Learning Robust Gaussian Process Regression for Visual Tracking

Linyu Zheng^{1,2*}, **Ming Tang**^{1,2} and **Jinqiao Wang**^{1,2}
^{1}University of Chinese Academy of Sciences, Beijing, China
^{2}National Lab of Pattern Recognition, Institute of Automation, CAS, Beijing 100190, China
{linyu.zheng, tangm, jqwang}@nlpr.ia.ac.cn

Abstract

Recent developments of Correlation Filter based trackers (CF trackers) have attracted much attention because of their top performance. However, the boundary effect imposed by the basic periodic assumption in their fast optimization seriously degrades the performance of CF trackers. Although there existed many recent works to relax the boundary effect in CF trackers, the cost was that they can not utilize the kernel trick to improve the accuracy further. In this paper, we propose a novel Gaussian Process Regression based tracker (GPRT) which is a conceptually natural tracking approach. Compared to all the existing CF trackers, the boundary effect is eliminated thoroughly and the kernel trick can be employed in our GPRT. In addition, we present two efficient and effective update methods for our GPRT. Experiments are performed on two public datasets: OTB-2013 and OTB-2015. Without bells and whistles, on these two datasets, our GPRT obtains 84.1% and 79.2% in mean overlap precision, respectively, outperforming all the existing trackers with hand-crafted features.

1 Introduction

Visual object tracking is one of the fundamental problems in computer vision with many applications. In generic visual object tracking, given the initial state (e.g., position and size) of a target object in the first frame, the task is to estimate the states of the target in the subsequent frames. It is commonly known that despite significant progresses in recent decades, visual object tracking is still a challenging problem due to some extremely challenging factors (e.g. large appearance changes, occlusions, background clutters and fast motion) and very limited sample datas [Wu *et al.*, 2015]. Therefore, it is crucial to research how to construct a robust appearance model from very limited samples to distinguish target from background in visual tracking.

Recently, Correlation Filter based trackers (CF trackers) have shown high accuracy and robustness when tracking different types of targets. In CF trackers, they mainly learn the filter from circular samples to regress a gaussian response map in fourier domain, and the process of learning and target detection can be accelerated by using the Convolution Theorem and Fast Fourier Transform (FFT). [Bolme *et al.*, 2010] proposed to learn a Minimum Output Sum of Squared Error (MOSSE) filter for visual tracking on gray-scale images. They used a base image patch and the circulant virtual ones to train the filter directly in the Fourier domain. [Henriques *et al.*, 2015] proposed Kernelized Correlation Filter (KCF) which not only expended the MOSSE tracker to learn multi-channel filters on multi-dimensional features, but also took advantage of the kernel trick. Compared to MOSSE, the tracking performance had been greatly improved by KCF when using the HOG [Dalal and Triggs, 2005] feature and gaussian kernel on OTB-2013 [Wu *et al.*, 2013]. However, the basic periodic assumption leads to an inaccurate representation of the real samples, and produces unwanted boundary effects in CF trackers. This problem obviously reduces the discriminative power of the learned model and severely degrades the performance of standard CF trackers. In order to relax the boundary effect, [Kiani Galoogahi *et al.*, 2015] and [Danelljan *et al.*, 2015] respectively proposed Correlation Filters with Limited Boundaries (CFLB) and Spatially Regularized Discriminatively Correlation Filters (SRDCF). In CFLB, Galoogahi et al. added a rectangular window to the circular samples in CF trackers, and then used the Alternating Direction Method of Multipliers (ADMM) algorithm to solve a objective optimization problem with equality constraints. In SRDCF, Danelljan et al. added a spatial regularization component in the standard CF formulation to penalize the correlation filter coefficients depending on their spatial location in the learning process. Even though the CFLB and SRDCF relaxed the boundary effect in CF trackers, they still can not solve the problem thoroughly. Furthermore, CFLB and SRDCF can not use the kernel trick to improve the accuracy further [Henriques *et al.*, 2015], because there is no way to calculate the value of the kernel after they add a window to sample space or filter space.

In this paper, to solve the problems in CF trackers, we apply the Gaussian Process Regression (GPR) [Rasmussen and Williams, 2006] to visual tracking and propose a novel GPR based tracker (GPRT) which is a conceptually natural tracking approach. In our GPRT, following the basic GPR theory, we construct gaussian covariance matrix using training sam-

[*]This work was supported by National Natural Science Foundation of China under Grants 61772527 and 61375035.

