Real-time RGB-D Tracking with Depth Scaling Kernelised Correlation Filters and Occlusion Handling

Massimo Camplani
massimo.camplani@bristol.ac.uk
Sion Hannuna
sh1670@bristol.ac.uk
Majid Mirmehdi
majid@cs.bris.ac.uk
Dima Damen
dima.damen@bristol.ac.uk
Adeline Paiement
csatmp@bristol.ac.uk
Lili Tao
lili.tao@bristol.ac.uk
Tilo Burghardt
tb2935@bristol.ac.uk

Visual Information Laboratory
Faculty of Engineering
University of Bristol
Bristol, UK

Abstract

We present a real-time RGB-D object tracker which manages occlusions and scale changes in a wide variety of scenarios. Its accuracy matches, and in many cases outperforms, state-of-the-art algorithms for precision and it far exceeds most in speed. We build our algorithm on the existing colour-only KCF tracker which uses the 'kernel trick' to extend correlation filters for fast tracking. We fuse colour and depth cues as the tracker's features and exploit the depth data to both adjust a given target's scale and to detect and manage occlusions in such a way as to maintain real-time performance, exceeding on average 35fps when benchmarked on two publicly available datasets. We make our easy-to-extend modularised code available to other researchers.

1 Introduction

Object tracking is one of the fundamental components in many vision based applications and a very active research area due to challenges such as object appearance changes, illumination variations, occlusions, shape deformation, and camera motion. While the majority of tracking algorithms are based on 2D visual cues (hereafter referred to as RGB trackers), the recent surge in popularity of real-time RGB-D sensors has encouraged research into combining colour and depth data for tracking.

Pages 145.1-145.11
DOI: https://dx.doi.org/10.5244/C.29.145

The results from a few, recent works in RGB-D tracking have demonstrated that state-of-the-art RGB tracking algorithms can be outperformed by approaches that fuse colour and depth, for example [6, 16, 18, 19]. Song and Xiao [18], presented one of the first RGB-D tracking works using off the shelf depth sensors. HOG features computed on both colour and depth appearance data were linearly combined with the outcome of optical flow and classified using a Support Vector Machine (SVM) classifier to improve efficiency. Occlusion was also handled to a considerable extent in [18], based on the assumption that the depth distribution of the plane closest to the camera contains the tracked object. During occlusion, optical flow tracked the object occluding the target until the target re-emerged. The overall approach was computationally expensive at 0.26 fps on average, due to the exhaustive search, optical flow computations, and elaborate colour and depth segmentation. However, in terms of precision it outperformed state of the art 'RGB only' trackers. In Wang et al. [19], the target region's colour histogram and optical flow, was combined with the target's mean depth, to track its motion. Their work did not detail how their model was updated, and their results were reported on sequences not publicly available. The algorithm presented by Garcia et al. in [6] extended the condensation-based RGB tracker of Klein and Cremers [13] to incorporate depth data and predict the 3D spatial state of the particles in the condensation algorithm. Their boosting classifier was built from a pool of grayscale, colour, and depth features, and in particular, the invariant gradient features of [13] were extended to depth data. A small set of candidate features were kept in the pool, providing the right balance between computational efficiency and accuracy. Tracking was adaptive as their classifier was re-trained with tracked examples. Occlusions were detected when the tracker response was below a certain threshold. They reported an average processing rate of 30 fps. A recently published RGB-D tracker is [16], in which a colour and depth based 'occlusion aware' particle filter tracking framework was introduced. A particle represents a region's bounding box and an occlusion flag. When this flag's value exceeds a fixed threshold occlusion is detected and the bounding box search area is expanded. The authors report an average processing rate of 1 fps.

In this paper, we propose a real-time RGB-D tracker that is based on, and improves upon, the RGB Kernelised Correlation Filters tracker (KCF) from [9]. The KCF tracker combines high accuracy and fast processing speeds as demonstrated in [9] and elsewhere, e.g. [20], where over 150fps processing was reported. We enhance the RGB KCF tracker in three ways: (i) we fuse colour and depth cues, evaluating a variety of feature combination strategies, (ii) we exploit the target's depth distribution to identify scale changes and efficiently model these scale changes in the Fourier domain (iii) we handle occlusions by identifying sudden changes in the target region's depth histogram and recovering lost tracks by searching for the unoccluded object in specifically identified key areas. These improvements are all achieved while maintaining a high frame processing throughput of on average 35fps. The proposed RGB-D tracker, which we refer to as the Depth Scaling Kernalised Correlations Filters (DS-KCF) tracker, is compared against state-of-the-art algorithms [6, 16, 18] on publicly available datasets. The proposed tracker is efficient since only a single target model is kept and updated, outperforming the KCF modification proposed in [15] where multiple models are continuously updated. The Matlab code is available[1] for research purposes and comparative evaluation.

