Kart U, Lukezic A, Kristan M, Kamarainen JK, Matas J. Object Tracking by Reconstruction with View-Specific Discriminative Correlation Filters. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2019 (pp. 1339-1348).

Object Tracking by Reconstruction with View-Specific Discriminative Correlation Filters

Uğur Kart*, Alan Lukežič†, Matej Kristan†, Joni-Kristian Kämäräinen*,Jiří Matas‡
*Laboratory of Signal Processing, Tampere University, Finland
† Faculty of Computer and Information Science, University of Ljubljana, Slovenia
‡ Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic
{ugur.kart,joni.kamarainen}@tuni.fi
{alan.lukezic,matej.kristan}@fri.uni-lj.si
matas@cmp.felk.cvut.cz

Abstract

Standard RGB-D trackers treat the target as a 2D structure, which makes modelling appearance changes related even to out-of-plane rotation challenging. This limitation is addressed by the proposed long-term RGB-D tracker called OTR – Object Tracking by Reconstruction. OTR performs online 3D target reconstruction to facilitate robust learning of a set of view-specific discriminative correlation filters (DCFs). The 3D reconstruction supports two performance-enhancing features: (i) generation of an accurate spatial support for constrained DCF learning from its 2D projection and (ii) point-cloud based estimation of 3D pose change for selection and storage of view-specific DCFs which robustly localize the target after out-of-view rotation or heavy occlusion. Extensive evaluation on the Princeton RGB-D tracking and STC Benchmarks shows OTR outperforms the state-of-the-art by a large margin.

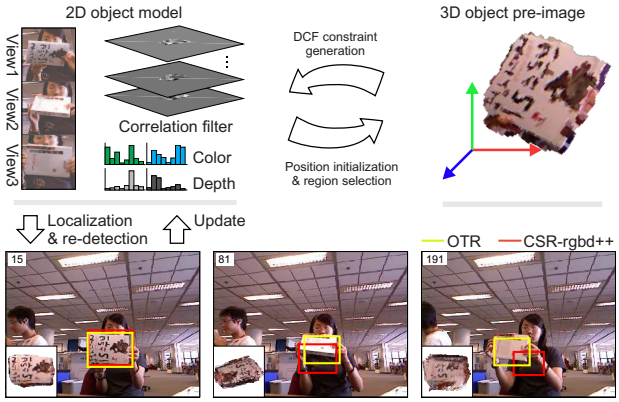

Figure 1. The OTR – Object Tracking by Reconstruction – object model consists of a set of 2D view-specific DCFs and of an approximate 3D object reconstruction. The OTR thus copes well with out-of-view rotation with a significant aspect change, while a state-of-the-art tracker CSR-rgbd++ [19] drifts and fails.

1. Introduction

Visual object tracking (VOT) is one of the core problems in computer vision; it has many applications [18, 8]. The field has progressed rapidly, fueled by the availability of large and diverse datasets [39, 34] and the annual VOT challenge [22, 23]. Until recently, tracking research has focused on RGB videos, largely neglecting RGB-D (rgb+depth) tracking as obtaining a reliable depth map at video frame rates has not been possible without expensive hardware. In the last few years, depth sensors have become widely accessible, which has lead to a significant increase of RGB-D tracking related work [6, 1, 26]. Depth provides important cues for tracking since it simplifies reasoning about occlusions and facilitates depth-based object segmentation. Progress in RGB-D tracking has been further boosted by the emergence of standard datasets and evaluation protocols [35, 40].

In RGB-D tracking, direct extensions of RGB methods by adding the D-channel as an additional input dimension have achieved considerable success. In particular, discriminative correlation filter (DCF) based methods have shown excellent performance on the Princeton RGB-D tracking benchmark [35], confirming the reputation gained on RGB benchmarks [22, 23, 19, 20, 6, 1]. Furthermore, DCFs are efficient in both learning of the visual target appearance model and in target localization, which are both implemented by FFT, running in near real time on a standard CPU.

A major limitation of the standard RGB and RGB-D trackers, regardless of the actual method (e.g. DCF [4], Siamese deep nets [2], Mean shift [9], Lucas Kanade [27]),

is that they treat the tracked 3D object as a 2D structure. Thus even a simple rotation of a rigid 3D object is interpreted as potentially significant appearance change in 2D that is conceptually indistinguishable from partial occlusion, tracker drift, blurring and ambient light changes.

Consider a narrow object, e.g., a book, with its front cover facing the camera, that rotates sideways and ends with its back side facing the camera (Figure 1). From the perspective of a standard RGB tracker, the object has deformed and the appearance has completely changed. Since most of the standard trackers cannot detect (do not model) aspect changes, the target bounding box and the appearance model contain mostly pixels belonging to the background when the narrow side of the book is facing the camera. Furthermore, the model update is carried out by implicit or explicit temporal averaging of the tracked views. Consequently, the appearance observed in the earlier frames is lost after a certain time period, limiting re-detection capability in situation when the target is completely occluded, but later re-appears, since its appearance no longer matches the last observed view. The above-mentioned problems are almost trivial to solve if a 3D model with attached photometric information is available for the tracked object.

We exploit the opportunity of using the depth component in RGB-D signal to build a simple, yet powerful 3D object representation based on the surface splat model, i.e., the object surface is approximated by a collection of 3D points with color, radius and the normal – *surfels*. This model has been proven very powerful in the context of SLAM [33]. The 3D model is aligned and updated to the current 2D target appearance during tracking by an ICP-based matching mechanism [33] – thus a *pre-image* of the 2D target projection is maintained during tracking. The 3D object pre-image significantly simplifies detection and handling of (self-)occlusion, out-of-plane rotation (view changes) and aspect changes.

The ICP-based 3D pre-image construction [33] requires accurate identification of the object pixels in the current frame prior to matching, and it copes with only small motions due to a limited convergence range. A method from a high-performance RGB-D DCF tracker [19] is thus used to robustly estimate potentially large motions and to identify object pixels for the pre-image construction. The DCF learning is improved by generating appearance constraints from the pre-image. Object appearance changes resulting from out-of-view rotation are detected by observing the pre-image 3D motion and a set of view-specific DCFs is generated. These 2D models are used during tracking for improved localization accuracy as well as for target re-detection using the recent efficient formulation of the DCF-based detectors [30]. The resulting tracker thus exhibits a long-term capability, even if the target re-appears in a pose different from the one observed before the occlusion.

