent years, the community has witnessed rapid development and seen many successful trackers emerging thanks to standardized evaluation protocols and publicly available benchmarks and competitions [1–6]. In order to adapt to target appearance changes, short-term trackers update their tracking models over time. However, that makes them prone to model corruption and drifting in case of persistent occlusions when the tracker adapts to the occluding object and starts to track it.

To avoid corruption, a tracker should be able to discriminate between the target object and the rest of the scene so that it can stop model updating if the target is occluded. However, this is a challenging task in the RGB space if there are occluders with similar visual appearance. To alleviate this issue, adding the depth cue to short-term trackers is intuitive; even if a tracked object is occluded by another object with similar appearance, the difference in their depth levels will be distinctive and will help to detect the occlusion. The availability of affordable depth sensors makes adoption of the depth cue even more attractive.

Since the depth channel lacks texture, depth itself may not provide useful information for visual tracking. On the other hand, RGB trackers perform competitively as long as no occlusions occur (see Table. 1). Therefore, our work aims at benefiting from the huge amount of effort that has been put on generic short-term RGB trackers and adopts depth as means for occlusion detection. As a novel solution, we propose a generic framework that can be used with any VOT compliant [6] short-term tracker to convert it into an RGBD tracker with depth-augmented occlusion detection. By applying the proposed framework through a clear interface and not changing the internal structure of a short-term tracker, a fast integration is ensured and the framework will benefit from ever improving short-term tracker performance in the future.

The proposed framework contains two main components: *short-term failure detection* and *recovery from occlusion*. Short-term failure detector continuously evaluates the target region to decide whether to allow the short-term tracker to update its model or switch to the recovery from occlusion mode. The framework also contains an optional, powerful third component which can be used with RGB trackers that accepts *foreground masks* that explicitly indicate occluded regions that do not belong to the target(*e.g.* CSR-DCF [7]).

The main contributions of this paper are:

- A generic framework to convert an arbitrary RGB short-term tracker into an RGBD tracker.
- Formulation of the framework's core component - *non-occluded foreground segmentation* - as an energy function of three terms, depth, color and spatial prior, which is optimized using graph cuts.
- RGBD versions of one baseline and three state-of-the-art short-term RGB trackers: DCF [8], ECO (ECOhc and ECOgpu variants) [9] and CSR-DCF [7]

The rest of the paper is organized as follows; Section 2 summarizes existing literature on generic, short-term tracking and RGBD trackers, Section 3 explains the proposed framework in detail, Section 4 provides the experiments and finally Section 5 concludes the paper.

2 Related Work

The aim of the provided generic framework is to convert any existing short-term RGB tracker into an RGBD tracker. We are motivated by the facts that the field

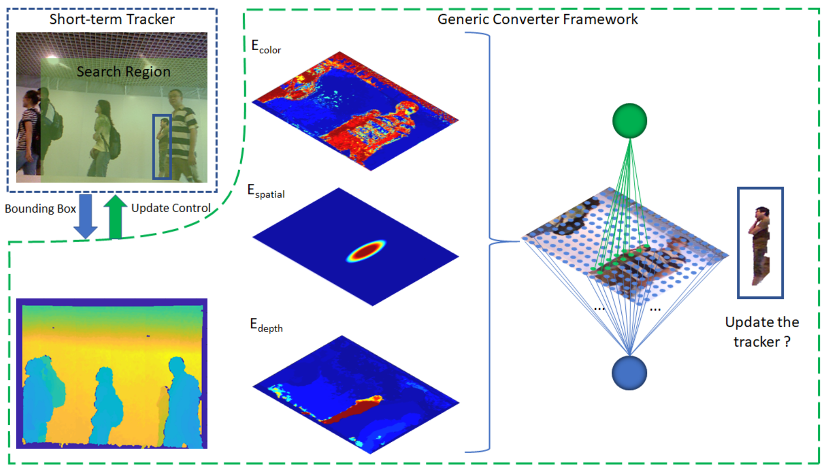

Fig. 1. Overview of the proposed framework. The short-term RGB tracker provides bounding box coordinates to the framework to be used for segmenting the visible target region with the help of depth. Using the ratio of the visible region to the bounding box, occlusions are detected hence, short-term tracker update is stopped and recovery mode is started. If the object is re-detected during recovery, the RGB tracker is resumed.

of short-term RGB tracking progresses on a steady basis and RGB provides a strong cue for tracking. On the other hand, we also believe that depth can be used as a complementary cue to instruct when a short-term tracker should be stopped and switched to recovery mode. In this sense, the proposed framework benefits from state-of-the-art short-term trackers which are briefly surveyed in addition to the recent RGBD trackers.