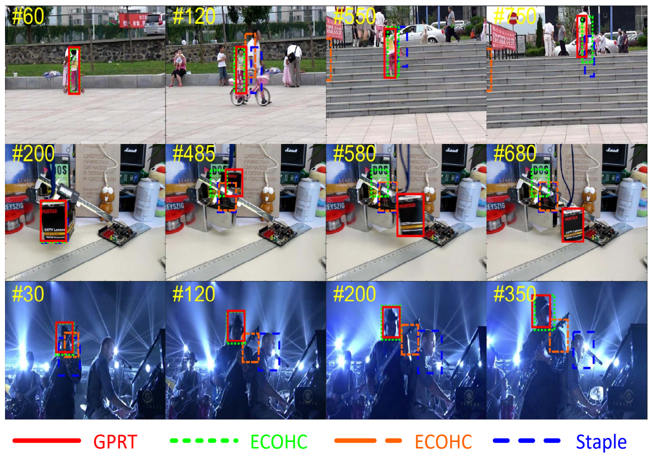

Figure 1: Qualitative comparisons of our GPRT with the other state-of-the-art trackers, ECOHC, SRDCF, and Staple on 3 sequences of OTB-2015. In all three cases, CF trackers suffer from the boundary effect or can only use the linear discriminator with low discriminant. These issues cause inaccurate target location in cases of occluded by similar objects (top row), complex background (middle row) and blur foreground (bottom row). However, our GPRT successfully tackles these situations and lead to better performance.

ples and employ the constructed GPR model to predict the regression values of the test samples in the detection process. Through above, our GPRT does not have the boundary effect and we can also promote the tracking performance by utilizing the kernel trick (see Figure. 1). For the sake of simplicity, in experiments we only use single gaussian kernel with hand-crafted features (i.e. HOG [Dalal and Triggs, 2005] and Color-Names [Danelljan *et al.*, 2014b]). As for the parameters in gaussian kernel, it is time-consuming to solve the maximum likelihood to get the optimal parameters for each frame, and we found that there is no significant difference in tracking performance between only solving them in the first frame and just like the KCF to design them by the rule of thumb. Additionally, considering the importance of the update method in visual tracking, we present two different update methods which are efficient and effective for updating our GPRT in the tracking process. The one is based on the directly apply GPR to the tracking problem which mainly update the appearance model in order to make the detection process faster, and the other, which not only the appearance model but also the GPR model needed to be updated in tracking process, is based on the relationship between GPR and Kernel Ridge Regression (KRR) [Rasmussen and Williams, 2006]. It is worth mentioning that our GPRT can be expanded to multi-kernel, conveniently. Furthermore, convolutional neural networks (CNN) features can also accessibly be added to our approach, even though they may have different resolutions in different layers.

In summary, we make the following three contributions.

- We propose a novel and conceptually natural tracking approach, called GPRT, which mainly apply the GPR to object tracking.
- We present two different update methods for our GPRT, and these methods are proved to be effective and efficient in experiments.
- We perform our GPRT on two public datasets: OTB-2013 and OTB-2015, and our GPRT achieves state-of-the-art results. Without bells and whistles, on these two datesets, our GPRT obtains 84.1% and 79.2% in mean overlap precision, respectively, and outperforms all the existing trackers with hand-crafted features.

We will release code to facilitate future research.

2 Related Work

[Rasmussen and Williams, 2006] briefly introduced what GPR is and how to use it in practice to solve regression problem. Different from general tasks based on the regression problem, image processing, especially video analysis task, has the characteristics of needing to deal with a large number of high-dimensional data. There are many approximation schemes which rely on the use of a series of inducing points to reduce the high computational complexity when applying the GPR to large data [Quiñonero-Candela and Rasmussen, 2005]. [Titsias, 2009] suggested a variational approach which provides an objective function to optimize these inducing points. In addition, [Hensman *et al.*, 2013] proposed Stochastic Variational Inference (SVI) for GPR to extend the variational idea. They showed how the variational objective could be reformulated with additional parameters to enable stochastic optimization. Furthermore, they trained the GPR model using mini-batches which allowed them to learn from a large dateset which contains 700,000 samples. However, their method may not suitable for object tracking problem, because we need a simple and efficient online update method for object tracking in order to make the tracking model adapt to the change of the target appearance. Unlike [Hensman *et al.*, 2013], for applying GPR to object tracking, we propose two efficient and effective methods to update the GPR model, and after effective update we can gradually obtain better generalization ability of the GPR model through a large number of data in online tracking process.

In recent years, there are a few researches applying the GPR to the visual tracking problem. [Gao *et al.*, 2014] proposed the TGPR which directly analyzes the probability of target appearance using the GPR. They divided the labeled samples into auxiliary and target samples, and learned the observation model for regression in a semi-supervised fashion. Unlike TGPR, we apply all the known samples as many as possible to construct and update the GPR model, and our online learning and update method are effective and efficient. The experiments confirm that in the mean distance and overlap precision, our GPRT obviously exceeds (about 11% and 14%, respectively) the TGPR on OTB-2013 dataset [1].

3 Proposed Method

In this section, we firstly review the gaussian process regression, and secondly introduce our GPRT tracking framework including online learning, update and detection, finally we show how to accelerate calculation in our GPRT.

[1]The TGPR does not report its performance on OTB-2015 dataset, so here we only compared with it on OTB-2013 dataset.