In Section 2, the main aspects of the KCF tracker are described to align the reader with its methodology. Then, in Section 3 the proposed DS-KCF tracker is presented. Experimental evaluation of our new tracker appears in Section 4 and conclusions are outlined in Section 5.

[1] See http://www.irc-sphere.ac.uk/work-package-2/DS-KCF

2 The KCF tracker

Henriques et al. [9] proposed the use of the 'kernel trick' [17] to extend correlation filters for very fast RGB tracking. Their so-called KCF tracker has important characteristics, in particular its ability to combine high accuracy and processing speed. In the comprehensive RGB tracking benchmarking work in [20], KCF ranked in the top ten while coming in first as the fastest and obtaining accuracy comparable to other state-of-the-art approaches. These attributes make KCF a method of choice for those who need a very fast and reliable tracker. The processing pipeline in the KCF tracker comprises training, detection and retraining at the new target location. Here, we introduce KCF, focusing on issues relevant to our proposed extensions. For a detailed description and proofs the reader is referred to [9]. Additionally, the reader should refer to the works presented in [2, 5, 9, 10] for a comprehensive view of correlation filters.

KCF exploits the properties of circulant matrices to achieve efficient learning by implicitly encoding convolution. Given the circulant matrix $C(\mathbf{x})$, and $C(\mathbf{x})\mathbf{y}$ as the convolution of $\mathbf{x}$ and $\mathbf{y}$, then it may be performed in the Fourier domain via element wise multiplication:

$$C(\mathbf{x})\mathbf{y} = \mathcal{F}^{-1}(\mathcal{F}^{*}(\mathbf{x}) \odot \mathcal{F}(\mathbf{y})), \tag{1}$$

where $*$ is the complex conjugate, $\odot$ is the element wise multiplication and $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the Fourier transformation and its inverse. Applying the above ideas to linear regression, it may be shown that the standard formulation for ridge regression, where X is the design matrix (X^H is the Hermitian transpose), λ is a penalty term and $\mathbf{y}$ is the regression target, is $\mathbf{w} = (X^H X + \lambda I)^{-1} X^T \mathbf{y}$ and may be reformulated as:

$$\mathcal{F}(\mathbf{w}) = \frac{\mathcal{F}^{*}(\mathbf{x}) \odot \mathcal{F}(\mathbf{y})}{\mathcal{F}^{*}(\mathbf{x}) \odot \mathcal{F}(\mathbf{x}) + \lambda}, \tag{2}$$

where the fraction indicates element wise division. The efficiency of KCF is apparent from (2), as it reduces the computational complexity of the general ridge regression problem from $O(n^3)$ to element-wise operations. To recover the values of $\mathbf{w}$, the DFT complexity is $O(n \log n)$. Similar expressions can be derived also when non-linear regression is used. By using the kernel trick, the solution $\mathbf{w}$ may be stated as $\mathbf{w} = \sum_i \alpha_i \varphi(\mathbf{x}_i)$ where the variables α_i need to be estimated instead of $\mathbf{w}$ and φ is the function that maps $\mathbf{x}$ into the nonlinear feature space. The kernelised version of ridge regression can be written as: $\alpha = (K + \lambda I)^{-1} \mathbf{y}$, where K is the circulant Kernel matrix, with elements K_{ij} corresponding to $\gamma(x_i, x_j)$ and γ is the selected kernel function. As in [9], this can be reformulated using similar circulant matrices concepts such that

$$\alpha = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(\mathbf{y})}{\mathcal{F}^{*}(\gamma^{xx}) + \lambda}\right), \tag{3}$$

where γ^{xx} is the first row of K. We refer to (3) generally as the training phase of the KCF tracker. Once α is calculated, the circulant matrix properties can be exploited to measure the response of the classifier $f(\mathbf{z})$ at various image patches $\mathbf{z}$,

$$f(\mathbf{z}) = \mathcal{F}^{-1}\left(\mathcal{F}^{*}(\gamma^{xz}) \odot \mathcal{F}(\alpha)\right). \tag{4}$$

This operation is performed in the Fourier domain, employing element wise multiplications and the DFT. We use the Gaussian kernel as it demonstrated optimal tradeoff between accuracy and computational complexity in [9]. To summarise, the KCF tracker is based on a

simple processing chain. An image patch is extracted at the estimated target location, and a precomputed cosine window is applied to the patch to reduce the noise in the Fourier domain. The target position is detected by maximising $f(\mathbf{z})$ (as in (4)). The model is trained using Gaussian shaped regression targets that provide smooth responses. Model update is introduced by linearly interpolating the new parameters α and $\mathbf{x}$ with the current ones. In [15], KCF was modified to cope with scale change, yet a brute force search for scale was used - in particular, one model per scale was constantly estimated and updated. Target position and the current target scale were selected as the one corresponding to the maximum response value among all the scales.