RGB Trackers – generic, short-term visual object tracking on RGB videos is a well-established research topic in computer vision and the main approaches can be grouped under two main categories. In *Generative Trackers*, a target model is stored and the goal is to find the best matching region in the next frame. A few descriptive examples for this category are Incremental Visual Tracking (IVT) [10], Structural Sparse Tracking (SST) [11] and kernel-based object tracking [12]. On the other hand, *Discriminative Trackers* continuously train a classifier using positive and negative samples that are acquired during the tracking process. Prominent examples of this category are Tracking-Learning-Detection (TLD) [13], Continuous Convolutional Operators Tracking (CCOT) [14], Multi-Domain Convolutional Neural Networks (MDNet) [15], Efficient Convolution Operators for Trackers (ECO) [9], and Discriminative Correlation Filter with Channel and Spatial Reliability (CSR-DCF) [7]. Due to their success in the last few years, discriminative trackers have been widely adopted in the recent works. For example, in the VOT 2017 challenge, 67% of the submissions were from this category [6]. However, training a classifier can be computationally expensive which has prompted the adoption of simple yet powerful methods for the training stage. Starting with the seminal work of Bolme *et al.* [16], Discriminative Correlation Filter (DCF) based trackers have gained momentum due to their performance, fast model update (training) and mathematical elegance. Henriques *et al.* [17] proposed a method for efficient training of multiple samples that improves performance while providing very high FPS. To suppress the border artefacts resulting from circular correlation, Galoogahi *et al.* [8] posed the DCF learning as a more complex optimization problem which can still be efficiently solved with the help of the Augmented Lagrangian Method (ALM). Lukezic *et al.* [7] further improved their idea by introducing spatial reliability maps to extract unpolluted foreground masks. In VOT 2017, DCF based algorithms constitute almost 50% of the submitted trackers [6] with ECO [9] and CSR-DCF [7] being among the top performers while CSR-DCF C++ implementation won the best real-time tracker award.

RGBD Trackers – as compared to generic, short-term tracking on RGB, RGBD tracking is a relatively unexplored area. This can be partly attributed to the lack of datasets with groundtruths until recently. Song *et al.* [18] captured and annotated a dataset consisting of 100 videos with an online evaluation system and their benchmark is still the largest available. They also provided multiple baseline algorithms under two main categories; *depth as an additional cue* and *point cloud tracking*. Depth as an additional cue trackers treat depth as an extra channel to HOG features [19] whereas point cloud tracking methods use 3D point clouds for generating 3D bounding boxes. Among the ten proposed variations the one with RGBD HOG features and boosted by optical flow and occlusion detector achieved the best performance.

The seminal work of Song *et al.* inspired many followups. Meshgi *et al.* [20] proposed an occlusion-aware particle filter based tracker that can handle persistent occlusions in a probabilistic manner. Bibi *et al.* [21] also used a particle filter framework with sparse parts for appearance modeling. In their model, each particle is a sparse, linear combination of 3D cuboids which stays fixed during the tracking. Without occlusion, they first make a coarse estimation of the target location using 2D optical flow and then sample particles over the rotation R and translation T spaces. Occlusion is detected by counting the number of points in the 3D cuboid representation. Success of the DCF approach naturally caught the attention in the RGBD community as well. To the best of our knowledge, the first DCF based RGBD tracker was proposed by Camplani *et al.* [22]. They first cluster the depth histogram and then apply a single Gaussian distribution to model the tracked object in the depth space. To extract the foreground object, they assume that the cluster with the smallest mean is the object. The second method using DCF was proposed by An *et al.* [23] where Kernelized Correlation Filters (KCF) are used in conjunction with depth based segmentation for target localization. Heuristic approaches were adopted for detecting whether an object is in the occlusion state or not.

Recently, Kart *et al.* [24] proposed an algorithm for using the depth as a means to generate masks for DCF updates. Although this work is in a similar spirit, our method differs from theirs in multiple, fundamental aspects; first of all, the authors incorporate neither color nor spatial cues for the mask creation. This results with the loss of very vital information sources. Especially in sequences where the target object and the occluding object have similar depth levels, it is very likely that the algorithm will not be able to discriminate in between even if they have different colors. Secondly, their foreground segmentation consists of a simple thresholding of depth probabilities which is an ad-hoc approach that requires careful fine tuning. Finally, the authors propose a brute force, full-frame grid search for recovering from the occlusion state whereas we propose to use the motion history of the target object to adaptively generate significantly smaller search areas to avoid redundant computational complexity.

3 RGBD Converter Framework

The proposed framework offers two levels of integration with the level-two being optional for trackers that can use a foreground mask in their model update. In the *level-one integration*, the framework continuously calculates the visibility state of a target object by casting the visibility problem as a pixel-wise foreground-background segmentation from multiple information sources: color, spatial proximity and depth. The segmentation result is the *foreground mask.* Without interfering with the internal structure of an RGB tracker, the framework uses the tracker output and bounding box to obtain a region of interest (ROI) for the segmentation step. If the ratio between the visible and occluded pixels is below a threshold, model updating of the RGB tracker is stopped and the framework goes into occlusion recovery mode. In the occlusion recovery mode, model update is stopped and the search region is continuously expanded around the last known location of the target. The search is performed by running the RGB tracker in a coarse-to-fine manner to find its maximum response r in the search region. The score is compared to the mean of last N valid responses (Section 3.4, Alg. 1). Once the target is detected, RGB tracker updating resumes. The *level-two integration* is available for trackers that use foreground masks in their model update.