3.1 Gaussian Process Regression

For a useful introduction to GPR, we suggest the reader referring to [Rasmussen and Williams, 2006; Ebden, 2015].

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 (such as neural networks) may have a stronger expressive power, they may not easily work in practice. In order to solve the problems above, the advent of GPR has opened the possibility of flexible models which are practical to work for complex datasets [Rasmussen and Williams, 2004].

Given a training set $\mathbb{D} = \{(\mathbf{x}_i, y_i) \mid i = 1, ..., n\}$, where $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \in \mathbb{R}^{D\times n}$ is the matrix of input data, $\mathbf{y} = \{y_1, ..., y_n\} \in \mathbb{R}^n$ is the vector of desired regression values with respect to $\mathbf{X}$ and D is the dimension of input data, the main assumption of GPR is $y = f(\mathbf{x}) + \epsilon$ where the output y are considered as the points sequence sampled from a multivariate gaussian distribution, and $\epsilon \sim N(0, \sigma_n^2)$ represents the noice 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 and it can be expressed as:

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

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

In GPR method, the reliability of our regression is dependent on how well we select the covariance function, so it is necessary to estimate the values of the hyperparameters in kernel. According to the GPR theory, the optimal hyperparameters of RBF corresponding to maximizing:

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

where $\boldsymbol{\theta} = \{l, \sigma_f, \sigma_n\}$.

According to above, given the training samples and labels, it is necessary to establish three covariance matrices as follows when we would like to estimate the regression value of the test samples.

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & ... & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & ... & k(\mathbf{x}_2, \mathbf{x}_n) \\ ... & ... & ... & ... \\ ... & ... & ... & ... \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & ... & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \quad (3a)$$

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

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

where $\mathbf{x}_*$ is the sample to be predicted. And then, the probability of y_* follows a gaussian distribution:

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

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

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

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

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

3.2 Our Tracking Framework

Here, we predefine the $\mathbf{K}_{\mathbf{X}_i\mathbf{x}_j}$ as follows.

$$\mathbf{K}_{\mathbf{X}_i\mathbf{x}_j} = \begin{bmatrix} k(\mathbf{x}_{i1}, \mathbf{x}_{j1}) & k(\mathbf{x}_{i1}, \mathbf{x}_{j2}) & ... & k(\mathbf{x}_{i1}, \mathbf{x}_{jn}) \\ k(\mathbf{x}_{i2}, \mathbf{x}_{j1}) & k(\mathbf{x}_{i2}, \mathbf{x}_{j2}) & ... & k(\mathbf{x}_{i2}, \mathbf{x}_{jn}) \\ ... & ... & ... & ... \\ ... & ... & ... & ... \\ k(\mathbf{x}_{in}, \mathbf{x}_{j1}) & k(\mathbf{x}_{in}, \mathbf{x}_{j2}) & ... & k(\mathbf{x}_{in}, \mathbf{x}_{jn}) \end{bmatrix} \quad (6)$$

where $\mathbf{X}_i$ and $\mathbf{X}_j$ come from the different frames when $i \neq j$ or same frame when $i = j$, and $\mathbf{x}_{in}$ represents the n-th sample in $\mathbf{X}_i$. (6) represents the constructed covariance matrices with $\mathbf{X}_i$ and $\mathbf{X}_j$, and it is often used in subsequent sections.

Online Learning. For each frame f_i, where $i = 1, ..., T$, assuming that the target position is p_i and the area of the target is $s_i = w_i \times h_i$. We densely sample the training data around p_i with sampling region $S_i = \sigma\sqrt{w_i \times h_i} \times \sigma\sqrt{w_i \times h_i}$ and all samples $\mathbf{X}_i = \{\mathbf{x}_{ij} \mid j = 1, ..., N\}$ in f_i have the same size s_i. Then, we straightforwardly construct the covariance matrix $\mathbf{K}_{\mathbf{X}_i\mathbf{X}_i}$ based on training samples $\mathbf{X}_i$, and in common with KCF to construct the gaussian like label $\mathbf{y}_i$ for calculating the expectation of test samples.

Online Update and Detection (I) Considering that the general object detection method based on linear weighted in tracking can be formulated as:

$$\mathbf{f}(\mathbf{X}_t) = \sum_{i=1}^{t-1} \beta_i \mathbf{h}(\mathbf{X}_t, \mathbf{X}_i) \quad (t \geq 2) \quad (7)$$

where $\mathbf{X}_t$ represents the test samples in the current frame f_t, $\mathbf{X}_i$ $(i < t)$ represents the training samples in the previous frames f_i $(i < t)$, β_i represents the weight of f_i, $\mathbf{h}(\mathbf{X}_t, \mathbf{X}_i)$ represents the predicted expectation of $\mathbf{X}_t$ based on $\mathbf{X}_i$, and according to (5a) in our GPRT it can be formulated as:

$$\mathbf{h}(\mathbf{X}_t, \mathbf{X}_i) = \mathbf{K}_{\mathbf{X}_t\mathbf{X}_i}\mathbf{K}_{\mathbf{X}_i\mathbf{X}_i}^{-1}\mathbf{y}_i \quad (8)$$

where $\mathbf{y}_i$ is the label vector which obeys the gaussian distribution, and in our GPRT $\mathbf{y}_i = \mathbf{y}_j, \forall i, j$.