Contributions The main contribution of the paper is a new long-term RGB-D tracker, called OTR – Object Tracking by Reconstruction that constructs a 3D model with view-specific DCFs attached. The DCF-coupled estimation of the object pre-image and its use in DCF model learning for robust localization has not been proposed before. The OTR tracker achieves the *state-of-the-art*, outperforming prior trackers by a large margin on two standard RGB-D tracking benchmarks. An ablation study confirms the importance of view-specific DCF appearance learning that is tightly connected to the 3D reconstruction. We will make the reference implementation of OTR available at https://github.com/ugurkart.

2. Related Work

RGB Tracking Of the many approaches proposed in the literature, DCF-based methods have demonstrated excellent performance – efficiency trade-off in recent tracking challenges [24, 22, 23]. Initially proposed by Bolme *et al*. [4], DCF-based tracking captured the attention of the vision community due to its simplicity and mathematical elegance. Improvements of the original method include multi-channel formulation of correlation filters [12, 15], filter learning using kernels exploiting properties of circular correlation [17] and scale estimation with multiple one-dimensional filters [11]. Following these developments, Galoogahi *et al*. [14] tackled the boundary problems that stem from the nature of circular correlation by proposing a filter learning method where a filter with size smaller than the training example is adopted. Lukezic *et al*. [28] further improved this idea by formulating the filter learning process using a graph cut based segmentation mask as a constraint.

RGB-D Tracking The most extensive RGB-D object tracking benchmark has been proposed by Song *et al*. [35] (Princeton Tracking Benchmark). The benchmark includes a dataset, evaluation protocol and a set of baseline RGB-D trackers. Several RGB-D trackers have been proposed since. Meshgi *et al*. [31] used an occlusion-aware particle filter framework. A similar approach was proposed by Bibi *et al*. [3] but using optical flow to improve localization accuracy. As an early adopter of DCF based RGB-D trackers, Hannuna *et al*. [16] used depth as a clue to detect occlusions while tracking is achieved by KCF [17]. An *et al*. [1] performed a depth based segmentation along with a KCF tracker. Kart *et al*. [20] proposed a purely depth based segmentation to train a constrained DCF similarly to CSR-DCF [28] and later extended their work to include color in segmentation [19]. Liu *et al*. [26] proposed a context-aware 3-D mean-shift tracker with occlusion handling. At the time of writing this paper [26] is ranked first at Princeton Tracking Benchmark. Xiao *et al*. [40] recently proposed a new RGB-D tracking dataset (STC) and an RGB-D tracker by

adopting an adaptive range-invariant target model.

3D Tracking Klein *et al*. [21] proposed a camera pose tracking algorithm for small workspaces which works on low-power devices. The approach is based on tracking keypoints across the RGB frames and bundle adjustment for joint estimation of the 3D map and camera pose. Newcombe *et al*. [32] proposed an iterative closest point (ICP) based algorithm for depth sequences for dense mapping of indoor scenes. In a similar fashion, Wheelan *et al*. [38] used surfel-based maps and jointly optimized color and geometric costs in a dense simultaneous localization and mapping (SLAM) framework. All three methods are limited to static scenes and are inappropriate for object tracking. This limitation was addressed by Rünz *et al*. [33], who extended [38] by adding the capability of segmenting the scene into multiple objects. They use a motion consistency and semantic information to separate the object from the background. This limits the method to large, slow moving objects.

Lebeda *et al*. [25] combined structure from motion, SLAM and 2D tracking to cope with 3D object rotation. Their approach reconstructs the target by tracking keypoints and line features, however, it cannot cope with poorly-textured targets and low-resolution images.

3. Object tracking by 3D reconstruction

In OTR, object appearance is modeled at two levels of abstraction which enables per-frame target localization and re-detection in the case of tracking failure. The appearance level used for localizing the target in the image is modelled by a a set of view-specific discriminative correlation filters, i.e., a DCF $\mathbf{h}_t$ that models the current object appearance, and a set of snapshots $\{\mathbf{h}^{(s)}\}_{s=1}^{S}$ modelling the object from previously observed views. In addition to the filters, the object color and depth statistics are modelled by separate color and depth histograms for the foreground and the background.

The second level of object abstraction is a model of the object pre-image $\Theta_t = \{\mathbf{P}_t, \mathbf{R}_t, \mathbf{T}_t\}$, where $\mathbf{P}_t$ is the surfel-based object 3D model specified in the object-centered coordinate system and $\{\mathbf{R}_t, \mathbf{T}_t\}$ are the rotation and the translation of the 3D model into the current object position.

The two models interact during tracking for improved DCF training and 3D pose change detection (e.g., rotations). We describe the DCF framework used by the OTR tracker in Section 3.1, the multi-view DCFs with the pre-image model is detailed in Section 3.2, Section 3.3 details target loss recovery and Section 3.4 summarizes the full per-frame tracking iteration.

3.1. Constrained DCF

The core DCF tracker in the OTR framework is the recently proposed constrained discriminative correlation filter CSR-DCF [29], which is briefly outlined here. Given a search region of size $W \times H$ a set of N_d feature channels $\mathbf{f} = \{\mathbf{f}_d\}_{d=1}^{N_d}$, where $\mathbf{f}_d \in \mathcal{R}^{W \times H}$, are extracted. A set of N_d correlation filters $\mathbf{h} = \{\mathbf{h}_d\}_{d=1}^{N_d}$, where $\mathbf{h}_d \in \mathcal{R}^{W \times H}$, are correlated with the extracted features and the object position is estimated as the location of the maximum of the weighted correlation responses

$$\mathbf{r} = \sum_{d=1}^{N_d} w_d (\mathbf{f}_d \star \mathbf{h}_d), \tag{1}$$

where $\star$ represents circular correlation, which is efficiently implemented by a Fast Fourier Transform with $\{w_d\}_{d=1}^{N_d}$ being the channel weights. The target scale can be efficiently estimated by another correlation filter trained over the scale-space [11].

