HIGH-SPEED AND ACCURATE SCALE ESTIMATION FOR VISUAL TRACKING WITH GAUSSIAN PROCESS REGRESSION

Linyu Zheng[*], *Ming Tang*[*], *Yingying Chen*[*], *Jinqiao Wang*[*] *and Hanqing Lu*[*]

[*]National Laboratory of Pattern Recognition, Institute of Automation, CAS, Beijing, 100190, China
[*]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China
[*]University of Chinese Academy of Sciences, Beijing, 100049, China
{linyu.zheng, tangm, yingying.chen, jqwang, luhq}@nlpr.ia.ac.cn

ABSTRACT

Recent years have seen remarkable progress in the visual tracking domain. However, it remains a challenging task to estimate the scale of target efficiently and accurately. In this paper, we present a novel and high-performance scale estimation approach for tracking-by-detection framework. The proposed approach, named GPAS, formulates the scale estimation as a Gaussian process regression problem based on scale pyramid representation. In general, it enjoys the following there advantages. (i) Efficient. It only takes 2ms to estimate the scale of a target on a single CPU. (ii) Accurate. Without bells and whistles, its accuracy surpasses all previous hand-crafted features based scale estimation methods by large margins. (iii) Generic. It can be incorporated into any tracking-by-detection framework based trackers easily. Experiment results show that compared to the latest and classical scale estimation method, fDSST, our GPAS significantly improves the performance by 6.2% in mean distance precision, 8.9% in mean overlap precision, and 5.5% in mean AUC on 28 sequences of OTB2013 with significant scale variations.

Index Terms— Visual Tracking, Scale Estimation, Gaussian Process Regression

1. INTRODUCTION

Visual object tracking is one of the fundamental problems in computer vision with many applications. In generic visual tracking, the goal is to estimate the state (*e.g.*, position and size) of the target in a whole image sequence only with its initial state [1]. This problem is challenging due to several interferences, such as large appearance change, scale variation, fast motion, and background clutter.

In recent years, tracking-by-detection framework based trackers have shown top performance [2, 3, 4, 5, 6]. They mainly train the appearance models, *i.e.*, locator [1], of the target by using discriminative methods in the current frame, and then locate the target in the next frame. However, it seems that there are few studies working on the scale estimation of target. From a technical standpoint, under hand-crafted features, existing scale estimation methods roughly fall into two categories. One category is to perform an exhaustive scale search by evaluating the locator at multi-levels scale pyramid. Typically, SAMF [7] belongs to this category. However, this brute-force search strategy is computationally demanding because the time complexity of its detection process is N times that of no scale estimation one, where N is the level of scale pyramid. Therefore, it struggles when encountered with large scale variations [8]. Another category is to train an efficient scale estimation model independent of the locator. Typically, DSST [9] belongs to this category. Specifically, it follows the procedure of first locating the target by locator and then estimating its size by scale estimation model. The advantage of this category is that efficient scale estimation can be done even in large scale space (*i.e.* N is large). The novel scale

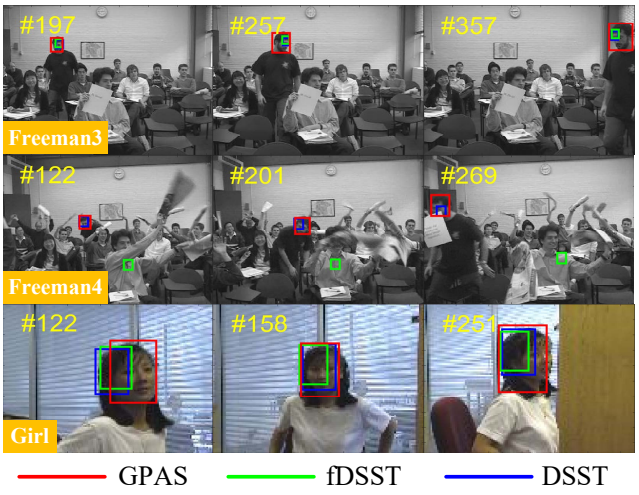

Fig. 1. Qualitative comparisons of the proposed GPAS with DSST and fDSST on three sequences of OTB-2013 with significant scale variations. Our GPAS is superior to DSST and fDSST. Best viewed in color.