Combining (7) and (8), it is easy to obtain the detection formula in our GPRT as follows.

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

In (9), we can observe that for getting $\mathbf{f}(\mathbf{X}_t)$ we need to construct the $\mathbf{K}_{\mathbf{X}_t\mathbf{X}_i}$ as well as $\mathbf{K}_{\mathbf{X}_i\mathbf{X}_i}$ and calculate the $\mathbf{K}_{\mathbf{X}_i\mathbf{X}_i}^{-1}\mathbf{y}_i$ as well as $\mathbf{K}_{\mathbf{X}_t\mathbf{X}_i}\left(\mathbf{K}_{\mathbf{X}_i\mathbf{X}_i}^{-1}\mathbf{y}_i\right)$ for all $i < t$. Compared to the other three operations, the complexity of calculating $\mathbf{K}_{\mathbf{X}_t\mathbf{X}_i}\left(\mathbf{K}_{\mathbf{X}_i\mathbf{X}_i}^{-1}\mathbf{y}_i\right)$ can be negligible. Despite constructing $\mathbf{K}_{\mathbf{X}_i\mathbf{X}_i}$ and calculating $\mathbf{K}_{\mathbf{X}_i\mathbf{X}_i}^{-1}\mathbf{y}_i$ can be done only once then saved, we still need to utilize (6) $(t-1)$ times to construct the $\mathbf{K}_{\mathbf{X}_t\mathbf{X}_i}$ $(i < t)$ in each prediction process and it is extremely time-consuming when t and N is large.

For applying the GPR to tracking, we propose (9) can be approximately replaced by the following expression:

$$\mathbf{f}\left(\mathbf{X}_t\right) = \mathbf{K}_{\mathbf{X}_t \sum_{i=1}^{t-1}\beta_i \mathbf{X}_i} \sum_{i=1}^{t-1} \beta_i \mathbf{K}_{\mathbf{X}_i\mathbf{X}_i}^{-1} \mathbf{y}_i \quad (t \geq 2) \tag{10}$$

and after such a simple approximation we have been greatly reduced the amount of calculation in the process of detection.

After we replace the $\mathbf{f}\left(\mathbf{X}_t\right)$ in (7) with (10), it is easy to obtain the ultimate detection and update formula as follows.

$$\mathbf{f}\left(\mathbf{X}_t\right) = \mathbf{K}_{\mathbf{X}_t\mathbf{A}_{t-1}} \mathbf{B}_{t-1} \quad (t \geq 2) \tag{11a}$$

$$\mathbf{A}_t = \left(1-\delta\right)\mathbf{A}_{t-1} + \delta \mathbf{X}_t \tag{11b}$$

$$\mathbf{B}_t = \left(1-\delta\right)\mathbf{B}_{t-1} + \delta \mathbf{K}_{\mathbf{X}_t\mathbf{X}_t}^{-1} \mathbf{y}_t \tag{11c}$$

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

$$\begin{aligned} &\beta_1 = (1-\delta)^{t-2} \\ &\sum_{i=1}^{t-1} \beta_i = 1 \\ &\beta_{i-1}/\beta_i = 1-\delta \quad (i > 2) \end{aligned} \tag{12}$$

Following (11), we can not only predict the regression value of test samples quickly by our GPRT model but also update our GPRT model in an efficient way. It is worth noting that this scheme allows the model to be updated without storing all the previous information, and only the current model $\mathbf{A}_t$ and $\mathbf{B}_t$ need to be saved for predicting the next frame. By using this predict and update method for our GPRT, we call it GPRTE in the following sections.

Online Update and Detection (II) In the Kernel Ridge Regression (KRR) [Vovk, 2013], we are mainly solving empirical risk minimisation in dual space as:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \left[y_i - \left\langle \mathbf{w}, \phi\left(\mathbf{x}_i\right)\right\rangle\right]^2 + \lambda \left\|\mathbf{w}\right\|^2 \\ &\text{s.t. } \mathbf{w} = \sum_{i=1}^{N} \alpha_i \phi\left(\mathbf{x}_i\right) \end{aligned} \tag{13}$$

where $\phi\left(\mathbf{x}_i\right)$ is mapping the input of a linear problem to a non-linear space.

Considering the relationship between the GPR and KRR [Rasmussen and Williams, 2006], it is easy to prove that if in KRR we use the same kernel as the covariance function in GPR and moreover, the regularisation parameter λ in KRR is the same as the noise variance σ_n^2 in GPR, then the GPR posterior mean coincides with the KRR estimates of the function. For this reason, we wish to take advantage of the relationship between the GPR and KRR to get the update method when applying the GPR to tracking.