Filter learning is formulated in CSR-DCF as a constrained optimization that minimizes a regression loss

$$\varepsilon(\mathbf{h}) = \sum_{d=1}^{N_c} \|\mathbf{f}_d \star \mathbf{h}_d - \mathbf{g}\|^2 + \lambda \|\mathbf{h}_d\|^2 \; ; \; \mathbf{h}_d \equiv \mathbf{m} \odot \mathbf{h}_d, \tag{2}$$

where $\mathbf{g}$ is a desired output and $\mathbf{m}$ is a binary mask $\mathbf{m} \in \{0,1\}^{W \times H}$ that approximately separates the target from the background. The mask thus acts as a constraint on the filter support, which allows learning a filter from a larger training region as well as coping with targets that are poorly approximated by an axis-aligned bounding box. CSR-DCF applies a color histogram-based segmentation for mask generation, which is not robust to visually similar backgrounds and illumination change. We propose generating the mask from the RGB-D input and the estimated pre-image in Section 3.2.1.

Minimization of (2) is achieved by an efficient ADMM scheme [5]. Since the support of the learned filter is constrained to be smaller than the learning region, the maximum response on the training region reflects the reliability of the learned filter [28]. These values are used as per-channel weights w_d in (1) for improved target localization (we refer the reader to [29] for more details).

3.2. A multi-view object model

At each frame, the current filter $\mathbf{h}_t$ is correlated within a search region centered on the target position predicted from the previous frame following (1). To improve localization during target 3D motion, we introduce a "memory" which is implemented by storing captured snapshots $\{\mathbf{h}^{(s)}\}_{s=1}^{S}$ from different 3D view-points (i.e., a set of view-specific DCFs). At every N_{R}-th frame, all view-specific DCFs are evaluated, and the location of the maximum of the correlation response is used as the new target hypothesis $\mathbf{x}_t$. If the maximum correlation occurs in the set of snapshots, the current

filter is replaced by the corresponding snapshot filter. Target presence is determined at this location by the test described in Section 3.3.1. In the case the test determines target is lost, the tracker enters a re-detection stage described in Section 3.3.

If the target is determined to be present, the current filter $\mathbf{h}_t$ is updated by a weighted running average

$$\mathbf{h}_{t+1} = (1-\eta)\mathbf{h}_t + \eta\tilde{\mathbf{h}}_t, \tag{3}$$

where $\tilde{\mathbf{h}}_t$ is a new filter estimated by the constrained filter learning in Section 3.1 at the estimated position $\mathbf{x}_t$ and η is the update factor.

In addition to updating the current filter, the object color and depth histograms are updated as in [19], the object pre-image is updated as described in Section 3.2.2 and the set of view-specific DCFs $\{\mathbf{h}^{(s)}\}_{s=1}^{S}$ is updated following Section 3.2.3.

3.2.1 Object pre-image-based filter constraint

The binary mask $\mathbf{m}$ used in the constrained learning in (2) is computed at the current target position at filter learning stage. In the absence of other inputs, the mask is estimated by a recent segmentation approach from [19]. This approach uses an MRF segmentation model from CSR-DCF [29] within the filter learning region and estimates per-pixel unary potentials by color and depth (foreground/background) histograms backprojection in the RGB-D image.

However, the pre-image Θ_t can be used to better outline the object in the filter training region, leading to a more accurately learned filter. Thus, at DCF training stage, the pre-image is generated by fitting the object 3D model $\mathbf{P}_t$ onto the current object appearance (Section 3.2.2). If the fit is successful, the segmentation mask used in filter learning (2) is replaced by a new mask generated as follows. The 3D model $\mathbf{P}_t$ is projected into the 2D filter training region. Pixels in the region corresponding to the visible 3D points are set to one, while others to zero, thus forming a binary object occupancy map. The map is dilated to remove holes in the object mask and only the largest connected component is retained, while others are set to zero to reduce the effect of potential reconstruction errors in the 3D model. An example of the 2D mask construction from the 3D object pre-image is demonstrated in Figure 2.

3.2.2 Object pre-image update

The object pre-image Θ_t is updated from the object position estimated by the multi-view DCF (Section 3.2). Pixels corresponding to the target are identified by the color+depth segmentation mask from Section 3.2.1. The patch is extracted from the RGB-D image and used to update the object 3D model $\mathbf{P}_t$. The 3D model $\mathbf{P}_t$ is first translated to

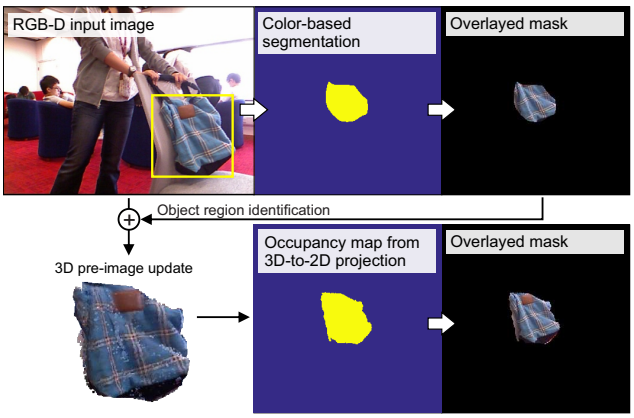

Figure 2. A 2D DCF localizes the target (top-left), the target color+depth pixels are approximately segmented (top-right) and used to update the 3D pre-image (bottom-left). The pre-image is projected to 2D generating an occupancy map (bottom-right). The resulting mask better delineates the object, which improves the constrained DCF learning.

the 3D position determined by the target location from the multi-view DCF. The ICP-based fusion from [33], that uses color and depth, is then applied to fine align the 3D model with the patch and update it by adding and merging the corresponding surfels (for details we refer to [33]). The updated model is only retained if the ICP alignment error is reasonably low (i.e., below a threshold τ_{ICP}), otherwise the update is discarded.