3 Proposed DS-KCF tracker

We propose extending the RGB KCF tracker into an RGB-D tracker based on the efficient combination of colour and depth features, along with more efficient management of scale changes and occlusions. The improvements we implement provide higher rates of accuracy while still operating at better than real-time frame rates. The block diagram of the proposed Depth Scaling Kernelised Correlation Filters tracker is shown in Figure 1. Depth data in the target region is segmented (Section 3.1) to extract relevant features for the target's depth distribution. Modelled as a Gaussian distribution, changes in scale guide the update in the target's model (Section 3.2). Region depth distribution enables the detection of possible occlusions. During an occlusion, the model is not updated and the occluding object is tracked to guide the target's search space (Section 3.3). Note that the KCF tracker has been selected for its unique combination of accuracy and real time performance, however, the proposed RGB-D extension paradigm can potentially be ported to other RGB trackers.

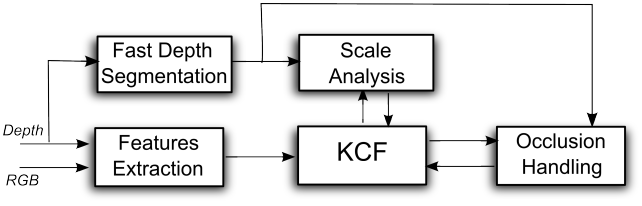

Figure 1: Block diagram of the proposed DS-KCF tracker.

3.1 Fast Depth Segmentation

The first improvement to the KCF tracker we introduce is to fuse depth information into its RGB approach. Indeed, the scale change and occlusion handling described later are also made possible by the proposed two-stage depth segmentation approach: (a) a fast one-dimensional (1D) implementation of K-means to estimate initial clusters or regions of interest (ROI), followed by, (b) connected component analysis that incorporates spatial features to refine the ROI. K-means' computational burden depends on the number of clusters, the number of points to be clustered and the feature dimensionality. We apply K-means to a tracked region's 1D depth histogram, and hence reduce the number of features and the number of points. A similar approach was proposed for colour image segmentation in [4]. We initialise the centroids to the local maxima of the equally-spaced depth histogram. Each

cluster is then assigned to the closest bin and the centroids are updated until convergence. In the second stage, connected components are analysed in the image plane to distinguish between objects located within the same depth plane, and to remove clusters corresponding to small regions. The bin width influences the results of the segmentation. This is selected adaptively according to the tracked object's standard deviation (σ_{obj}) as well as the noise model of the depth device [9, 17]. The target region Ω_{obj} corresponds to the cluster with the minimum mean depth. The cluster's mean μ_{obj} and standard deviation σ_{obj} are constantly updated while the object is tracked.

3.2 Detecting and handling scale changes

As described in Section 2, one way that the KCF tracker achieves faster throughput is by substituting convolution in the spatial domain with element-wise multiplication in the frequency domain. However, element-wise operations expect the matrices to be of the same size. Accordingly, we propose to estimate the target object scale s_{obj} by scaling the object's template relative to its initial depth $d^{t_0}_{obj}$. Our approach utilises two types of scale factors. The first is a continuous scale factor $\mathcal{S}^r = d_{obj}/d^{t_0}_{obj}$ obtained from the relative depth of the target. The second is a set of quantised scale factors $\mathcal{S}^q = \{s_j, (\forall j = 1 \dots J)\}$, which enables pre-computing different matrices (the regression targets of the training phase and the pre-processing cosine windows in Section 2). The current scale is chosen to be the closest level $s_j \in \mathcal{S}^q$ to $\mathcal{S}^r$. We use $\mathcal{S}^q$ to improve computational efficiency and tracker robustness as the models and coefficients are not constantly re-estimated, and at the same time $\mathcal{S}^r$ helps to refine the tracker's output as it represents a finer change in scale.

When a change in scale is detected, the model template needs to be updated in the Fourier domain and we use *interpolation* and *decimation* for increases and decreases in scale respectively. This solution is more stable than model reintialisation, especially in case of very frequent changes of scale. Model resampling overhead is added when the tracker moves to a different scale level in $\mathcal{S}^q$. Yet, during tracking, the proposed method that only one target's model at scale s_{obj}, corresponding to the selected s_j, is kept and updated. When scaling up, interpolation involves zero padding the higher frequency Fourier coefficients to fit the increased template size, and then adjusting the frequency component amplitude. This is due to the duality between the spatial and Fourier domains and it can be easily shown in the following example. Let us for simplicity start our analysis in the spatial domain and consider a 1D signal $f(n)$, for which we want to increase the number of samples by a factor M. *Interpolation* inserts $M-1$ zero samples such that the new set of samples $g(nM+m)$ is equal to $f(n)$ when $m=0$ and zero otherwise. Now, we can analyse the DFT of $g(nM+m)$ as