For tracking, we can formulate the cost function of KRR in dual space as:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \varepsilon = \sum_{j=1}^{P} \beta_j \left(\sum_{m=1}^{N} \left[\left\langle \phi\left(\mathbf{x}_j^m\right), \mathbf{w}_j \right\rangle - y_j^m\right]^2 + \lambda \left\|\mathbf{w}_j\right\|^2 \right) \\ &\text{s.t. } \mathbf{w}_j = \sum_{i=1}^{N} \alpha_i \phi\left(\mathbf{x}_j^i\right) \end{aligned} \tag{14}$$

where P represents the number of frames, N represents the number of training samples in each frame, $\mathbf{x}_j^m$ and y_j^m represents the m-th sample in j-th frame and its label, similar to (7), β_j represents the weight of each frame, $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_N\}$ represents the dual variables to be optimized and the solution is restricted to only contain one set of dual variables for keeping the simplicity

The cost function in (14) is minimized by:

$$\boldsymbol{\alpha} = \left[\sum_{j=1}^{P} \beta_j \mathbf{K}_{\mathbf{X}_j\mathbf{X}_j} \left(\mathbf{K}_{\mathbf{X}_j\mathbf{X}_j} + \lambda \mathbf{I}\right)\right]^{-1} \left[\sum_{j=1}^{P} \beta_j \mathbf{K}_{\mathbf{X}_j\mathbf{X}_j} \mathbf{y}_j\right] \tag{15}$$

According the relationship between GPR and KRR and compared (5a) and (15), we can naturally bring (15) into (8), and utilize the approximate method in (10) as well, then it is easy to obtain the ultimate detection and update formula as follows.

$$\mathbf{f}\left(\mathbf{X}_t\right) = \mathbf{K}_{\mathbf{X}_t\mathbf{A}_{t-1}} \mathbf{M}_{t-1}^{-1} \mathbf{N}_{t-1} \quad (t \geq 2) \tag{16a}$$

$$\mathbf{A}_t = \left(1-\delta\right)\mathbf{A}_{t-1} + \delta \mathbf{X}_t \tag{16b}$$

$$\mathbf{M}_t = \left(1-\delta\right)\mathbf{M}_{t-1} + \delta \mathbf{K}_{\mathbf{X}_t\mathbf{X}_t} \left(\mathbf{K}_{\mathbf{X}_t\mathbf{X}_t} + \lambda \mathbf{I}\right) \tag{16c}$$

$$\mathbf{N}_t = \left(1-\delta\right)\mathbf{N}_{t-1} + \delta \mathbf{K}_{\mathbf{X}_t\mathbf{X}_t} \mathbf{y}_t \tag{16d}$$

Compared to GPRTE, this update approach considers the relationship between GPR and KRR, moreover, from the perspective of minimizing empirical risk directly derived how to update the GPR model during tracking and it is slightly more complex than GPRTE. Similar to GPRTE, this scheme also allows the model to be updated without storing all the previous information, and only the current model $\mathbf{A}_t$, $\mathbf{M}_t$ and $\mathbf{N}_t$ need to be saved for predicting the next frame. By using this update method for our GPRT, we call it GPRT in the following sections.

3.3 Accelerated Calculation

As we know, in GPR, when the number of training samples and the feature dimension of each sample are large, it is highly time-consuming to calculate the gaussian covariance matrix $\mathbf{K}$. However, it is not difficult to observe that there are a large number of parallel operations in calculating the $\mathbf{K}$. Assuming that the target size is $w \times h$, the sampling region is $W \times H = \sigma w \times \sigma h$, the channels of hand-crafted features is d, so the feature dimension of each sample is $d \times w \times h$ and the number of samples is $(W - w + 1) \times (H - h + 1)$. To establish the gaussian covariance matrix $\mathbf{K}$, the time complexity is about $\mathcal{O}\left(dwhWH\right) = \mathcal{O}\left(dw^2h^2\sigma^2\right)$. We propose that using the GPUs allows the $\mathbf{K}$ to be computed in the time complexity $\mathcal{O}\left(dw^2h^2\sigma^2/P\right)$ instead of $\mathcal{O}\left(dw^2h^2\sigma^2\right)$, where P is the maximum degree of parallelism in GPU. In practice, w and h