3.2.3 A multi-view DCF update

Continuous updates may lead to gradual drift and failure whenever the target object undergoes a significant appearance change. Recovery from such situations essentially depends on the diversity of the target views captured by the snapshots $\{\mathbf{h}^{(s)}\}_{s=1}^{S}$ and their quality (e.g., snapshots should not be contaminated by the background). The following conservative update mechanism that maximizes snapshot diversity and minimizes contamination is applied.

The current filter is considered for addition to the snapshots only if the target passed the presence test (Section 3.3.1) and the object pre-image Θ_t is successfully updated (Section 3.2.2). Passing these two tests, the target is considered visible with the pre-image accurately fitted. A filter is added if the object view has changed substantially and results in a new appearance (viewpoint). The change is measured by a difference between the reference aspect ρ_0 (i.e., a bounding box width-to-height ratio) and the aspect ρ_t obtained from the current 2D projection of the object pre-image. Whenever this difference exceeds a threshold, i.e., $\|\rho_0 - \rho_t\| > \tau_\rho$, a new snapshot is created and the current ratio becomes a new reference, i.e., $\rho_0 \leftarrow \rho_t$. In our preliminary experiments, we tested using Euler angles of the estimated rotation matrix $\mathbf{R}$, but this was found sensitive to

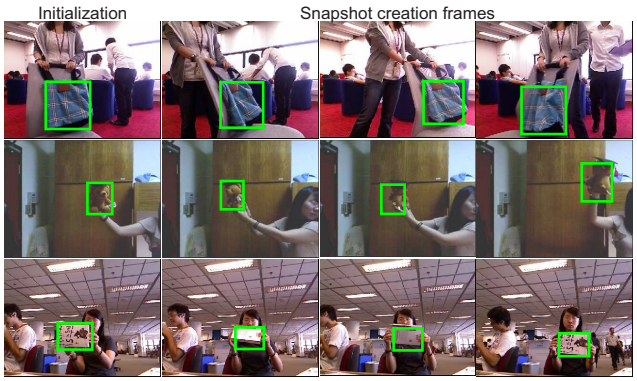

Figure 3. Examples of view-specific DCFs creation. The tracker was initialized on the images in the left-most column, while the remaining images represent frames in which a new view was detected and stored in the set of view-specific DCFs.

ICP estimation errors and therefore aspect ratio test proved to be more robust. Examples of images used to create separate DCF views are shown in Figure 3.

3.3. A multi-view DCF target detection

Target presence is determined at each frame using the test described in Section 3.3.1. Whenever the target is lost, the following re-detection mechanism is activated. At each frame all filters in the snapshot set $\{\mathbf{h}^{(s)}\}_{s=1}^{S}$ are correlated with features extracted from a region centered at the last confident target position. To encode a motion model, the search region size is gradually increased in subsequent frames by a factor $\alpha_s^{\Delta t}$, where $\alpha_s > 1$ is a fixed scale factor and Δt is the number of frames since the last confident target position estimation. The correlation is efficiently calculated by padding the snapshots with zeros to the current search region size and applying FFT [30].

Since the target may change the size, a two-stage approach for re-detection is applied. First, the hypothesized target position is estimated as the location of the maximum correlation response and the filter $\mathbf{h}^{(m)}$ that yielded this response is identified. The current object scale is then computed as the ratio $s_f = \frac{D_0}{D_t}$ between the depth of the target in the first frame (D_0), and the depth D_t at the current position. The depth is calculated by the median of the D channel within the target bounding box. The filter that yielded the best correlation response ($\mathbf{h}^{(m)}$) is correlated again on the search region scaled by s_f and target presence test is carried out (Section 3.3.1). In case the test determines the target is present, the current filter is replaced, i.e., $\mathbf{h}_t \leftarrow \mathbf{h}^{(m)}$, and the re-detection process is deactivated.

3.3.1 Target presence test

Recently, a target presence test has been proposed for long-term discriminative correlation filters [30]. The test is based on computing tracking uncertainty value as a ratio $q_t = \frac{R_t}{\overline{R}}$ between the maximum correlation response in the current frame (R_t) and a moving average of these values in the recent N_q frames when the target was visible. The test considers target lost whenever the ratio exceeds a pre-defined threshold $q_t > \tau_q$. It was showed in [30] that the test is robust to a range of thresholds.

To allow early occlusion detection, however, [19] introduce a test that compares the area of the segmentation mask with the area of the axis-aligned bounding box of the DCF. This test improves performance during occlusion, but gradual errors in scale estimation result in disagreement between the bounding box and the actual object and might lead to a reduced accuracy of the test.

The two tests are complementary and computationally very efficient, and the target presence is reported only if the considered target position passes the both tests.

3.4. Object tracking by reconstruction

Our object tracking by reconstruction approach (OTR) is summarized as follows.

Initialization. The tracker is initialized from a bounding box in the first frame. Color and depth histograms are sampled as in [19] and a segmentation mask $\mathbf{m}$ is generated. The segmentation mask $\mathbf{m}$ is used to learn the initial filter $\mathbf{h}_0$ according to (2), as well as to identify target pixels in the RGB-D model to initialize the pre-image Θ_0 by [33]. The set of snapshots is set to an empty set.

Localization. A tracking iteration at frame t starts with the target position $\mathbf{x}_{t-1}$ from the previous frame. A region is extracted around $\mathbf{x}_{t-1}$ in the current image and the position $\mathbf{x}_t$ with maximum correlation response is computed using the current filter $\mathbf{h}_{t-1}$ (along with all snapshots every N_R frames) as described in Section 3.2. The position $\mathbf{x}_t$ is tested using the target presence test from Section 3.3.1. If the test is passed, the target is considered as well localized, and the visual models (i.e., filters and pre-image) are updated. Otherwise, target re-detection (Section 3.3) is activated in the next frame.

Update. A color+depth segmentation mask $\mathbf{m}$ is computed within a region centered at $\mathbf{x}_t$ according to [19] to identify target pixels. The corresponding RGB-D pixels are used to update the pre-image Θ_t, i.e., the 3D surfel representation along with its 3D pose (Section 3.2.2).