[1]In this paper, we call the tracking model, which is mainly used to estimate the position of the target rather than the size of it, as locator.

978-1-7281-1331-9/20/$31.00 ©2020 IEEE

estimation approach proposed in this paper is based on this idea and is a technical innovation for DSST.

Inspired by the correlation filter based trackers (CF trackers) [2], DSST [9] and its improved version fDSST [8] efficiently train their scale estimation modules in frequency domain. Despite their excellent performance, we found that they mainly have the following two shortcomings. (1) They can not employ the kernel trick to improve their scale estimation accuracy further. (2) In order to be efficient, just like in CF trackers, they inevitably suffer from the boundary effect [2] which degrades their scale estimation accuracy seriously.

To solve the above two problems, in this paper, we propose a novel scale estimation approach, named GPAS, where Gaussian process regression (GPR) [10, 11] is applied to advance the scale estimation accuracy of DSST and fDSST. In our GPAS, the scale estimation is formulated as a GPR problem based on scale pyramid representation. Specifically, in the current frame, we construct a Gaussian covariance matrix by using the samples of the target at a set of different scales in training. Afterwards, given a new frame in detection, we first locate the target by locator, then employ the constructed GPR model to predict the regression values of the test samples at a set of different scales and obtain the accurate estimation of the target scale. By doing so, our GPAS does not suffer from the boundary effect and it can also improve the scale estimation accuracy of DSST and fDSST by utilizing the kernel trick (see Fig. 1). Additionally, considering the importance of model update along with its efficiency in visual tracking, we employ a simple yet effective update method to efficiently fit our GPAS to the variation of the target in online tracking.

Extensive experiments are performed on three public benchmarks, OTB2013, OTB2015, and VOT2018. To ensure fair comparisons with DSST and fDSST, we maintain the locators of DSST, fDSST, and ECO-HC unchanged and replace their scale estimation modules with our proposed GPAS, then obtain three new trackers denoted by GPAS, fGPAS, and eGPAS, respectively. On OTB2013, our GPAS and fGPAS outperform DSST and fDSST in both tracking accuracy and speed, respectively. Especially, on 28 sequences of OTB2013 with significant scale variations, our GPAS and fGPAS significantly improve the performance of DSST and fDSST by 2.8% and 6.2% in mean distance precision, 8.1% and 8.9% in mean overlap precision, and 3.4% and 5.5% in mean AUC, respectively. On OTB2015 and VOT2018, our eGPAS also outperforms ECO-HC on both accuracy and speed with larger margins. It is worth mentioning that our fGPAS can run beyond 100 FPS on a single CPU. Therefore, we believe that our proposed scale estimation approach which is accurate and efficient can be of interest for many tracking research efforts.

2. GAUSSIAN PROCESS REGRESSION

Traditionally parametric models can be used for regression and classification problems and they have an advantage in ease of interpretability. However, for complex datasets, simple parametric models may lack expressive power, and even though their more complex counterparts may have a stronger expressive power, they may not easily work in practice. To solve the problems above, the advent of Gaussian process regression (GPR) has opened the possibility of flexible models which are practical to work for complex datasets. For detailed introductions to GPR, we suggest readers referring to [10, 11].

Given a training set $\mathbb{D} = \{(\mathbf{x}_i, y_i) \mid i = 1, ..., N\}$, where $\mathbf{x}_i \in \mathbb{R}^D$ is the vector of training sample data and $y_i \in \mathbb{R}$ is the desired regression value with respect to $\mathbf{x}_i$. The main assumption of GPR is $y = f(\mathbf{x}) + \epsilon$ where the output y is considered as the point sampled from a multivariate Gaussian distribution and $\epsilon \sim N\left(0, \sigma_n^2\right)$ represents the noise in training outputs. Moreover, it can be assumed that this Gaussian distribution has the mean of zeros and a commonly used covariance function is the Radial Basis Function (RBF) with additive white noise which can be expressed as

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma_l^2}\right) + \sigma_n^2 \delta(\mathbf{x}, \mathbf{x}'), \quad (1)$$

where the hyperparameter σ_f governs the magnitude of covariance, σ_l modulates the exponential decay of covariance, and σ_n describes additive white noise.