For foreground segmentation, we adopt the energy minimization formulation in [25]:

$$E(f) = E_{smooth}(f) + E_{data}(f) \tag{1}$$

The goal is to find a pixel-wise labeling f (foreground/background) that minimizes the energy. E_{smooth} represents smoothness prior that penalizes neighboring pixels being labeled differently and E_{data} represents the observed data based energy. For E_{smooth}, we adopt the efficient computation of smoothed priors in [26] and E_{data} we formulate as

$$E_{data}(f) = E_{color}(f) + E_{spatial}(f) + E_{depth}(f) \tag{2}$$

where E_{color} measures the likelihood of observed pixel color given the target color model, E_{depth} models target region's depth and finally $E_{spatial}$ the spatial prior which is driven by the tracker location in the current frame. At the core of our approach are proper formulations of E_{color} (Section 3.1), E_{depth} (Section 3.2)

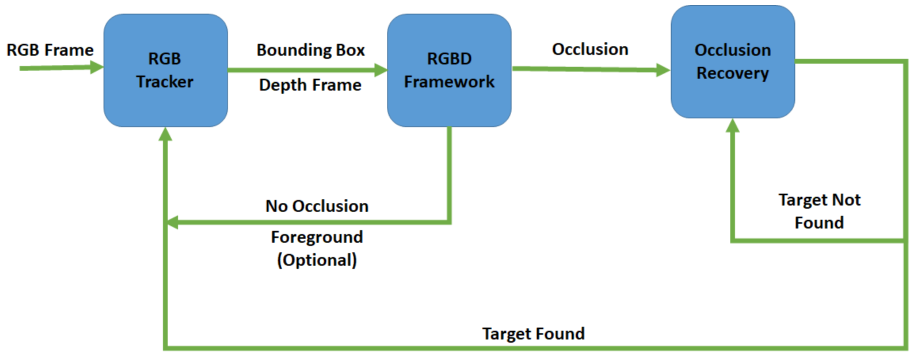

Fig. 2. The workflow diagram of the proposed framework. The framework uses bounding box coming from the RGB tracker and the depth frame to make a decision whether the target object is visible or not. In case it is visible, it allows the RGB tracker to update its model and continue tracking. If the target disappears, the framework runs the occlusion recovery module where the target object is searched using the last valid target model of the RGB tracker.

and $E_{spatial}$ (Section 3.3) so that the global optimum can be computed efficiently using the graph cuts algorithms [25, 27].

An example of the segmentation process is given in Fig. 3. As it can be seen, color based segmentation assigns both the target and the occluding object high confidence. However, the depth component is able to discriminate between the two while spatial component ensures high probability for the pixels that are close to the center of the tracking window.

3.1 Color-Based Target-Background Model E_{color}

In our formulation, E_{color} represents conditional dependencies between random variables (pixel fg/bg labels) for which we adopt a conditional random field formulation. The formulation uses the foreground/background probabilities as

$$E_{color} = \sum_{i \in \mathcal{V}} \psi_i(x_i) \tag{3}$$

where i is a graph vertex index (pixel) and x_i its corresponding label. ψ_i is encoded as a probability term

$$\begin{aligned} \psi_i(x_i = 0) &= -\log\left(p(x_i \notin fg)\right) \\ \psi_i(x_i = 1) &= -\log\left(p(x_i \in fg)\right) \end{aligned} \tag{4}$$

Since tracking is a temporal process, we need to add the frame number indicator to our notation $x_i \Rightarrow x_i^{(t)}$ where (t) is the current and $(t-1)$ the previous frame.

The probabilities $p(\cdot)$ can be efficiently computed using the color histograms of the foreground and background, h_f and h_b, respectively. It should be noted

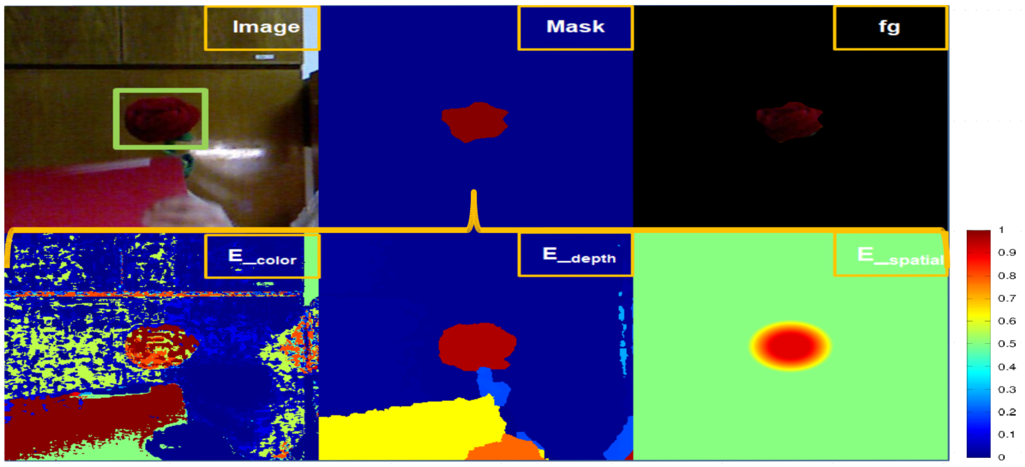

Fig. 3. Energy components in (2) and the segmentation output. The depth provides a strong cue even if the tracked and the occluding object have a very similar appearance.

that these histograms are updated after processing every frame for adapting to appearance changes. Therefore during processing frame t, the most recent color histograms are represented as h_f^{t-1} and h_b^{t-1}. Now, the color probability term is

$$p\left(x_i^t \in fg\right) = p\left(x_i^t = 1 \mid \text{hsv}(x_i^t), h_f^{(t-1)}, h_b^{(t-1)}\right) \quad . \tag{5}$$

where the $hsv(\cdot)$ function returns the HSV color space value of the pixel corresponding the label x_i in the current frame. The histograms are computed in 3D using $8 \times 8 \times 8 = 512$ uniformly distributed bins.