The filter $\mathbf{h}_{t-1}$ is updated (3) by the filter learned at the current position (2) with support constraint computed from the pre-image (Section 3.2.1). Finally, the target aspect change is computed using the updated pre-image and the set of snapshots are updated if significant appearance change is detected (Section 3.2.3)

4. Experimental analysis

In this section, we validate OTR by a comprehensive experimental evaluation. The implementation details are pro-

Table 1. Experiments on the Princeton Tracking Benchmark using the PTB protocol. Numbers in the parenthesis are the ranks.

Method	*Avg.* Rank	*Avg.* Success	*Human*	*Animal*	*Rigid*	*Large*	*Small*	*Slow*	*Fast*	*Occ.*	*No-Occ.*	*Passive*	*Active*
OTR	**2.36**	**0.769(1)**	0.77(2)	0.68(6)	0.81(2)	0.76(4)	**0.77(1)**	0.81(2)	**0.75(1)**	0.71(3)	0.85(2)	**0.85(1)**	0.74(2)
ca3dms+toh [26]	4.55	0.737(5)	0.66(9)	0.74(2)	**0.82(1)**	0.73(7)	0.74(2)	0.80(4)	0.71(7)	0.63(9)	**0.88(1)**	0.83(2)	0.70(6)
CSR-rgbd++ [19]	5.00	0.740(3)	0.77(3)	0.65(8)	0.76(7)	0.75(5)	0.73(3)	0.80(3)	0.72(4)	0.70(4)	0.79(8)	0.79(6)	0.72(4)
3D-T [3]	5.64	0.750(2)	**0.81(1)**	0.64(9)	0.73(12)	**0.80(1)**	0.71(6)	0.75(9)	0.75(2)	**0.73(1)**	0.78(11)	0.79(7)	0.73(3)
PT [35]	6.09	0.733(6)	0.74(6)	0.63(11)	0.78(3)	0.78(3)	0.70(7)	0.76(5)	0.72(6)	0.72(2)	0.75(13)	0.82(4)	0.70(7)
OAPF [31]	6.09	0.731(7)	0.64(12)	**0.85(1)**	0.77(6)	0.73(8)	0.73(5)	**0.85(1)**	0.68(9)	0.64(8)	0.85(3)	0.78(9)	0.71(5)
DLST [1]	6.45	0.740(4)	0.77(4)	0.69(5)	0.73(13)	0.80(2)	0.70(9)	0.73(11)	0.74(3)	0.66(6)	0.85(4)	0.72(13)	**0.75(1)**
DM-DCF [20]	6.91	0.726(8)	0.76(5)	0.58(13)	0.77(5)	0.72(9)	0.73(4)	0.75(8)	0.72(5)	0.69(5)	0.78(10)	0.82(3)	0.69(9)
DS-KCF-Shape [16]	7.27	0.719(9)	0.71(7)	0.71(4)	0.74(9)	0.74(6)	0.70(8)	0.76(6)	0.70(8)	0.65(7)	0.81(6)	0.77(11)	0.70(8)
DS-KCF [6]	9.91	0.693(11)	0.67(8)	0.61(12)	0.76(8)	0.69(10)	0.70(10)	0.75(10)	0.67(11)	0.63(10)	0.78(12)	0.79(8)	0.66(10)
DS-KCF-CPP [16]	10.09	0.681(12)	0.65(10)	0.64(10)	0.74(10)	0.66(12)	0.69(12)	0.76(7)	0.65(12)	0.60(12)	0.79(9)	0.80(5)	0.64(12)
hiob-lc2 [36]	10.18	0.662(13)	0.53(13)	0.72(3)	0.78(4)	0.61(13)	0.70(11)	0.72(12)	0.64(13)	0.53(13)	0.85(5)	0.77(12)	0.62(13)
STC [40]	10.45	0.698(10)	0.65(11)	0.67(7)	0.74(11)	0.68(11)	0.69(13)	0.72(13)	0.68(10)	0.61(11)	0.80(7)	0.78(10)	0.66(11)

vided in Section 4.1. Performance analysis on two challenging RGB-D datasets, Princeton Tracking Benchmark (PTB) and STC, is reported in Section 4.2 and Section 4.3, respectively. Ablation studies are presented in Section 4.4 to verify our design choices.

4.1. Implementation details

We use HOG features [10] and colornames [37] in our tracker. The parameters related to the tracker are taken from [19]. The ICP error threshold is empirically set to $\tau_{\text{ICP}} = 5 \cdot 10^{-4}$ and the aspect ratio change threshold is set to $\tau_\rho = 0.20$. Maximum filter evaluation period is equal to $N_{\text{R}} = 5$ frames and $\alpha_s = 1.07$. All experiments are run on a single laptop with Intel Core i7 3.6GHz and the parameters for both tracking and 3D reconstruction are kept constant throughout the experiments. Our non-optimized implementation runs at 2 fps.

4.2. Performance on PTB benchmark [35]

The Princeton Tracking Benchmark [35] is the most comprehensive and challenging RGB-D tracking benchmark to date. The authors have recorded and manually annotated 100 RGB-D videos in real-life conditions using a Kinect v1.0. Ground truth bounding boxes of five sequences are publicly available whereas the ground truth for the remaining 95 sequences are kept hidden to prevent overfitting. Tracking performance is evaluated on the 95 sequences with the hidden ground-truth. The sequences are grouped into 11 categories: *Human*, *Animal*, *Rigid*, *Large*, *Small*, *Slow*, *Fast*, *Occlusion*, *No Occlusion*, *Passive* and *Active*. We use Bibi *et al*. [3] protocol with improved depth registration in the experiments.

The performance is measured by employing a PASCAL VOC [13] type of evaluation. Per-frame overlap o_t is defined as

$$o_t = \begin{cases} \frac{area(B_{TR} \cap B_{GT})}{area(BTR \cup B_{GT})}, & \text{if both } B_{TR} \text{ and } B_{GT} \text{ exist} \\ 1, & \text{if neither } B_{TR} \text{ and } B_{GT} \text{ exists} \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

where B_{TR} is the output bounding box of the tracker and B_{GT} is the ground truth bounding box. Tracking performance is given as *success rate* which represents average overlap [7]. The PTB evaluation protocol sorts the trackers according to the primary performance measures with respect to each object category and computes the final ranking as the average over these ranks. In addition, the overall success rate is reported for detailed analysis.