From the GPR theory, the reliability of regression is dependent on how well the covariance function is selected. Therefore, it is necessary to estimate the values of the hyperparameters in kernel $\kappa(\cdot, \cdot)$. It is easy to know that the optimal hyper-parameters of RBF corresponding to maximize

$$\log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}| - \frac{n}{2}\log 2\pi, \quad (2)$$

where $\boldsymbol{\theta} = \{l, \sigma_f, \sigma_n\}$, $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$, $\mathbf{K}$ is the kernel matrix with respect to $\mathbf{X}$, and $\mathbf{y} = \{y_1, ..., y_n\}$.

Based on above, it is necessary to construct the following three covariance matrices with the training set $\mathbf{X}$ when we would like to estimate the regression value of test sample.

$$\mathbf{K} = \begin{bmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \kappa(\mathbf{x}_1, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_n) \\ \kappa(\mathbf{x}_2, \mathbf{x}_1) & \kappa(\mathbf{x}_2, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_n, \mathbf{x}_1) & \kappa(\mathbf{x}_n, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}, \quad (3a)$$

$$\mathbf{K}_* = [\kappa(\mathbf{x}_*, \mathbf{x}_1), \kappa(\mathbf{x}_*, \mathbf{x}_2), \cdots, \kappa(\mathbf{x}_*, \mathbf{x}_n)], \quad (3b)$$

$$\mathbf{K}_{**} = [\kappa(\mathbf{x}_*, \mathbf{x}_*)], \quad (3c)$$

where $\mathbf{x}_* \in \mathbb{R}^D$ is the vector of test sample data. Consequently, the probability of output y_* with respect to the test sample $\mathbf{x}_*$ follows the Gaussian distribution

$$y_* \mid \mathbf{y} \sim N\left(\mathbf{K}_*\mathbf{K}^{-1}\mathbf{y}, \mathbf{K}_{**} - \mathbf{K}_*\mathbf{K}^{-1}\mathbf{K}_*^\top\right). \quad (4)$$

Namely, the mean and variance of y_* can be formulated as

$$\bar{y}_* = \mathbf{K}_*\mathbf{K}^{-1}\mathbf{y}, \quad (5a)$$

$$var(y_*) = \mathbf{K}_{**} - \mathbf{K}_*\mathbf{K}^{-1}\mathbf{K}_*^\top, \quad (5b)$$

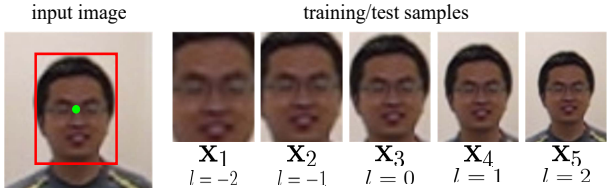

Fig. 2. Illustration of the sampling method in our GPAS. The size and center position (green point) of the target box (red box) are (w, h) and (x, y), respectively. After multi-scales sampling, all samples are resized to (W, H). Here, $N = 5$ and $a = 1.2$. See text for details.

where $\bar{y}_*$ and $var\,(y_*)$ are the output expectation and variance of the test sample $\mathbf{x}_*$, respectively.