$$\begin{aligned} G(k) &= \frac{1}{\sqrt{MN}} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} g(nM+m) \exp\left(-j2\pi k(nM+m)/MN\right) \\ &= \frac{1}{\sqrt{MN}} \sum_{n=0}^{N-1} g(nM) \exp\left(-j2\pi k(nM)/MN\right) \\ &= \frac{1}{\sqrt{MN}} \sum_{n=0}^{N-1} f(n) \exp\left(-j2\pi k(nM)/MN\right) = \frac{F(k)}{\sqrt{M}}. \end{aligned} \tag{5}$$

Equation (5) shows how the Fourier coefficients $G(k)$ of the upsampled signal g can be calculated starting from the coefficients $F(k)$ of the original signal, for the scaling factor M. Importantly in DS-KCF, the zero Fourier coefficients are substituted by the corresponding ones

from the new patch. In the case of target size reduction, *decimation* retains the lower portion of the coefficients and discards those of the highest frequencies. In fact, while increasing the spatial sampling in the image, the frequencies spanned by the DFT are reduced [7].

To gain an intuition for the efficacy of this approach, consider upsampling and downsampling an image in the Fourier domain. When upsampling, higher resolution spatial features are interpolated - avoiding a pixelation in the upsampled image. Conversely, when downsampling, the coefficients representing lower frequencies are retained as the image is now at a coarser resolution. In both cases, previous models' information can be partially preserved and used to build a robust tracker.

3.3 Detecting and handling occlusions

We model the tracked object's depth distribution as a single Gaussian, and identify candidate occluding objects as the regions not belonging to this model. The work proposed in [18] is based on similar concepts as depth cue is used to discriminate the tracked object from an occluded one. However, our approach differs from [18] as it has been optimized to exploit and keep the advantages of KCF tracking core. In particular, we introduce local search for target candidates (see (7) and (8)) by considering depth continuity between the occluded target and the candidates. Furthermore, the occluding regions are segmented with fast depth segmentation (see Section 3.1) and occluding objects are tracked with RGB KCF. With this approach, we obtain a processing rate 300 times greater than that achieved in [18] (see Section 4). Again noting μ_{obj} as the mean depth of the tracked object T_{obj} and σ_{obj} as its standard deviation, then given a segmented region Ω_{obj} and its corresponding patch $\mathbf{z}$, occlusion is detected if

$$\left(\Phi\left(\Omega_{obj}\right) > \lambda_{occ}\right) \wedge \left(\widehat{f(\mathbf{z})}_{max} < \lambda_{r1}\right), \tag{6}$$

where $\Phi(\Omega_{obj})$ is the fraction of pixels belonging to Ω_{obj} up to two standard deviations from the object's mean. We have determined empirically that an occlusion should be detected when almost a third of Ω_{obj} is occupied by the occluding object T_{occ}, i.e. $\lambda_{occ} = 35\%$. The second term in (6) reduces false detections of occlusion in the case of objects moving fast towards the camera. In these situations the overlap condition can be satisfied due to a fast shift in the object's depth distribution. However, the maximum response of our tracker, $\widehat{f(\mathbf{z})}_{max}$, would still be high, with $\widehat{f(\mathbf{z})}_{max}$ obtained by weighting $f(z)$ in (4) with a sigmoidal function that takes into account distance between the depth data and μ_{obj} to guarantee continuity on depth. The value of λ_{r1} was also determined empirically as $\lambda_{r1} = 0.4$. When (6) is true, the occluding object T_{occ} is tracked. Note, only a portion of T_{occ} is contained in Ω_{obj}. To obtain the entire occluding object and the depth values of μ_{occ} and σ_{occ}, the connected component is extracted from the depth frame.

During occlusion, a search region $\Omega^i_{T_{search}}$ at frame i is defined, and its response is computed to detect the re-appearance of the target T_{obj}. In fact, the target object will re-emerge with high probability in those image areas gradually uncovered by T_{occ}. We identify the region where target candidates are searched as

$$\Omega^i_{T_{search}} = \Omega^{i-1}_{T_{occ}} \cup \Omega^{i-1}_{T_{bc}} \cup \Omega^i_{T_{occ}}, \tag{7}$$

where $\Omega^{i-1}_{T_{bc}}$ is the region previously occupied by the *b*est target *c*andidate T_{bc} (Figure 2). For tracking T_{occ}, the accuracy of the baseline KCF tracker is sufficient for our purposes as our goal is to only have a rough estimate of $\Omega_{T_{occ}}$.