Algorithm 1 Proposed tracking framework.

```
Input:
    Initial target location and scale $\mathbf{x}_1 = (p_1, s_1)$;
Output:
    Estimated target location and scale $\mathbf{x}_t = (p_t, s_t)$;
 1: Compute $\mu = \sqrt{\min(4000, \max(s_1, 1000)) / s_1}$ as the
    scaling ratio to ensure the speed and accuracy;
 2: repeat
 3:     Crop out the square searching region $S_t$ in frame $t$
        based on $\mathbf{x}_{t-1}$ and crop ratio $\sigma$;
 4:     Resize $S_t$ according to the $s_1$ and scaling ratio $\mu$;
 5:     Extract the HOG and Color-Names features from $S_t$
        and do L2 regularization to 1 and 0.4, respectively;
 6:     Compute the regression value of test samples using
        (11a) or (16a) and find the maximum value to locate
        the new position of the target $p_t$;
 7:     Build the scale pyramid based on $p_t$ and $s_{t-1}$, then
        using DSST to estimate the scale of the target $s_t$;
 8:     Crop out the square learning region $L_t$ in frame $t$ based
        on $\mathbf{x}_t$ and crop ratio $\sigma$;
 9:     Do 4 and 5 operations on $L_t$;
10:     Update the GPRT model using (11) or (16);
11:     Build the scale pyramid based on $p_t$ and $s_t$, then up-
        date the DSST model;
12: until End of video sequences;
```

are about 15 (HOG and Color-Names features with cell size 4), σ is about 4, d is about 40, and P is about 3600. So, we can probably reduce the time complexity from $\mathcal{O}\left(dw^2h^2\sigma^2\right)$ to $\mathcal{O}\left(dwh\right)$ by GPU which allows us to perform about 3000 times faster than before.

4 Implementation Details

We present an outline of our method in Algorithm 1 and more implementation details are discussed as follows.

Features. For simplicity, our GPRTE and GPRT are both based on hand-crafted features. We use 31-channels HOG [Dalal and Triggs, 2005] and 10-channels Color-Names [Danelljan *et al.*, 2014b] features since they show high robustness in object tracking [Danelljan *et al.*, 2016a; Tang and Feng, 2015; Danelljan *et al.*, 2016a]. Similar to MKCF tracker [Tang and Feng, 2015], we also set the cell size of 4×4 in both HOG and Color-Names features, but unlike it, we normalize the HOG and Color-Names features and make their L2-norm equal to 1 and 0.4, respectively.

Scale. As for scale estimation, we use DSST [Danelljan *et al.*, 2014a] and similar to it we only use HOG features.

Parameters Setup. We set the learning rate δ in section 3.2 to 0.004 and 0.007 for our GPRTE and GPRT, respectively. Meanwhile, we set the learning and search ratio σ in section 3.2 to 4 for both GPRTE and GPRT [2]. For accuracy and speed, we resize the target in the first frame to ensure the minimum and maximum area are 1000 and 4000 pixels, respectively, and in the subsequence, to ensure the number of the training samples in each frame are the same, and the feature dimension of each sample are the same too, we resize the target in each frame to the same as the size in the first frame.

[2]In fact, because of our GPRT does not exist the boundary effect, this characteristic allow us to set up a wider range to learning and detection. But considering the speed of the tracker, we set the learning and search ratio similar to SRDCF.

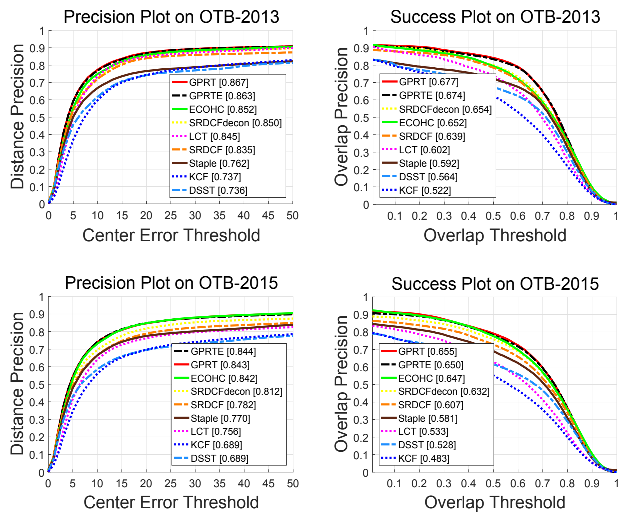

Figure 2: The average precision and success plots of GPRT, GPRTE, ECOHC, SRDCFdecon, SRDCF, Staple, LCT, KCF and DSST on OTB-2013 and OTB-2015 datasets. We report the mean distance precision score and AUC score of each tracker in the legend. Our GPRT and GPRTE outperform all the existing state-of-the-art trackers including ECOHC with hand-crafted features.

Kernel Selection. We use single gaussian kernel described in (1). In gaussian kernel, we set $\sigma_f = 1.0$ and $\sigma_n = 0.01$, and even though the parameter l can be obtained by maximization (2) in the first frame of every sequence, we observe that this parameter roughly presents a gaussian distribution centered at 1.4 on OTB-2015 dataset, therefore for the sake of simplicity we set it to 1.4 on all sequences.

Platform. Our GPRTE and GPRT are both implemented under MATLAB and C++. The experiments are performed on Linux with IntelE5-2673 2.4GHz CPU and single TITAN X GPU with CUDA-8.0.