The OTR tracker is compared to all trackers available on the PTB leaderboard: ca3dms+toh [26], CSR-rgbd++ [19], 3D-T [3], PT [35], OAPF [31], DM-DCF [20], DS-KCF-Shape [16], DS-KCF [6], DS-KCF-CPP [16], hiob_lc2 [36] and we added two recent trackers STC [40] and DLST [1]. Results are reported in Table 1.

OTR convincingly sets the new *state-of-the-art* in terms of both overall ranking and the average success by a large margin compared to the next-best trackers (Table 1). In terms of average success, OTR obtains a 4.3% gain compared to the second ranking tracker ca3dms+toh [26], which tracks the target in 3D as well, but without reconstruction. This result speaks in favour of our 3D-based pre-image construction and its superiority for RGB-D tracking.

In addition to being the top overall tracker, the performance of OTR is consistent across all categories. OTR is consistently among the top trackers in each category and achieves the top rank in three categories and the second best in five categories. This suggests that our tracker does not overfit to a certain type of scenario and it generalizes very well unlike some other methods in the benchmark.

A closely related work to our own is recent CSR-rgbd++, which combines a single CSR-DCF with color and depth segmentation and implements a target re-detection. OTR obtains a significant 6.6% increase over CSR-rgbd++ in *Rigid* category, which speaks in favor of our DCFs approach with several views connected to a 3D pre-image that localizes the target more precisely. On the *No-Occ.* category, OTR outperforms CSR-rgbd++ by a 7.6% success rate. This can be attributed to the advantage of using a pre-image Θ for DCF training described in Section 3.2.1.

Table 2. The normalized area under the curve (AUC) scores computed from one-pass evaluation on the STC Benchmark [40].

		Attributes									
Method	*AUC*	*IV*	*DV*	*SV*	*CDV*	*DDV*	*SDC*	*SCC*	*BCC*	*BSC*	*PO*
OTR	**0.49**	**0.39**	**0.48**	**0.31**	0.19	**0.45**	**0.44**	0.46	**0.42**	**0.42**	**0.50**
CSR-rgbd++ [19]	0.45	0.35	0.43	0.30	0.14	0.39	0.40	0.43	0.38	0.40	0.46
ca3dms+toh [26]	0.43	0.25	0.39	0.29	0.17	0.33	0.41	**0.48**	0.35	0.39	0.44
STC [40]	0.40	0.28	0.36	0.24	**0.24**	0.36	0.38	0.45	0.32	0.34	0.37
DS-KCF-Shape [16]	0.39	0.29	0.38	0.21	0.04	0.25	0.38	0.47	0.27	0.31	0.37
PT [35]	0.35	0.20	0.32	0.13	0.02	0.17	0.32	0.39	0.27	0.27	0.30
DS-KCF [6]	0.34	0.26	0.34	0.16	0.07	0.20	0.38	0.39	0.23	0.25	0.29
OAPF [31]	0.26	0.15	0.21	0.15	0.15	0.18	0.24	0.29	0.18	0.23	0.28

4.3. Performance on STC benchmark [40]

The STC benchmark [40] has been recently published to complement the PTB benchmark in the number of categories and diversity of sequences. 36 sequences are recorded indoors and outdoors using Asus Xtion sensors and the authors annotated every frame of every video with 10 attributes; *Illumination variation* (IV), *Depth variation* (DV), *Scale variation* (SV), *Color distribution variation* (CDV), *Depth distribution variation* (DDV), *Surrounding depth clutter* (SDC), *Surrounding color clutter* (SCC), *Background color camouflages* (BCC), *Background shape camouflages* (BSC), *Partial occlusion* (PO). These attributes were either automatically computed or manually annotated.

The tracking performance is measured by precision and success plots computed from a one-pass evaluation akin to [39]. Success plot shows the portion of correctly tracked frames with respect to the different values of the overlap thresholds. Tracking performance is measured by a non-normalized area under the curve on this graph, i.e., the sum of values on the plot. The standard AUC measure [39] is obtained by dividing the non-normalized AUC by the number of overlap thresholds. The number of thresholds is the same for all evaluated trackers and only scales the non-normalized AUC to interval $[0, 1]$. We therefore report the standard AUC values, which is the more familiar measure in the tracking community. Precision plot is constructed similarly to success plot, by measuring the portion of frames with center-error smaller than a threshold. The overall measure on precision plot is computed as the value at 20 pixels error threshold.

The OTR tracker is compared to the following trackers: CSR-rgbd++ [19], ca3dms+toh [26], STC [40], DS-KCF-Shape [16], PT [35], DS-KCF [6] and OAPF [31]. The results are presented in Table 2 and Figure 4. As on PTB benchmark (Section 4.2), OTR outperforms the *state-of-the-art* by a large margin not only in the overall score but in most of the categories except *CDV (Color Distribution Variation)* and *SCC (Surrounding Color Clutter)*, where it is ranked among top three trackers. The overall top performance and excellent per-attribute performance support our observations on PTB benchmark that OTR is capable of handling various tracking scenarios and generalizes well over the different datasets. Qualitative tracking results on the four sequences from STC dataset are shown in Figure 5. The computing times for the three best performing trackers are 2 fps (OTR), 6 fps (CSR-rgbd++) and 34 fps (ca3dms+toh).

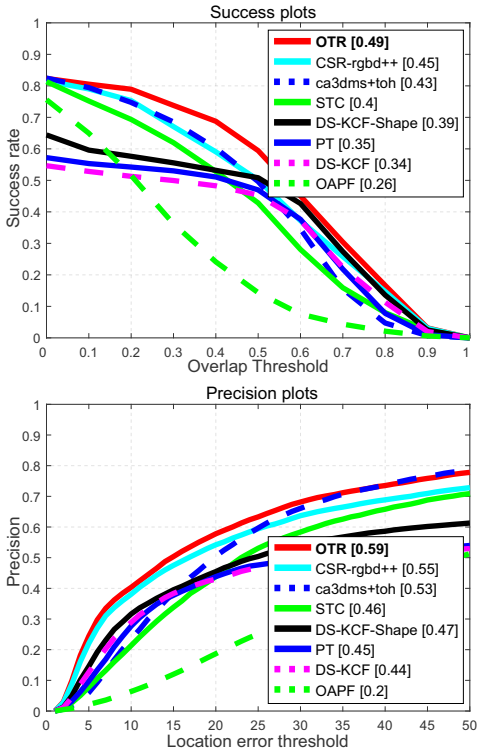

Figure 4. Success and precision plots on STC benchmark [40].