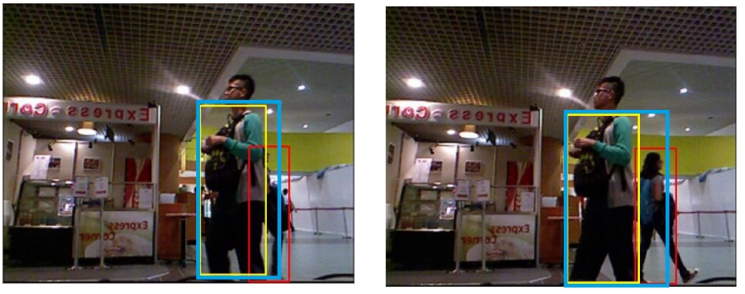

Figure 2: Occlusion search region: $\Omega^i_{T_{search}}$ (blue line), $\Omega^i_{T_{occ}}$ (yellow line), $\Omega^i_{T_{obj}}$ (red line) .

For each cluster with depth mean μ_n and position c_n in the image plane, we compute the maximum value of the normalised response $\widehat{f(z)_n}$. The best target candidate T_{bc} then corresponds to the one with maximum response $\widehat{f(z)_n}$. Target tracking is resumed when

$$\left(overlap\left(\Omega_{T_{occ}},\Omega_{T_{bc}}\right) < \lambda_{occ}\right) \vee \left(\widehat{f(z)_n} > \lambda_{r2}\right), \tag{8}$$

where the value of λ_{r2} has been empirically determined as 0.2. The computational requirements of DS-KCF are not significantly affected during occlusion as the search space enables computing the response only in specific key areas.

4 Experimental Results

We evaluate our method using two publicly available RGB-D tracking datasets both recorded with Microsoft Kinect v1. The first, the Princeton Dataset [18], contains 5 validation and 95 test sequences[2]. This dataset presents complex background clutter and intermittent occlusions. The second, the BoBoT-D Benchmark [6], is made up of 5 sequences with object rotations, occlusions and changes of scale. We compare our proposed tracker to the RGB-D trackers in [6, 16, 18], which have outperformed other RGB trackers such as [1, 8, 11, 14]. We shall refer to these as the Adapt3D[6], Prin-Track [18] and OAPF [16] trackers. Prin-Track [18] was assessed by considering three different feature sets, depth only (D), colour only (RGB) and their combination (RGB-D). Other RGB-D trackers, for example [19], do not report results on available datasets or make their code available for comparison. All our experiments were performed using Matlab on a workstation with an Intel I7-3770S CPU 3.1GHz processor and 8Gb RAM.

Comparison on the Princeton Dataset [18] - For this dataset we report Precision and Success plots [20]. Precision plots are obtained by computing the percentage of frames for which the location error is below a certain threshold, from which we select the percentage value corresponding to the threshold equal to 20 pixels (P20), as proposed in [20]. Success plots measure the bounding box overlap between the tracked object and the groundtruth, and provide the percentage of successful frames where the overlap is larger than a threshold as it is varied from 0 to 1. For these, we report the area-under-curve (AUC). We also provide the computational performance of the methods in terms of processed frames per second (fps).

[2]We synchronise and re-align their RGB and depth images as their validation set groundtruth has been estimated on only their RGB frames.

In [9], the authors of KCF proposed the use of HoG features, hence we compare DS-KCF against KCF based on not just colour HoG features, but an extended set, i.e. {*hog_colour*, *hog_depth*, *hog_linear*, *hog_max*, *hog_concat*}, where *hog_linear* linearly combines the responses from *hog_colour* and *hog_depth*, *hog_max* uses the maximum response between *hog_colour* and *hog_depth*, and *hog_concat* concatenates colour and depth features to obtain a single feature. Figure 3 and Table 1 show that the proposed DS-KCF tracker outperforms the baseline KCF leading to better results both in terms of AUC and P20 measures, whatever the feature set. In the worst and best case scenarios DS-KCF runs on average at 40 and 60fps respectively, which is better than real-time.

To study the impact of DS-KCF's proposed scale analysis, let's consider the performance on sequences 'child_no1' and 'zcup_move_1' where scale changes are prominent and there is no occlusion. As shown in Table 1, when using the same feature (*hog_colour*) for both KCF and DS-KCF, the latter leads to better success rate (AUC) and an equivalent precision (P20). Further experiments on the quantisation effect in the scale factor are available in the supplementary material. An example of severe occlusion that occurs in the complex *bear_front* sequence is a good demonstration of the superiority of the DS-KCF tracker against all other methods, and in particular KCF itself when using DS-KCF with the *hog_depth* feature (with AUC of 81.9%). Figure 4 shows example scale and ooclusion comparisons.