3.2 Depth-Based Target-Background Model E_{depth}

We model the depth induced energy E_{depth} similar to color using depth histograms $\hat{h}_f$ and $\hat{h}_b$

$$p\left(x_i^t \in fg\right) = p\left(x_i^t = 1 \mid \text{depth}(x_i^t), \hat{h}_f^{(t-1)}, \hat{h}_b^{(t-1)}\right) \tag{6}$$

where the depth probability is defined via the Bayesian rule (we use d to denote $\text{depth}(x_i^{(t)})$ for more compact representation)

$$p\left(x_i^{(t)} = 1 \mid d, \hat{h}_f^{(t-1)}, \hat{h}_b^{(t-1)}\right) = \frac{p\left(x_i^{(t)} = 1 \mid d, \hat{h}_f^{(t-1)}\right)}{p\left(x_i^{(t)} = 1 \mid d, \hat{h}_f^{(t-1)}\right) + p\left(x_i^{(t)} = 0 \mid d, \hat{h}_b^{(t-1)}\right)} \tag{7}$$

The above depth histograms are computationally efficient, but strongly biased against unseen depth levels. To be more precise, since probabilities for previously seen depth levels are high, the current frame (t) pixels with the same depth levels are more likely to be assigned to the foreground and the model easily fails to introduce new depth levels. For tackling this problem, we add foreground and background distribution priors in the spirit of Bayesian estimation theory. For the foreground histogram estimation prior, we adopt the triangle function which has a maximum at the foreground depth mode (d denotes depth(x_i) for more compact notation and $||\cdot||$ is the length of the histogram)

$$\Psi_f(d) = tri(d) = \left(1 - \frac{|d - \text{mode}(\hat{h}_f^{(t-1)})|}{||\hat{h}_f^{(t-1)}||}\right) * \gamma \tag{8}$$

and for the background histogram estimation, we adopt the uniform distribution as a non-informative prior

$$\Psi_b(d) = \text{unif}(x_i) = \frac{1}{||\hat{h}_b^{(t-1)}||} * \theta \tag{9}$$

γ and θ are constants that control the prior gains. The choice of using a triangle distribution for foreground and uniform distribution for background stems from the following; in case of the foreground depth levels, it is expected that the newly seen depth levels will be similar to the current depth (e.g. a rotating object) and depth values in general are concentrated around the mode/mean. However, we cannot make any assumptions about the background and therefore we adopt the non-informative prior in (9).

To ensure continuous depth levels while not compromising from quick updates, we propose to apply a smoothening filter $g^t(d)$ to the observed histogram in the updating stage where $g^t(d)$ is a single Gaussian function centered at the histogram mode. By suppressing depth values that are highly unlikely to belong to the current observation, it provides a safety mechanism against wrong detections and drifting. Thus, the depth histogram updating process takes the following online update form:

$$\begin{aligned} \hat{h}_f^{(t)} &= \alpha \hat{h}_f^{(t-1)} + \left((1-\alpha)\hat{h}_f^{(t)}\right) \odot g^t(d) \\ \hat{h}_b^{(t)} &= \alpha \hat{h}_b^{(t-1)} + \left((1-\alpha)\hat{h}_b^{(t)}\right) \end{aligned} \tag{10}$$

3.3 Spatial Prior $E_{spatial}$

The third energy term in our model is a spatial prior that gives preference to foreground labels near the object center suggested by the short-term tracker;

$$p\left(x_i^t \in fg\right) = p\left(x_i^t = 1 \mid \boldsymbol{x}(x_i^t)\right) = k\left(\boldsymbol{x}(x_i^t); \sigma\right) \tag{11}$$

where $\boldsymbol{x}(\cdot)$ provides the spatial location (x,y) of the label x_i and $k(x;\sigma)$ is a clipped Epanechnikov kernel commonly used in kernel density estimation.

3.4 Occlusion Recovery

Given the energy terms E_{color}, E_{depth} and $E_{spatial}$, graph cut [25] provides labeling of each pixel in the tracker window by minimizing the energy. If the number of foreground pixels falls below a threshold τ, the tracker is stopped and recovery process started. To this end, we propose to use the trained RGB model M^t as an object detector since the depth information is no longer reliable, especially when the occlusion is persistent.