3. ACCURATE SCALE ESTIMATION WITH GAUSSIAN PROGRESS REGRESSION

In this section, we will introduce our proposed scale estimation framework including sampling, online learning, update, and detection. Here, we predefine the $\mathbf{K}_{\mathbf{X}_i\mathbf{X}_j}$ as follows:

$$\mathbf{K}_{\mathbf{X}_i\mathbf{X}_j} = \begin{bmatrix} \kappa(\mathbf{x}_{i1}, \mathbf{x}_{j1}) & \kappa(\mathbf{x}_{i1}, \mathbf{x}_{j2}) & \cdots & \kappa(\mathbf{x}_{i1}, \mathbf{x}_{jN}) \\ \kappa(\mathbf{x}_{i2}, \mathbf{x}_{j1}) & \kappa(\mathbf{x}_{i2}, \mathbf{x}_{j2}) & \cdots & \kappa(\mathbf{x}_{i2}, \mathbf{x}_{jN}) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_{iN}, \mathbf{x}_{j1}) & \kappa(\mathbf{x}_{iN}, \mathbf{x}_{j2}) & \cdots & \kappa(\mathbf{x}_{iN}, \mathbf{x}_{jN}) \end{bmatrix}, \tag{6}$$

where $\mathbf{X}_i$ and $\mathbf{X}_j$ come from different frames when $i \neq j$ or same frame when $i = j$, $\mathbf{x}_{in}$ represents the n-th sample in $\mathbf{X}_i$, and N denotes the level of scale pyramid used in the sampling process of both online learning and detection. $\mathbf{K}_{\mathbf{X}_i\mathbf{X}_j}$ represents the constructed covariance matrix with respect to $\mathbf{X}_i$ and $\mathbf{X}_j$, and it is often used in this section.

Sampling. Our sampling method is based on scale pyramid representation. Specifically, given the target's center position (x, y) and its size (w, h), we generate each $\mathbf{x}_i$ in $\mathbf{X}$ by extracting the image patch of size $a^l \times (w, h)$ centered around (x, y) and resize it to (W, H), where $l = i - \lfloor \frac{N+1}{2} \rfloor$, a denotes the scale factor between adjacent level of scale pyramid, and (W, H) is the predefined uniform size of all samples. Fig. 2 illustrates our sampling method vividly.

Online Learning. Given the target's center position and its size in i-th frame, we first generate the training set $\mathbf{X}_i$ based on the above sampling process. Then, the covariance matrix $\mathbf{K}_{\mathbf{X}_i\mathbf{X}_i}$ is constructed with $\mathbf{X}_i$ and Eq. 6. Finally, we create the Gaussian like label vector $\mathbf{y}$ as follows:

$$y_i = \exp\left(-\left(i - \lfloor (N+1)/2 \rfloor\right)^2 / 2\sigma^2 N\right) \tag{7}$$

where σ controls the magnitude of labels when N is fixed.

Online Update and Detection. Given the target's center position in j-th frame along with its size in $(j-1)$-th frame, we generate the test set $\mathbf{X}_j$ based on the above sampling process. Then, combining Eq. 5a and the popular linear weighted based prediction method in tracking [9, 2], it is easy to obtain the regression prediction formula of $\mathbf{X}_j$ as

$$\mathbf{f}(\mathbf{X}_j) = \sum_{i=1}^{j-1} \beta_i \mathbf{K}_{\mathbf{X}_j\mathbf{X}_i} \mathbf{K}^{-1}_{\mathbf{X}_i\mathbf{X}_i} \mathbf{y} \quad (j \geq 2) \tag{8}$$

where $\mathbf{X}_i$ $(i < j)$ represents the training set of previous i-th frame and β_i represents its weight.

However, we observe that in order to obtain $\mathbf{f}(\mathbf{X}_j)$ by Eq. 8, we have to construct $\mathbf{K}_{\mathbf{X}_j\mathbf{X}_i}$ $(j-1)$ times, and this is extremely time-consuming when j is large. Additionally, although we can store the $\mathbf{K}^{-1}_{\mathbf{X}_i\mathbf{X}_i}\mathbf{y}$ of each previous frame to save the calculation time of Eq. 8, this will require a lot of memory when j is large.