5 Experimental Results

We show the performance of our GPRTE and GPRT on two public benchmarks: OTB-2013 [Wu *et al.*, 2013] and OTB-2015 [Wu *et al.*, 2015], and compare them with the state-of-the-art trackers with hand-crafted features. All parameters of our GPRTE and GPRT are kept consistent across all experimental comparisons. The mean fps of our GPRTE over the OTB-2015 sequences is about 7 and GPRT is about 5.

All trackers are quantitatively evaluated by six metrics, (i) Center Error, which is calculated as the average Euclidean distance between the centers of located objects and their ground truths in a sequence; (ii) Distance Precision, which is the percentage of frames where the objects are located within the center errors of 0 to t_c pixels, with $t_c = 20$; (iii) Precision Plot, which is simply a curve of the distance precisions with t_c changing from 0 to 50 pixels; (iv) Overlap Ratio, which

Tracker	GPRT	GPRTE	ECOHC	SRDCFdecon	SRDCF	LCT	Staple	DSST	KCF
Mean OP (OTB-2013)	84.1	83.4	79.4	79.9	78.3	73.9	72.2	67.3	62.6
Mean OP (OTB-2015)	79.2	78.3	77.7	75.9	72.8	63.0	69.1	61.6	55.3

Table 1: A comparison with state-of-the-art trackers on the OTB-2013 and OTB-2015 datasets using mean overlap precision (in percent). The best three results for each dataset are shown in red, blue and magenta fonts, respectively. Our GPRT achieves a gain of 4.2% and 1.5% on OTB-2013 and OTB-2015 respectively compared to the second best tracker except for our GPRTE.

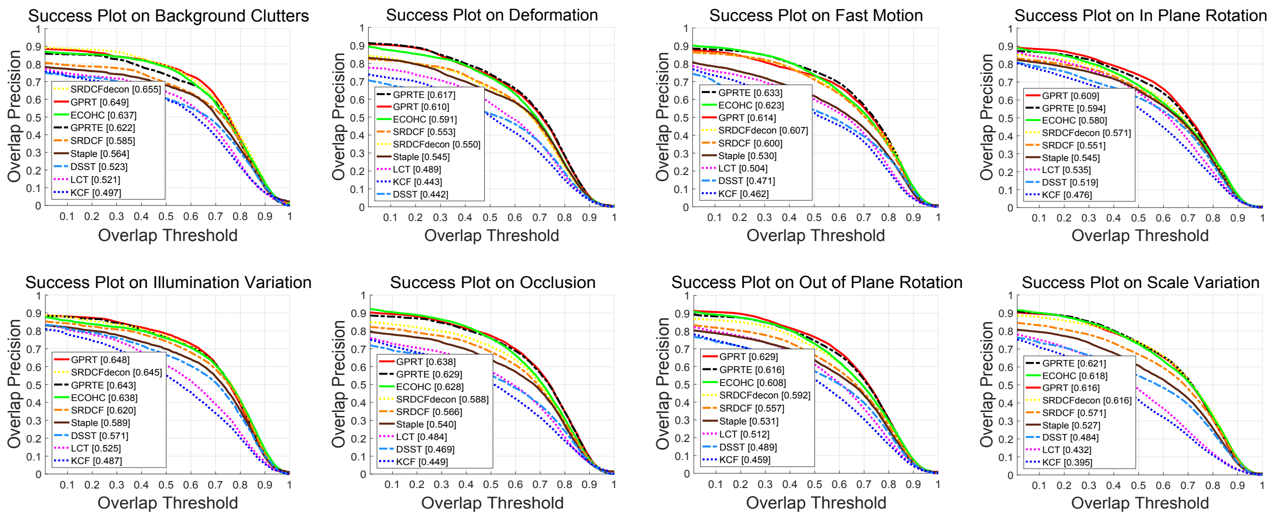

Figure 3: The success plots of GPRT, GPRTE, and the other state-of-the-art trackers on 8 main attributes of OTB-2015 (i.e. background clutters, deformation, fast motion, in plane rotation, illumination variation, occlusion, out of plane rotation and scale variation). We report the mean AUC score of each tracker in the legend. Our GPRT and GPRTE outperform the other state-of-the-art trackers in most cases.

is defined as the average ratio of intersection and union of the estimated bounding box and ground truth in a sequence; (v) Overlap Precision, which is the percentage of frames with the overlap ratio exceeding t_o in a sequence, with $t_o = 0.5$; (vi) Success Plot, which is simply a curve of overlap precisions with the overlap ratio changing from 0 to 1, and AUC is the area under the success plot.

Overall performance. Our tracker, denoted by GPRT and GPRTE, are compared with the other seven state-of-the-art trackers, including ECOHC [Danelljan *et al.*, 2016a], SRDCFdecon [Danelljan *et al.*, 2016b], SRDCF [Danelljan *et al.*, 2015], Staple [Bertinetto *et al.*, 2016], LCT [Ma *et al.*, 2015], KCF [Henriques *et al.*, 2015] and DSST [Danelljan *et al.*, 2014a]. All trackers are based on the hand-crafted features, and all except KCF have the scale estimation part.