The proposed recovery strategy is based on three principles: (i) target object will be found again near the spatial location where it was previously seen, (ii) tracker response of a recovered object must be similar (proportional by $\varOmega$) to the previous tracked frames ($N = 30$ in our experiments), and (iii) each region must be expanded with a speed proportional to the object's average speed before the object was lost. By expanding the search region adaptively, computational redundancy of processing irrelevant spatial regions is avoided. Algorithm 1 summarizes the occlusion recovery process.

Algorithm 1 Occlusion Recovery

```
Require: Current frame $I^t$, response threshold constant $\varOmega$ and target information
  before occlusion: $\{\boldsymbol{x}_i, \boldsymbol{bb}_i, r_i\}_{i=t-1,\ldots,t-10}$ (centroid, bounding box and response)
  Initialization: $n = 0$ {# of frames in recovery mode}
  Compute target speed $S = \max\left(5, \sum_{i=t-10}^{t-1} ||\boldsymbol{x}_i - \boldsymbol{x}_{i-1}||\right)$
  while max response $r^n < \varOmega * \mathrm{mean}(r_i)$ do
    Expand $W^n = n * S + 2 * \boldsymbol{bb}(1)$
    Expand $H^n = n * S + 2 * \boldsymbol{bb}(2)$
    Extract patch $I^{W^n \times H^n} \subset I^t$ centered at $\boldsymbol{x}_{t-1}$
    $n = n + 1$
    Find the tracker maximum response $r^n$ in $I^{W^n \times H^n}$
    Move to the next frame $t + 1$
  end while
  Reset depth histograms: $\hat{h}_f^{(t)}$ and $\hat{h}_b^{(t)}$
  Resume tracking with the short-term RGB tracker
```

3.5 Target-Background Mask Extension for CSR-DCF

This section is related to the *level-two integration* explained in the beginning of Section 3 and as the example case we use the CSR-DCF tracker in [7]. Since the original idea of Discriminative Correlation Filter (DCF) for tracking [28, 16], many improvements have been proposed. An efficient solution in the Fourier domain was proposed by Henriques *et al.* [29] and their work was followed by an important extension by Galoogahi *et al.* [8] who relaxed the assumption of circular symmetrical filters. These extensions were adopted in CSR-DCF [7] which constructs a reliability mask that is used to mask out background regions during tracker updates. Intuitively, the CSR mask can be replaced with the proposed foreground mask which is the output of graph cuts optimization (see Figure 3 for an example mask). In our experiments, it turns out that this significantly improves the performance of CSR-DCF since the proposed depth-based mask avoids model pollution more effectively. The level-two integration of our framework to CSR-DCF is simple: CSR mask is replaced with the mask produced by minimizing (1).

4 Experiments

In this section, we present the results for various trackers augmented with the proposed framework. Four generic, short-term trackers due to their proven success and efficiency are chosen; DCF [8], ECO [9] and CSR-DCF [7]. Since ECO has two variants, we applied the proposed framework to both ECO-gpu (deep features) and ECO-hc (hand crafted features).

4.1 Experimental Setup

Implementation Details – All experiments were run on a single laptop using a non-optimized Matlab code with Intel Core i7 3.6GHz and Ubuntu 16.04 OS. The parameters for the proposed algorithm were empirically set and kept constant during the experiments. Tracking parameters were as in the original works with the exception of DCF and CSR-DCF filter learning update rates were set to 0.03 and the number of bins for color histograms to 512. The rest of the parameters can be found in the publicly available code of our framework. [32]
Dataset – For validating the proposed framework we conducted experiments on the Princeton Tracking Benchmark (PTB) [18]. The dataset consists of 95 evaluation sequences and 5 validation sequences from 11 tracking categories, namely *human*, *animal*, *rigid*, *large*, *small*, *slow*, *fast*, *occlusion*, *no occlusion*, *passive motion* and *active motion*. The videos have been recorded with a standard Kinect 1.0 device and all frames annotated manually.
Evaluation Metrics – We use the metrics as they are provided by PTB [18].

However, the evaluation sequences do not contain publicly available ground truths except for the initial frame. To facilitate a fair comparison, Song *et al.* [18] also provide an online system where the resulting coordinates are uploaded for obtaining the final scores and ranking. The results of other methods in our paper were taken from the online system's website with the exception of DLST [23] who have not registered their methods. DLST scores were obtained from its paper. Bibi *et al.* [21] depth images are adopted in the experiments.

4.2 Comparison to State-of-the-Art

The results of the converted short-term trackers and the other top performing trackers on the PTB dataset are given in Table 1. Since the evaluation server did not allow multiple simultaneous submissions, we submitted each method

Table 1. Comparison of short-term RGB and RGBD tracking methods on the Princeton Tracking Benchmark (PTB) [18]. DCF [8] and three state-of-the-art trackers were used within the framework – ECOgpu [9], ECOhc [9] and CSR-DCF [7]; their level-one RGBD extensions are denoted DCF-rgbd, ECO-rgbd and CSR-DCF-rgbd, the level-two CSR-DCF integration where the original RGB-based mask is replaced by the proposed foreground mask (Section 3.5) is denoted CSR-DCF-rgbd++. (The table shows results for the Princeton Benchmark as of June 15, 2018)