In order to apply our approach to practice tracking problem efficiently and reduce its memory requirement as much as possible, we use Eq. 9 to approximate Eq. 8 by considering that the change between adjacent frames is small in tracking.

$$\mathbf{f}(\mathbf{X}_j) = \mathbf{K}_{\mathbf{X}_j \sum_{i=1}^{j-1} \beta_i \mathbf{X}_i} \sum_{i=1}^{j-1} \mathbf{K}^{-1}_{\mathbf{X}_i\mathbf{X}_i} \mathbf{y} \quad (j \geq 2). \tag{9}$$

Further, based on Eq. 9, it is easy to obtain the detection and update scheme of our approach as follows:

$$\mathbf{f}(\mathbf{X}_j) = \mathbf{K}_{\mathbf{X}_j\mathbf{A}_{j-1}} \mathbf{B}_{j-1} \quad (t \geq 2) \tag{10a}$$

$$\mathbf{A}_j = (1-\delta)\,\mathbf{A}_{j-1} + \delta \mathbf{X}_j \tag{10b}$$

$$\mathbf{B}_j = (1-\delta)\,\mathbf{B}_{j-1} + \delta \mathbf{K}^{-1}_{\mathbf{X}_j\mathbf{X}_j} \mathbf{y} \tag{10c}$$

where δ represents the learning rate. Moreover, the relationship between $\{\beta_i \mid i = 1, ..., t-1\}$ and δ can be described as

$$\begin{aligned} &\sum_{i=1}^{j-1} \beta_i = 1 \\ &\beta_1 = (1-\delta)^{j-2} \\ &\beta_{i-1}/\beta_i = 1 - \delta \quad (i > 2) \end{aligned} \tag{11}$$

It is clear that after the above approximation and based on Eq. 10, we can efficiently calculate $\mathbf{f}(\mathbf{X}_j)$ even when j is large. In addition, it is worth noting that the update and detection scheme Eq. 10 allows the model to be updated without storing all the previous information and only the current model $\mathbf{A}_j$ and $\mathbf{B}_j$ need to be saved for predicting the next frame.

4. IMPLEMENTATION DETAILS

In order to convincingly verify that our proposed scale estimation method is superior to its baseline methods DSST [9] and fDSST [8] in both accuracy and speed, we ensure that the following three important implementation details which are irrelevant to the principle of our method are same as those of compared trackers, DSST [9], fDSST [8], and ECO-HC [6].

Table 1. Comparison with DSST, fDSST, and SAMF trackers on the 28 sequences of OTB2013 benchmark with significant scale variations. The best two results are shown in red and blue, respectively. Our fGPAS obtains the best results.

Tracker	fGPAS	GPAS	fDSST	DSST	SAMF
mean DP	0.820	0.753	0.758	0.725	0.726
mean OP	0.765	0.718	0.676	0.637	0.638
mean AUC	0.631	0.582	0.576	0.548	0.546
mean FPS	113	64	110	62	40

Locator. The basic locators of DSST, fDSST, and ECO-HC are different. Therefore, we maintain their locators unchanged and combine our scale estimation method with them, respectively. As a result, we obtain three new trackers named GPAS, fGPAS and eGPAS, respectively.

Feature. All the scale estimation modules of DSST, fDSST, and ECO-HC employ HOG [12] feature. Therefore, we employ HOG feature in GPAS, fGPAS and eGPAS. Note that there is no obvious obstacle to applying convolutional neural networks features in our approach and improving its accuracy further. However, the time-consuming features extraction will prevent it from achieving high speed on CPU.

Platform. All DSST, fDSST, and ECO-HC are implemented in MATLAB. Therefore, in order to compare the running speed fairly, our GPAS, fGPAS, and eGPAS are also implemented in MATLAB. In addition, all trackers compared in this paper are tested on the same single Intel-i7 CPU.

In addition to above, we show other two implementation details which are relevant to the principle of our method.

Kernel. We use single Gaussian kernel described in Eq. 1. In general, the parameter σ_l can be obtained by maximizing Eq. 2, but doing so online is extremely time-consuming. By experimenting on a large number of images, we observe that this parameter roughly presents a Gaussian distribution centered at 0.1. Therefore, we set it to 0.1 on all sequences for convenience. In addition, we set $\sigma_n = 0.1$, experimentally.