4.4. Ablation studies

The main components of our tracker are (i) the 3D-based pre-image, which provides an improved target segmentation, (ii) the set of multiple view-specific target DCFs and (iii) the interaction between the former two components. An ablation study is conducted on the PTB [35] dataset to evaluate the extent of contribution of each component. We implemented three variants of the proposed tracker with the

Table 3. Ablation studies on the PTB benchmark [35].

Method	Avg. Success	*Human*	*Animal*	*Rigid*	*Large*	*Small*	*Slow*	*Fast*	*Occ.*	*No-Occ.*	*Passive*	*Active*
OTR	**0.769**	0.77	**0.68**	**0.81**	**0.76**	**0.77**	**0.81**	**0.75**	**0.71**	**0.85**	**0.85**	**0.74**
OTR_{-3D}	0.747	0.76	0.62	0.80	0.75	0.75	0.80	0.72	0.69	0.82	0.84	0.71
OTR_{-VS}	0.743	0.75	0.66	0.77	0.74	0.74	0.79	0.72	0.67	0.84	0.81	0.72
OTR_{-3D-VS}	0.740	**0.78**	0.61	0.76	0.75	0.73	0.79	0.72	**0.71**	0.78	0.79	0.72

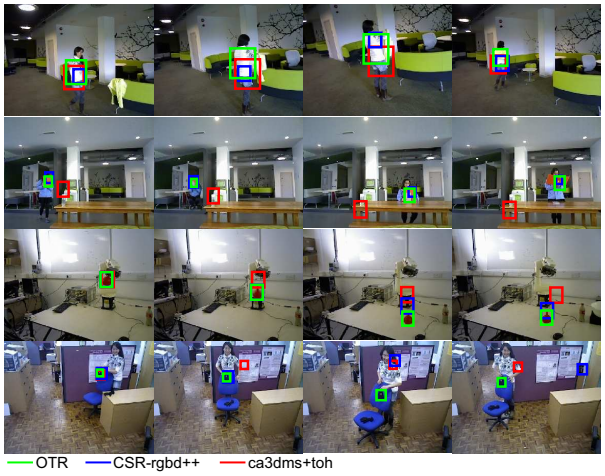

Figure 5. Tracking results on four sequences from STC dataset [40]. The proposed OTR tracker confidently tracks the target undergoing a substantial pose change. Two state-of-the-art RGB-D trackers (CSR-rgbd++ [19] and cs3dms+toh [26]), that do not apply the multi-view DCFs nor target 3D pre-image, result in less accurate localization or failure.

3D pre-image and view-specific DCFs. The first variant is the tracker without the 3D pre-image, denoted as OTR_{-3D}. The second variant is the tracker without the view-specific DCFs (OTR_{-VS}) and the third variant is the tracker without the view-specific DCFs and without the 3D pre-image (OTR_{-3D-VS}).

The results of the ablation study are reported in Table 3. The proposed OTR with all components achieves a 0.769 success rate. Removing the view-specific target representation (OTR_{-VS}) or 3D pre-image (OTR_{-3D}) result to approx. 3% success rate drop in tracking performance (0.747 and 0.743). Removing both view-specific and 3D pre-image representation (OTR_{-3D-VS}) further reduces the tracking performance to 0.740 success rate.

On the *Occlusion* category the OTR tracker outperforms the version without a view-specific formulation (OTR_{-VS}) by 6% increase in the success rate. The view-specific set of DCFs *remembers* the target appearance from different views, which helps in reducing drifting and improves re-detection accuracy after occlusion. On average, 4 views were automatically generated by the view-specific DCF in OTR per tracking sequence. The tracker version without the view-specific formulation *forgets* the past appearance, which reduces the re-detection capability.

In situations without occlusion, the 3D pre-image plays a more important role than the view-specific DCF formulation. Removing the 3D pre-image creation from the tracker results in 7% success rate reduction, which indicates the significant importance of using the 3D pre-image for robust DCF learning.

Overall, the addition of 3D pre-image and view-specific target representation improves performance of the baseline version OTR_{-3D-VS} by approximately 4% in tracking success rate. The ablation study results conclusively show that every component importantly contributes to the tracking performance boost.

5. Conclusions

A new long-term RGB-D tracker, called OTR – Object Tracking by Reconstruction is presented. The target 3D model, a pre-image, is constructed by a surfel-based ICP. The limited convergence range of the ICP and the requirement to automatically identify object pixels used for reconstruction is addressed by utilizing a DCF for displacement estimation and for approximate target segmentation. The 3D pre-image in turn constrains the DCF learning, and is used for generating view-specific DCFs. These are used for localization as well as for target re-detection, giving the tracker a long-term tracking quality.

The OTR tracker is extensively evaluated on two challenging RGB-D tracking benchmarks and compared to 12 *state-of-the-art* RGB-D trackers. OTR outperforms all trackers by a large margin, setting a new *state-of-the-art* on these benchmarks. An ablation study verifies that the performance improvements come from the 3D pre-image construction, the view-specific DCF set and the interaction between the two.

The view-specific DCF formulation allows long-term tracking of poorly textured and small objects over large displacements. Our future work will focus on extension to model-based tracking with pre-learned models on realistic, open-world scenarios. In addition, we plan to consider improvements by ICP robustification and deep features.

Acknowledgments

This work is supported by Business Finland under Grant 1848/31/2015 and Slovenian research agency program P2-0214 and project J2-8175. J. Matas is supported by the Technology Agency of the Czech Republic project TE01020415 – V3C Visual Computing Competence Center.

References

[1] N. An, X.-G. Zhao, and .-G. Hou. Online RGB-D Tracking via Detection-Learning-Segmentation. In *ICPR*, 2016.