	Avg Rank	**Tracking Category**										
Method		*Human*	*Animal*	*Rigid*	*Large*	*Small*	*Slow*	*Fast*	*Occ.*	*No-Occ.*	*Passive*	*Active*
⋆*CSR-DCF-rgbd++*	**3.64**	0.77(2)	0.65(5)	0.76(6)	0.75(4)	**0.73(1)**	0.80(3)	0.72(3)	0.70(3)	0.79(5)	0.79(5)	0.72(3)
OAPF [20]	5.27	0.64(14)	**0.85(1)**	0.77(4)	0.73(6)	0.73(2)	**0.85(1)**	0.68(8)	0.64(8)	**0.85(1)**	0.78(9)	0.71(4)
3D-T [21]	5.36	**0.81(1)**	0.64(7)	0.73(15)	**0.80(1)**	0.71(6)	0.75(8)	**0.75(1)**	**0.73(1)**	0.78(11)	0.79(6)	0.73(2)
RGBDOcc+OF [18]	5.55	0.74(5)	0.63(9)	0.78(2)	0.78(3)	0.70(7)	0.76(5)	0.72(4)	0.72(2)	0.75(17)	0.82(2)	0.70(5)
∘*ECOhc-rgbd*	6.18	0.70(7)	0.55(15)	**0.81(1)**	0.69(9)	0.72(4)	0.78(4)	0.68(7)	0.65(6)	0.79(6)	**0.83(1)**	0.66(8)
DSKCF-Shape [30]	6.64	0.71(6)	0.71(3)	0.74(11)	0.74(5)	0.70(8)	0.76(6)	0.70(6)	0.65(7)	0.81(4)	0.77(11)	0.70(6)
DLST [23]	6.73	0.77(3)	0.69(4)	0.73(16)	0.80(2)	0.70(9)	0.73(14)	0.74(2)	0.66(5)	0.85(2)	0.72(16)	**0.75(1)**
DM-DCF [24]	6.73	0.76(4)	0.58(12)	0.77(5)	0.72(8)	0.73(3)	0.75(10)	0.72(5)	0.69(4)	0.78(13)	0.82(3)	0.69(7)
DSKCF [22]	9.36	0.67(10)	0.61(10)	0.76(8)	0.69(10)	0.70(10)	0.75(9)	0.67(9)	0.63(9)	0.78(12)	0.79(7)	0.66(9)
⋄*ECOgpu-rgbd*	9.82	0.66(11)	0.58(11)	0.76(7)	0.65(14)	0.71(5)	0.81(2)	0.64(14)	0.62(10)	0.77(14)	0.78(8)	0.65(12)
DSKCF-CPP [22]	10.36	0.65(12)	0.64(8)	0.74(12)	0.66(13)	0.69(12)	0.76(7)	0.65(13)	0.60(12)	0.79(7)	0.80(4)	0.64(14)
RGBD+OF [18]	11.36	0.64(15)	0.65(6)	0.75(9)	0.72(7)	0.65(17)	0.73(15)	0.66(10)	0.60(13)	0.79(8)	0.74(15)	0.66(10)
hiob [31]	11.64	0.53(19)	0.72(2)	0.78(3)	0.61(16)	0.70(11)	0.72(16)	0.64(15)	0.53(16)	0.85(3)	0.77(12)	0.62(15)
⋆*CSR-DCF-rgbd*	11.91	0.68(9)	0.57(13)	0.74(10)	0.68(11)	0.68(14)	0.74(12)	0.65(12)	0.62(11)	0.75(16)	0.77(10)	0.64(13)
∘*ECOhc* [9]	12.18	0.69(8)	0.56(14)	0.72(17)	0.67(12)	0.68(13)	0.74(11)	0.65(11)	0.59(14)	0.78(9)	0.74(14)	0.65(11)
⋄*ECOgpu* [9]	15.36	0.58(16)	0.54(16)	0.73(13)	0.59(18)	0.65(15)	0.73(13)	0.58(17)	0.51(17)	0.78(10)	0.69(17)	0.60(17)
•*DCF-rgbd*	15.45	0.64(13)	0.54(17)	0.73(14)	0.65(15)	0.65(16)	0.71(17)	0.63(16)	0.59(15)	0.74(18)	0.76(13)	0.61(16)
•*DCF* [8]	18.09	0.56(17)	0.52(19)	0.66(18)	0.60(17)	0.59(19)	0.65(18)	0.57(18)	0.48(18)	0.74(19)	0.68(18)	0.56(18)
⋆*CSR-DCF* [7]	18.36	0.54(18)	0.53(18)	0.64(19)	0.56(19)	0.59(18)	0.61(19)	0.56(19)	0.44(19)	0.76(15)	0.64(19)	0.55(19)

separately and generated the leaderboard using the official protocol; methods were first ranked in each category and then the average rank was calculated. The symbols •, ⋆, ⋄, and ∘ in Table 1 mark the trackers that our framework applied to. As it can be observed, the proposed method clearly has a big impact on overall rankings for all three trackers. Especially in sequences with occlusions, this impact becomes more visible. CSR-DCF improves 8 ranks, ECOgpu ranking sees 7 ranks improvement and ECOhc rank improves by 8. In terms of accuracy, the improvement is as strong as in rankings. When the proposed framework (without foreground masked updates) is applied to CSR-DCF, its performance in occlusion sequences increases 18% while ECOhc grows by 6% and ECOgpu 11%. The level-two integration further boosts occlusion sequences accuracy for CSR-DCF to a total of 26%.