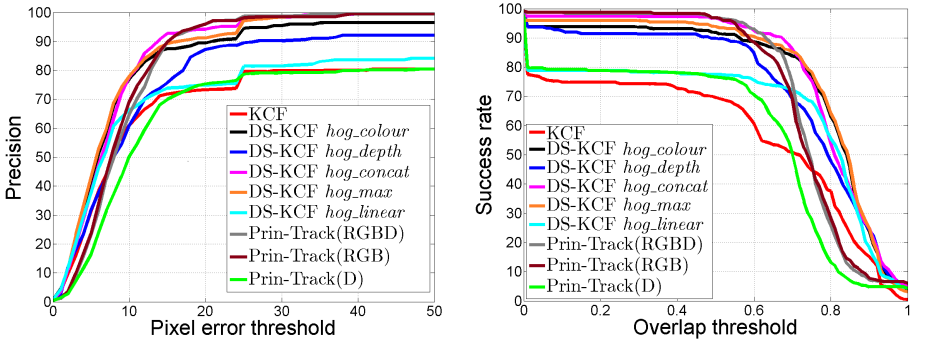

Figure 3: Average precision plot (left) and average success plot (right) for Princeton Dataset [18].

Table 1: Trackers' performance on Princeton Dataset [18]. All measures are in percentages, except for the frames per second (fps) columns. OAPF results are reproduced from [16].

	Sequences																	
	'bear_front'			'child_no1'			'zcup_move_1'			'face_occ5'			'new_ex_occ4'			Average		
	AUC	P20	fps	AUC	P20	fps	AUC	P20	fps	AUC	P20	fps	AUC	P20	fps	AUC	P20	fps
KCF [9]	18.6	19.8	**117**	66.3	**96.8**	**55**	72.5	**100.0**	**164**	79.7	93.1	**88**	46.1	56.8	**92**	56.6	73.3	**103**
DS-KCF *hog_colour*	73.6	82.4	64	**83.2**	93.7	33	82.3	**100.0**	105	69.9	80.7	38	78.1	90.2	61	77.4	89.4	60
DS-KCF *hog_depth*	**81.9**	91.3	54	69.0	91.3	34	78.8	**100.0**	111	82.2	95.1	51	53.5	58.5	33	73.1	87.2	57
DS-KCF *hog_concat*	75.7	85.3	42	76.5	92.1	19	81.5	**100.0**	81	**85.5**	**98.6**	22	**78.3**	**95.1**	38	**79.5**	94.2	40
DS-KCF *hog_max*	72.1	81.3	45	83.1	92.1	20	82.3	**100.0**	74	79.4	92.2	37	78.1	90.2	40	79.0	91.2	43
DS-KCF *hog_linear*	67.7	74.2	35	82.6	92.1	20	**82.5**	**100.0**	75	85.0	**98.6**	26	10.5	9.1	67	65.6	74.8	45
Prin-Track(RGB-D) [18]	78.9	**96.2**	0.08	71.4	**96.8**	0.14	78.3	**100.0**	0.19	70.1	96.8	0.17	71.9	90.2	0.10	74.1	96.0	0.14
Prin-Track(RGB) [18]	80.6	94.9	0.09	71.8	**96.8**	0.19	78.9	**100.0**	0.24	68.1	98.2	0.15	68.7	90.2	0.12	73.6	**96.0**	0.16
Prin-Track(D) [18]	69.9	81.6	0.09	15.9	21.4	0.18	73.7	**100.0**	0.24	54.1	81.0	0.24	72.6	92.7	0.12	57.2	75.3	0.18
OAPF [16]	78.9	-	0.60	77.2	-	1.20	72.7	-	1.00	79.2	-	1.30	74.6	-	0.50	76.5	-	0.90

The results for our DS-KCF tracker show that *hog_concat* is the most optimum feature on average with AUC of 79.5% and P20 of 94.2%, but clearly its inherent makeup means it is the slowest of our features - and yet at 40fps, it is still real-time. DS-KCF also outperforms the other two RGB-D trackers tested on this dataset, Prin-Track [18] (RGB, or D, or RGB-D) and OAPF [16]. Furthermore, the average processing rate in the Prin-Track (RGB-D) is

0.14fps and 0.9fps for the OAPF tracker in striking contrast to 40fps for DS-KCF. Qualitative examples of the trackers performance, for the sequences 'child_no1' and 'face_occ5', are shown in Figure 4 (c) and (e). Note how DS-KCF obtains a more accurate bounding box (red) producing a bigger overlap with the groundtruth (green). The 'face_occ5' example in 4(f) is particularly important as it clearly illustrates how the proposed solution is able to produce better tracking after occlusion.