Table. 1 shows the mean Overlap Precision (OP) of our GPRT and GPRTE, as well as the other state-of-the-art trackers on OTB-2013 and OTB-2015 datasets. The best results on these two datasets are obtained by our GPRT with a mean OP of 84.1% and 79.2% respectively, leading to a significant gain of 4.2% and 1.5% compared to the second best tracker except for our GPRTE. In addition, our GPRTE also ahead of the other state-of-the-art trackers on both datasets in mean OP.

Figure. 2 shows the average precision and success plots of our GPRT and GPRTE, as well as the other state-of-the-art trackers on OTB-2013 and OTB-2015 datasets, respectively. We also report the distance precisions and AUCs in the legend. Our GPRT and GPRTE have the similar performance on both datasets. Our GPRT obtains the distance precision score of 86.7% and 84.3% as well as the AUC score of 67.7% and 65.5% on OTB-2013 and OTB-2015 datasets, respectively. Furthermore, our GPRT and GPRTE outperform the best existing tracker (ECOHC) with hand-crafted features.

Compared to all the existing CF trackers which achieve the state-of-the-art performance, the boundary effect is eliminated thoroughly and the kernel trick can be employed in our GPRT and GPRTE, therefore our GPRT and GPRTE can achieve better tracking performance than them.

Attribute-based evaluation. The videos in the benchmark dataset OTB-2015 are annotated with 11 attributes which are useful for analyzing the performance of trackers in different aspects. Following [Ma *et al.*, 2015], we also compared our GPRT and GPRTE to the other state-of-the-art trackers on 8 main challenging attributes of OTB-2015 dataset. Figure. 3 shows the success plots and AUCs of our GPRT and GPRTE as well as the other state-of-the-art trackers on 8 main challenging attributes, respectively. It illustrates that our GPRT and GPRTE outperform the other state-of-the-art trackers in most cases.

6 Conclusion

A novel tracking framework, GPRT which applying the Gaussian Regression Processes to visual tracking, has been presented in this paper. Compared to all the existing CF trackers, our GPRT not only does not exist the boundary effect, but also can take advantage of the kernel trick at the same time. In addition, we propose two different efficient and effective update methods for our GPRT. We perform comprehensive experiments on two benchmark datasets: OTB-2013 and OTB-2015. Our GPRT outperforms all the existing trackers with hand-crafted features on both two datasets.

References

[Bertinetto *et al.*, 2016] 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*, pages 1401–1409, 2016.

[Bolme *et al.*, 2010] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2544–2550. IEEE, 2010.

[Dalal and Triggs, 2005] 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*, volume 1, pages 886–893. IEEE, 2005.

[Danelljan *et al.*, 2014a] 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.

[Danelljan *et al.*, 2014b] Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg, and Joost Van de Weijer. Adaptive color attributes for real-time visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1090–1097, 2014.

[Danelljan *et al.*, 2015] 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*, pages 4310–4318, 2015.

[Danelljan *et al.*, 2016a] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: Efficient convolution operators for tracking. *arXiv preprint arXiv:1611.09224*, 2016.

[Danelljan *et al.*, 2016b] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1430–1438, 2016.

[Ebden, 2015] Mark Ebden. Gaussian processes: A quick introduction. *arXiv preprint arXiv:1505.02965*, 2015.

[Gao *et al.*, 2014] Jin Gao, Haibin Ling, Weiming Hu, and Junliang Xing. Transfer learning based visual tracking with gaussian processes regression. In *European Conference on Computer Vision*, pages 188–203. Springer, 2014.

[Henriques *et al.*, 2015] 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*, 37(3):583–596, 2015.

[Hensman *et al.*, 2013] James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

[Kiani Galoogahi *et al.*, 2015] Hamed Kiani Galoogahi, Terence Sim, and Simon Lucey. Correlation filters with limited boundaries. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4630–4638, 2015.

[Ma *et al.*, 2015] Chao Ma, Xiaokang Yang, Chongyang Zhang, and Ming-Hsuan Yang. Long-term correlation tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5388–5396, 2015.

[Quiñonero-Candela and Rasmussen, 2005] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

[Rasmussen and Williams, 2004] Carl Edward Rasmussen and Christopher KI Williams. Gaussian processes in machine learning. *Lecture notes in computer science*, 3176:63–71, 2004.

[Rasmussen and Williams, 2006] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[Tang and Feng, 2015] Ming Tang and Jiayi Feng. Multi-kernel correlation filter for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3038–3046, 2015.

[Titsias, 2009] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574, 2009.

[Vovk, 2013] Vladimir Vovk. Kernel ridge regression. In *Empirical inference*, pages 105–116. Springer, 2013.

[Wu *et al.*, 2013] 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*, pages 2411–2418, 2013.

[Wu *et al.*, 2015] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.