Parameters. In Sampling of Sec. 3, we set $N = 17$, $a = 1.02$, and $W \times H = 512$ while maintaining the aspect ratio of samples unchanged. We set the hyper-parameter $\sigma = 0.7$ in Eq. 7 and the learning rate $\delta = 0.016$ in Eq. 10.

5. EXPERIMENTS

First, following fDSST [8], we show the tracking performance of our GPAS and fGPAS on the public benchmark OTB2013 [1], then compare them with the trackers participating in the comparison in fDSST. Second, we show the tracking performance of our eGPAS on two public benchmarks: OTB2015 [13] and VOT2018 [14], then compare it with the state-of-the-art hand-crafted features based trackers.

5.1. Experiment on OTB2013

Following the standard benchmark protocols on the OTB-2013, all trackers are quantitatively evaluated by four standard evaluation metrics, namely distance precision (DP), overlap precision (OP), area-under-the-curve (AUC), and frames per second (FPS). The DP score is defined as the percentage of frames in a video where the Euclidean distance between the tracking output and ground truth centroid is smaller than 20. The OP score is computed as the percentage of frames in a video where the intersection-over-union overlap with the ground truth exceeds 0.5. The AUC is computed as the area under the success plot which measures the exhaustive overlap between the predicted bounding box and the ground truth.

Table 1 shows the comparison of our GPAS and fGPAS with classical scale estimation methods, DSST, fDSST, and SAMF on the 28 sequences of OTB-2013 benchmark with significant scale variations. Our fGPAS and GPAS improve the performance of their corresponding baseline trackers fDSST and DSST by 6.2% and 2.8% in mean distance precision, 8.9% and 8.1% in mean overlap precision, 5.5% and 3.4% in mean AUC, and 3 and 2 in mean FPS, respectively. In addition, they also outperform SAMF, which extends the KCF with multiple features and scale estimation using a multi-resolution translation filter approach, with large margins in both tracking accuracy and speed. These experimental results strongly confirm that our proposed scale estimation approach is more accurate and efficient than its counterparts, DSST, fDSST, and SAMF.

Table 2 provides a comprehensive comparison of our GPAS and fGPAS with eight hand-crafted features based trackers on the whole OTB2013 benchmark. Our fGPAS and GPAS obtain 81.0% and 75.3% in mean distance precision, 76.0% and 71.4% in mean overlap precision, 61.3% and 58.3% in mean AUC, and 112 and 64 in mean FPS, respectively, and outperform their corresponding baseline trackers fDSST and DSST in both tracking accuracy and speed, respectively. In addition, they also outperform other hand-crafted features based trackers with large margins.

It is worth noting that according to above experimental results, our fGPAS is consistently better than our GPAS in both accuracy and speed. This is because their basic locators are different, and the locator of fGPAS is more robust and efficient than that of GPAS. Furthermore, this is also the main reason why the tracking accuracy of our GPAS is inferior than that of fDSST on the whole OTB2013 benchmark.

5.2. Experiment on OTB2015

Table 3 shows the comprehensive comparison of our eGPAS with eight state-of-the-art hand-crafted features based trackers on OTB2015 benchmark. Our eGPAS obtains 79.0% in mean overlap precision, 65.2% in mean AUC, and 60 in mean FPS, outperforming its baseline tracker ECO-HC, where fDSST is employed for scale estimation, with 0.8%

Table 2. Comparison with eight hand-crafted features based trackers on OTB2013 benchmark. The best two results are shown in red and blue, respectively. Our fGPAS outperforms other trackers in both accuracy and speed.

Tracker	fGPAS	GPAS	fDSST	DSST	SAMF	KCF [2]	Struck [15]	ASLA [16]	SCM [17]	EDFT [18]
mean DP	0.810	0.753	0.799	0.736	0.777	0.740	0.687	0.592	0.563	0.567
mean OP	0.760	0.714	0.747	0.673	0.697	0.623	0.588	0.564	0.530	0.498
mean AUC	0.613	0.583	0.609	0.564	0.577	0.518	0.492	0.484	0.442	0.426
mean FPS	113	64	110	62	40	307	26	3	0.3	55

Table 3. Comparison with eight state-of-the-art hand-crafted features based trackers on OTB2015 benchmark. The best three results are shown in red, blue and magenta, respectively. Our eGPAS achieves the best trade-off between accuracy and speed.