On the Princeton Dataset test sequences, DS-KCF has ranked 3^{rd} amongst 20 different state-of-the-art RGB and RGB-D trackers[3].

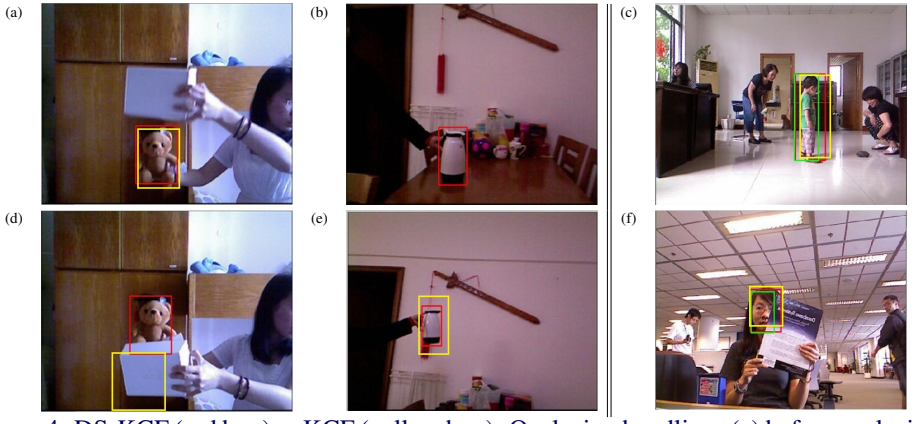

Figure 4: DS-KCF (red box) v. KCF (yellow box). Occlusion handling: (a) before occlusion, and (d) after occlusion. Change of scale: (b) initial target, and (e) target after change of scale. DS-KCF (red) v. Prin-Track(RGBD) (yellow) v. groundtruth (green) on (c) 'child_no1', and (f) 'face_occ5'.

Results with BoBoT-D [6] - For this dataset we report the trackers' performance (see Table 2) considering the mean overlap (Ov. in Table 2) between tracker output and groundtruth across each sequence and by considering the Hit measure (percentage of frame where the overlap is greater than 33%) as proposed in [6]. Again we demonstrate that (in this data set too) DS-KCF provides the better results when the feature *hog_concat* is used, providing the optimum balance between accuracy and real-time performance.

KCF is the fastest method at 141fps but it leads to less accurate results on average (overlap 32.5%, hit 47.7%) when compared with the best DS-KCF configuration (overlap 53.6%, hit 76.2%) at 30fps. The results obtained with DS-KCF with the feature set *hog_concat* show that for sequences 'Milk' and 'Person' it outperforms the state-of-the-art trackers Prin-Track and Adapt3D. On the other hand, for the 'Tank' sequence the performance of DS-KCF drops, mainly due to the variable shape and appearance of the tracked tank across the sequence (see Figure 5) when the small object can affect the quality of the depth segmentation. The measures for the 'Ball' sequence are very poor for both KCF and DS-KCF due to the speed of the ball and fast motion of the camera. Overall, the results for the BoBot-D dataset show that on average the proposed DS-KCF tracker performs second best to the Adapt3D tracker [6], but is ahead of all the other trackers.

[3]Ranks available at http://vision.princeton.edu/projects/2013/tracking/eval.php

Table 2: Average results for RGB-D trackers on the BoBot-D dataset. The best results in each column are in bold and the best DS-KCF results with *hog_concat* is highlighted.

	BoBot-D dataset Sequences												
	'Milk'		'Ball'		'Tank'		'Person'		'Box'		Average		
	Ov.	Hit	Ov.	Hit	Ov.	Hit	Ov.	Hit	Ov.	Hit	Ov.	Hit	fps
KCF [8]	58.1	82.2	2.0	2.4	13.1	10.8	31.4	43.0	57.9	**100.0**	32.5	47.7	**141**
DS-KCF *hog_colour*	74.4	**100.0**	9.1	10.3	26.4	42.1	64.8	98.3	63.9	**100.0**	47.7	70.1	61
DS-KCF *hog_depth*	45.5	60.9	16.6	23.0	15.2	18.9	**81.3**	**98.5**	66.2	**100.0**	45.0	60.3	64
DS-KCF *hog_concat*	**76.6**	**100.0**	7.9	10.3	43.9	72.5	73.4	98.0	66.1	**100.0**	53.6	76.2	30
DS-KCF *hog_max*	43.7	61.5	9.1	10.3	17.2	23.4	62.7	95.4	19.9	21.9	30.5	42.5	39
DS-KCF *hog_linear*	45.4	61.4	9.4	10.6	17.1	18.9	68.9	98.4	19.8	21.9	32.1	42.2	56
Prin-Track(RGB-D) [13]	64.5	83.5	27.1	39.3	7.6	8.3	66.9	94.1	66.9	**100.0**	46.6	65.0	0.16
Prin-Track(RGB) [13]	41.9	52.5	9.5	12.7	11.1	15.3	67.0	86.5	58.4	98.1	37.6	53.0	0.20
Prin-Track(D) [13]	48.2	64.8	26.9	36.9	10.6	11.8	65.6	93.8	65.2	98.9	43.3	61.2	0.27
Adapt3D [8]	73.5	96.8	**69.8**	**96.9**	**55.3**	**94.1**	70.7	95.3	**73.1**	99.8	**68.5**	**96.6**	30.60