Unlike other top performing methods, CSR-DCF-rgbd++ also maintains a well-balanced performance over all the categories by staying among the top in every one. This suggests that it does not overfit to specific categories but it provides similar performance for different scenarios which makes it a very suitable candidate for real-life applications.

Fig. 4 shows that the proposed framework adds to tracker's occlusion resilience to both short-term and long-term occlusions. For example, ECOhc-rgbd was able to detect the occlusion and it also re-detected the target object when it reappeared in the scene instead of drifting due to model pollution. As a good

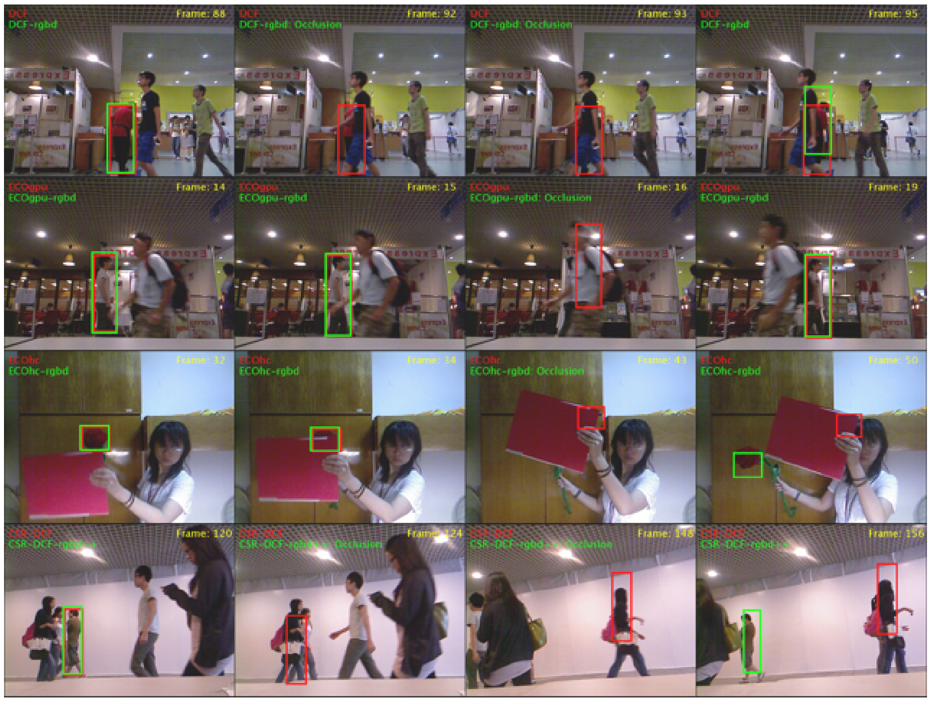

Fig. 4. Short-term and long-term occlusion examples comparing the original methods(red) and their RGBD versions(green). Top row – DCF, second row – ECOgpu, third row – ECOhc, bottom row – CSR-DCF.

example of long-term recovery example, CSR-DCF-rgbd++ was able to recover even after 35 frames of occlusion state since it avoided model corruption and expanded the search region gradually.

The reason why CSR-DCF-rgbd++ performs better than the other RGBD methods can be possibly explained by its masked DCF update mechanism which uses the foreground provided by the framework. In the discriminative tracking paradigm, the tracker's target model is updated over the time for coping with the visual changes. However, when a rectangular bounding box is used for this purpose, it is likely to include background and occluding object's pixels as well. This results with learning of irrelevant information that may cause drifting. Whereas in the masked update approach, the updates are done only using the pixels that are confidently belong to the target object. Thus, the target model stays uncorrupted which results with better performance.

5 Conclusions

A generic framework was proposed for converting existing short-term RGB trackers into RGBD trackers. The framework is easy to adopt as it only requires control of model updating (stop/resume) and a tracking bounding box which are both provided in any tracker that is VOT compliant [6]. At the core of the framework is a foreground model which uses depth, color and spatial cues to efficiently detect occluded regions which are utilized at two-levels: occlusion detection and optionally, masked tracker updates. In all experiments, existing RGB trackers improved their ranks in the publicly available Princeton tracking benchmark [18]. CSR-DCF tracker which allows level-two integration of the proposed foreground model achieved state-of-the-art accuracy and was ranked the best on the day of submission. The full source code of the framework is publicly available. [32]

Acknowledgements

Uğur Kart was supported by two projects: Business Finland Project ”360 Video Intelligence - For Research Benefit” with Nokia, Lynx, JJ-Net, BigHill, Leonidas and Business Finland - FiDiPro Project ”Pocket - Sized Big Visual Data”. Jiří Matas was supported by the OP VVV MEYS project CZ.02.1.01/0.0/0.0/16_019/0000765 ”Research Center for Informatics”.