Tracker	eGPAS	MKCFup [19]	STRCF [20]	ECO-HC [6]	CSRDCF [21]	DLSSVM [22]	Staple [23]	BACF [4]	GPRT [24]
mean OP	0.790	0.687	0.796	0.782	0.705	0.661	0.691	0.776	0.792
mean AUC	0.652	0.581	0.651	0.648	0.587	0.563	0.581	0.631	0.655
mean FPS	60	150	24	60	15	6	70	35	6 (GPU)
where	this paper	CVPR2018	CVPR2018	CVPR2017	CVPR2017	CVPR2016	CVPR2016	ICCV2017	IJCAI2018

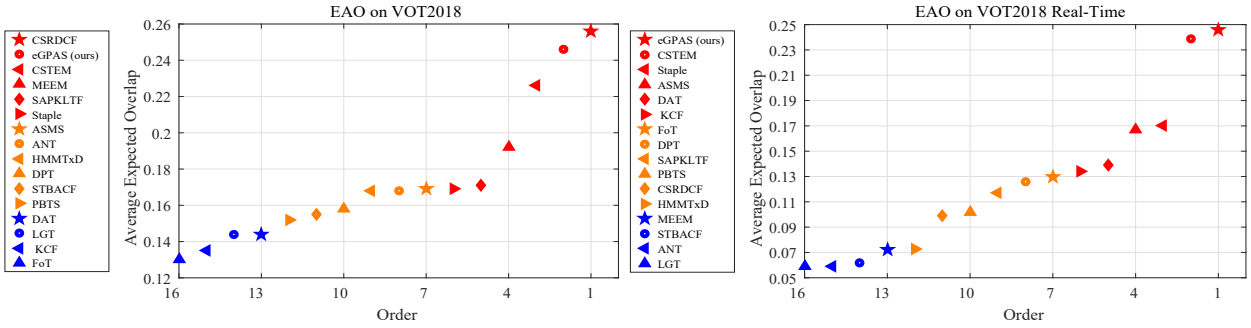

Fig. 3. Comparison with the top-15 hand-crafted features based trackers on VOT2018 and VOT2018 real-time challenges.

and 0.4% in mean overlap precision and AUC, respectively. Although GPRT achieves the best tracking accuracy, it can barely obtain the tracking speed of 6 FPS even with the help of GPU. In addition, although MKCFup achieves the fastest tracking speed, its tracking accuracy is relatively worse than others. Thus, we reasonably believe that our eGPAS achieves the best trade-off between accuracy and speed.

5.3. Experiment on VOT2018

Fig. 3 shows the comparison of our eGPAS with the top-15 hand-crafted features based trackers on VOT2018 and VOT2018 real-time challenges where expected average overlap (EAO) is employed to quantitatively evaluated all trackers. Our eGPAS obtains the EAO score of 0.246 on both challenges. Although CSRDCF achieves slightly higher accuracy than eGPAS on VOT2018 challenge, its real-time performance is obviously worse than eGPAS's. Additionally, the accuracy of eGPAS outperforms other state-of-the-art hand-crafted features based trackers with large margins on both VOT2018 and VOT2018 real-time challenges. Therefore, we reasonably believe that our eGPAS achieves the best trade-off between accuracy and speed.

6. CONCLUSION

In this paper, we formulate the target scale estimation as a Gaussian process regression problem, and propose a novel scale estimation approach, GPAS, for visual tracking. Compared to an exhaustive scale space search scheme, SAMF, GPAS improves tracking accuracy while being computationally efficient. Compared to modeling method based on scale space, DSST, GPAS improves tracking accuracy by utilizing the kernel trick and removing the boundary effect. Both quantitative and qualitative evaluations of experiments clearly demonstrate that GPAS outperforms its baseline methods, while operating at high-speed. Moreover, it can be incorporated into any tracking-by-detection framework based trackers easily. Therefore, we believe that our GPAS can be of interest for many tracking research efforts.