[2] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-Convolutional Siamese Networks for Object Tracking. In *ECCV Workshops*, 2016.

[3] A. Bibi, T. Zhang, and B. Ghanem. 3D Part-Based Sparse Tracker with Automatic Synchronization and Registration. In *CVPR*, 2016.

[4] D. S. Bolme, J.R. Beveridge, B. A. Draper, and Y.-M. Lui. Visual Object Tracking using Adaptive Correlation Filters. In *CVPR*, 2010.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, 2011.

[6] M. Camplani, S. Hannuna, M. Mirmehdi, D. Damen, A. Paiement, L. Tao, and T. Burghardt. Real-time RGB-D Tracking with Depth Scaling Kernelised Correlation Filters and Occlusion Handling. In *BMVC*, 2015.

[7] L. Čehovin, A. Leonardis, and M. Kristan. Visual Object Tracking Performance Measures Revisited. *IEEE TIP*, 25(3):1261–1274, 2016.

[8] F. Chaumette, P. Rives, and B. Espiau. Positioning of a Robot with Respect to an Object, Tracking it and Estimating its Velocity by Visual Servoing. In *ICRA*, 1991.

[9] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-Based Object Tracking. *IEEE PAMI*, 25:564–567, 2003.

[10] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR*, 2005.

[11] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg. Discriminative Scale Space Tracking. *IEEE PAMI*, 39(8):1561–1575, 2017.

[12] M. Danelljan, F. S. Khan, M. Felsberg, and J. van de Weijer. Adaptive Color Attributes for Real-Time Visual Tracking. In *CVPR*, 2014.

[13] M. Everingham, L. van Gool, C. K. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 88(2):303–338, 2010.

[14] H.K. Galoogahi, T. Sim, and S. Lucey. Correlation Filters with Limited Boundaries. In *CVPR*, 2015.

[15] H. K. Galoogahi, T. Sim, and S. Lucey. Multi-channel Correlation Filters. In *ICCV*, 2013.

[16] S. Hannuna, M. Camplani, J. Hall, M. Mirmehdi, D. Damen, T. Burghardt, A. Paiement, and L. Tao. DS-KCF: A Real-time Tracker for RGB-D Data. *Journal of Real-Time Image Processing*, 2016.

[17] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-Speed Tracking with Kernelized Correlation Filters. *IEEE PAMI*, 37(3):583–596, 2015.

[18] W. Hu, T. Tan, L. Wang, and S. Maybank. A Survey on Visual Surveillance of Object Motion and Behaviors. *IEEE Transactions on Systems, Man, and Cybernetics*, 34(3):334–352, 2004.

[19] U. Kart, J.-K. Kämäräinen, and J. Matas. How to Make an RGBD Tracker ? In *ECCV Workshops*, 2018.

[20] U. Kart, J.-K. Kämäräinen, J. Matas, L. Fan, and F. Cricri. Depth Masked Discriminative Correlation Filter. In *ICPR*, 2018.

[21] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *ISMAR*, 2007.

[22] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin, T. Vojír, and et al. The Visual Object Tracking VOT2016 Challenge Results. In *ECCV Workshops*, 2016.

[23] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, and et al. The Visual Object Tracking VOT2017 Challenge Results. In *ICCV Workshops 2017*.

[24] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Čehovin, G. Nebehay, T. Vojír, and et al. The Visual Object Tracking VOT2014 Challenge Results. In *ECCV Workshops*, 2014.

[25] K. Lebeda, S. Hadfield, and R. Bowden. 2D Or Not 2D: Bridging the Gap Between Tracking and Structure from Motion. In *ACCV*, 2014.

[26] Y. Liu, X.-Y. Jing, J. Nie, H. Gao, J. Liu, and G.-P. Jiang. Context-aware 3-D Mean-shift with Occlusion Handling for Robust Object Tracking in RGB-D Videos. *IEEE TMM*, 2018.

[27] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *IJCAI*, 1981.

[28] A. Lukežič, T. Vojír, L. Čehovin, J. Matas, and M. Kristan. Discriminative Correlation Filter with Channel and Spatial Reliability. In *CVPR*, 2017.

[29] A. Lukežič, T. Vojír, L. Čehovin Zajc, J. Matas, and M. Kristan. Discriminative Correlation Filter Tracker with Channel and Spatial Reliability. *IJCV*, 2018.

[30] A. Lukežič, L. Čehovin Zajc, T. Vojír, J. Matas, and M. Kristan. FCLT - A Fully-Correlational Long-Term Tracker. In *ACCV*, 2018.

[31] K. Meshgi, S. Maeda, S. Oba, H. Skibbe, Y. Li, and S. Ishii. An Occlusion-aware Particle Filter Tracker to Handle Complex and Persistent Occlusions. *CVIU*, 150:81 – 94, 2016.

[32] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011.

[33] M. Rünz and L. Agapito. Co-Fusion: Real-time Segmentation, Tracking and Fusion of Multiple Objects. In *ICRA*, 2017.

[34] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual Tracking: An Experimental Survey. *IEEE PAMI*, 36(7):1442–1468, 2014.

[35] S. Song and J. Xiao. Tracking Revisited Using RGBD Camera: Unified Benchmark and Baselines. In *ICCV*, 2013.

[36] P. Springstübe, S. Heinrich, and S. Wermter. Continuous Convolutional Object Tracking. In *ESANN*, 2018.

[37] J. van de Weijer, C. Schmid, J. Verbeek, and D. Larlus. Learning Color Names for Real-world Applications. *IEEE TIP*, 18(7):1512–1523, 2009.

[38] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger. Elasticfusion. *Int. J. Rob. Res.*, 35(14):1697–1716, 2016.

[39] Y. Wu, J. Lim, and Y. Ming-Hsuan. Object Tracking Benchmark. *IEEE PAMI*, 37:1834 – 1848, 2015.

[40] J. Xiao, R. Stolkin, Y. Gao, and A. Leonardis. Robust Fusion of Color and Depth Data for RGB-D Target Tracking Using Adaptive Range-Invariant Depth Models and Spatio-Temporal Consistency Constraints. *IEEE Transactions on Cybernetics*, 48:2485 – 2499, 2018.