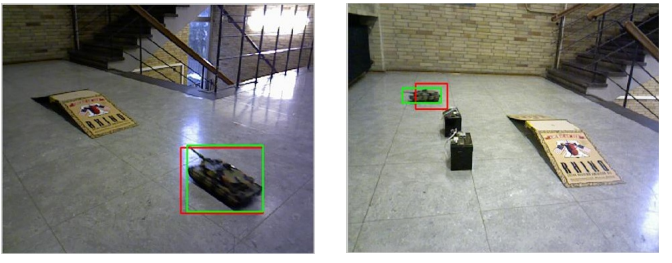

Figure 5: DS-KCF (red box) and groundtruth (green box) for the 'Tank' sequence.

5 Conclusions

We presented the DS-KCF tracker and demonstrated its performance against state-of-the-art RGB-D object trackers. DS-KCF provides a viable solution that combines precision with real-time performance, while improving on scale and occlusion handling. The method is either comparable or outperforms other trackers on two publicly available datasets for rigid and some deformable object categories. Possible future directions are multiple object tracking and handling highly deformable objects. We are making our code available to other researchers to encourage future use, comparison, and improvements.

Acknowledgements

This work was performed in the SPHERE IRC project funded by the UK Engineering and Physical Sciences Research Council (EPSRC), Grant EP/K031910/1.

References

[1] S. Avidan. Support vector tracking. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 26(8):1064–1072, 2004.

[2] D.S. Bolme, J.R. Beveridge, B.A. Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In CVPR, pages 2544–2550, 2010.

[3] M. Camplani, T. Mantecon, and L. Salgado. Depth-color fusion strategy for 3-D scene modeling with Kinect. Cybernetics, IEEE Transactions on, 43(6):1560–1571, 2013.

[4] T. Chen, Y. Chen, and S Chien. Fast image segmentation based on K-means clustering with histograms in HSV colour space. In Multimedia Signal Processing, 2008 IEEE 10th Workshop on, pages 322–325, 2008.

[5] H.K. Galoogahi, T. Sim, and S. Lucey. Multi-channel correlation filters. In ICCV, pages 3072–3079, 2013.

[6] G. García, D. Klein, J. Stückler, S. Frintrop, and A. Cremers. Adaptive multi-cue 3D tracking of arbitrary objects. In Pattern Recognition, pages 357–366. 2012.

[7] R.C. Gonzalez and R.E. Woods. Digital Image Processing. Addison-Wesley Longman Publishing Co., Inc., 1992.

[8] S. Hare, A. Saffari, and P.H.S. Torr. Struck: Structured output tracking with kernels. In ICCV, pages 263–270, 2011.

[9] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2015.

[10] Charles F. Hester and David Casasent. Multivariant technique for multiclass pattern recognition. Applied Optics, 19:1758–1761, 1980. doi: 10.1364/AO.19.001758.

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

[12] K. Khoshelham and S.O. Elberink. Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. Sensors, 12(2):1437–1454, 2012.

[13] D.A. Klein and A.B. Cremers. Boosting scalable gradient features for adaptive real-time tracking. In ICRA, pages 4411–4416, 2011.

[14] J. Kwon and K.M. Lee. Visual tracking decomposition. In CVPR, pages 1269–1276, 2010.

[15] Yang Li and Jianke Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In Computer Vision - ECCV 2014 Workshops, volume 8926, pages 254–265. 2015.

[16] T. Meshgi, S. Maeda, S. Oba, H. Skibbe, Y. Li, and S. Ishii. Occlusion aware particle filter tracker to handle complex and persistent occlusions. Computer Vision and Image Understanding, 2015 to appear.

[17] B. Scholkopf and A.J. Smola. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, 2001.

[18] S. Song and J. Xiao. Tracking revisited using RGBD camera: Unified benchmark and baselines. In ICCV, pages 233–240, 2013.

[19] Q. Wang, J. Fang, and Y. Yuan. Multi-cue based tracking. Neurocomputing, 131(0): 227 – 236, 2014.

[20] Y. Wu, J. Lim, and M. Yang. Online Object Tracking: A Benchmark. In CVPR, pages 2411–2418, 2013.