Acknowledgement. This work was supported by National Natural Science Foundation of China under Grants 61976210, 61876086, 61806200, and 61772527. This work was also supported by the Research and Development Projects in the Key Areas of Guangdong Province (No.2019B010153001).

7. REFERENCES

[1] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang, "Online object tracking: A benchmark," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2411–2418.

[2] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista, "High-speed tracking with kernelized correlation filters," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 3, pp. 583–596, 2014.

[3] Linyu Zheng, Ming Tang, Yingying Chen, Jinqiao Wang, and Hanqing Lu, "Fast-deepkcf without boundary effect," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[4] Hamed Kiani Galoogahi, Ashton Fagg, and Simon Lucey, "Learning background-aware correlation filters for visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1135–1143.

[5] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4310–4318.

[6] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg, "Eco: efficient convolution operators for tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6638–6646.

[7] Yang Li and Jianke Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *European conference on computer vision*. Springer, 2014, pp. 254–265.

[8] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg, "Discriminative scale space tracking," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 8, pp. 1561–1575, 2016.

[9] Martin Danelljan, Gustav Häger, Fahad Khan, and Michael Felsberg, "Accurate scale estimation for robust visual tracking," in *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press, 2014.

[10] Matthias Seeger, "Gaussian processes for machine learning," *International journal of neural systems*, vol. 14, no. 02, pp. 69–106, 2004.

[11] Carl Edward Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.

[12] Navneet Dalal and Bill Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. IEEE, 2005, vol. 1, pp. 886–893.

[13] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015.

[14] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Cehovin Zajc, Tomas Vojir, Goutam Bhat, Alan Lukezic, Abdelrahman Eldesokey, et al., "The sixth visual object tracking vot2018 challenge results," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[15] Sam Hare, Stuart Golodetz, Amir Saffari, Vibhav Vineet, Ming-Ming Cheng, Stephen L Hicks, and Philip HS Torr, "Struck: Structured output tracking with kernels," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2096–2109, 2015.

[16] Xu Jia, Huchuan Lu, and Ming-Hsuan Yang, "Visual tracking via adaptive structural local sparse appearance model," in *2012 IEEE Conference on computer vision and pattern recognition*. IEEE, 2012, pp. 1822–1829.

[17] Wei Zhong, Huchuan Lu, and Ming-Hsuan Yang, "Robust object tracking via sparsity-based collaborative model," in *2012 IEEE Conference on Computer vision and pattern recognition*. IEEE, 2012, pp. 1838–1845.

[18] Michael Felsberg, "Enhanced distribution field tracking using channel representations," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2013, pp. 121–128.

[19] Ming Tang, Bin Yu, Fan Zhang, and Jinqiao Wang, "High-speed tracking with multi-kernel correlation filters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4874–4883.

[20] Feng Li, Cheng Tian, Wangmeng Zuo, Lei Zhang, and Ming-Hsuan Yang, "Learning spatial-temporal regularized correlation filters for visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4904–4913.

[21] Alan Lukežič, Tom'aš Voj'iř, Luka Čehovin Zajc, Jiř'i Matas, and Matej Kristan, "Discriminative correlation filter tracker with channel and spatial reliability," *International Journal of Computer Vision*, 2018.

[22] Jifeng Ning, Jimei Yang, Shaojie Jiang, Lei Zhang, and Ming-Hsuan Yang, "Object tracking via dual linear structured svm and explicit feature map," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4266–4274.

[23] Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip HS Torr, "Staple: Complementary learners for real-time tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1401–1409.

[24] Linyu Zheng, Ming Tang, and Jinqiao Wang, "Learning robust gaussian process regression for visual tracking.," in *IJCAI*, 2018, pp. 1219–1225.