References

1. Wu, Y., Lim, J., Yang, M.H.: Online Object Tracking: A Benchmark. In: CVPR. (2013)
2. Kristan, M., Pflugfelder, R., Leonardis, A., Matas, J., Porikli, F., et al.: The Visual Object Tracking VOT2013 Challenge Results. In: CVPR Workshops. (2013)
3. Kristan, M., Pflugfelder, R., Leonardis, A., Matas, J., Cehovin, L., Nebehay, G., Vojir, T., et al.: The Visual Object Tracking VOT2014 Challenge Results. In: ECCV Workshops. (2014)
4. Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Cehovin, L., et al.: The Visual Object Tracking VOT2015 challenge results. In: ICCV Workshops. (2015)
5. Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., Cehovin, L., Vojir, T., et al.: The Visual Object Tracking VOT2016 Challenge Results. In: ECCV Workshops. (2016)
6. Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., et al.: The Visual Object Tracking VOT2017 Challenge Results. In: ICCV Workshops (2017)
7. Lukezic, A., Vojir, T., Cehovin, L., Matas, J., Kristan, M.: Discriminative Correlation Filter with Channel and Spatial Reliability. In: CVPR. (2017)
8. Galoogahi, H., Sim, T., Lucey, S.: Correlation Filters with Limited Boundaries. In: CVPR. (2015)
9. Danelljan, M., Bhat, G., Shahbaz Khan, F., Felsberg, M.: ECO: Efficient Convolution Operators for Tracking. In: CVPR. (2017)
10. Ross, D.A., Lim, J., Lin, R.S., Yang, M.H.: Incremental Visual Tracking. International Journal of Computer Vision (IJCV) **77** (2008) 125–141
11. Zhang, T., Liu, S., Xu, C., Yan, S., Ghanem, B., Ahuja, N., Yang, M.H.: Structural Sparse Tracking. In: CVPR. (2015)
12. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-Based Object Tracking. IEEE PAMI **25** (2003) 564–567
13. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-Learning-Detection. In: IEEE PAMI. Volume 34. (2011) 1409–1422
14. Danelljan, M., Robinson, A., Shahbaz Khan, F., Felsberg, M.: Beyond Correlation Filters: Learning Continuous Convolution Operators for Visual Tracking. In: ECCV. (2016)
15. Nam, H., Han, B.: Learning Multi-Domain Convolutional Neural Networks for Visual Tracking. In: CVPR. (2016)
16. Bolme, D.S., Beveridge, J., Draper, B.A., Lui, Y.M.: Visual Object Tracking using Adaptive Correlation Filters. In: CVPR. (2010)
17. Henriques, J., Caseiro, R., Martins, P., Batista, J.: High-Speed Tracking with Kernelized Correlation Filters. In: IEEE PAMI. Volume 37. (2014) 1–14
18. Song, S., Xiao, J.: Tracking Revisited Using RGBD Camera: Unified Benchmark and Baselines. In: ICCV. (2013)
19. Dalal, N., Triggs, B.: Histograms of Oriented Gradients for Human Detection. In: CVPR. (2005)
20. Meshgi, K., Maeda, S.I., Oba, S., Skibbe, H., Li, Y.Z., Ishii, S.: An Occlusion-Aware Particle Filter Tracker to Handle Complex and Persistent Occlusions. CVIU **150** (2016) 81–94
21. Bibi, A., Zhang, T., Ghanem, B.: 3D Part-Based Sparse Tracker with Automatic Synchronization and Registration. In: CVPR. (2016)
22. Camplani, M., Hannuna, S., Mirmehdi, M., Damen, D., Paiement, A., Tao, L., Burghardt, T.: Real-time RGB-D Tracking with Depth Scaling Kernelised Correlation Filters and Occlusion Handling. In: BMVC. (2015)

23. An, N., Zhao, X.G., Hou, Z.G.: Online RGB-D Tracking via Detection-Learning-Segmentation. In: ICPR. (2016)
24. Kart, U., Kämäräinen, J.K., Matas, J., Fan, L., Cricri, F.: Depth Masked Discriminative Correlation Filter. In: ICPR. (2018)
25. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE PAMI **23**(11) (2001)
26. Diplaros, A., Vlassis, N., Gevers, T.: A spatially constrained generative model and an EM algorithm for image segmentation. IEEE Trans. on Neural Networks **18**(3) (2007)
27. Rother, C., Kolmogorov, V., Blake, A.: GrabCut interactive foreground extraction using iterated graph cuts. In: SIGGRAPH. (2004)
28. Hester, C., Casasent, D.: Multivariant technique for multiclass pattern recognition. In: Applied Optics. Volume 19. (1980) 1758–1761
29. Henriques, J., Caseiro, R., Martins, P., Batista, J.: Exploiting the circulant structure of tracking-by-detection with kernels. In: ECCV. (2012)
30. Hannuna, S., Camplani, M., Hall, J., Mirmehdi, M., Damen, D., Burghardt, T., Paiement, A., Tao, L.: DS-KCF: A Real-Time Tracker for RGB-D Data. In: Journal of Real-Time Image Processing. (2016)
31. Springstübe, P., Heinrich, S., Wermter, S.: Continuous Convolutional Object Tracking. In: ESANN. (2018)
32. https://github.com/ugurkart/rgbdconverter