Pattern Recognition Letters 56 (2015) 52–59

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Robust visual tracking based on product sparse coding ☆

Huang Hong-tu*, Bi Du-yan, Zha Yu-fei, Ma Shi-ping, Gao Shan, Liu Chang

Aeronautics and Astronautics Engineering College, Air Force Engineering University, Xi'an 710038, China

ARTICLE INFO

Article history:
Received 28 September 2014
Available online 16 February 2015

Keywords:
Visual tracking
Product sparse coding
L_1-norm minimization
Ridge regression
Support vector machine

ABSTRACT

In this paper, we propose a sparse coding tracking algorithm based on the Cartesian product of two sub-codebooks. The original sparse coding problem is decomposed into two sub sparse coding problems. And the dimension of sparse representation is intensively enlarged at a lower computational cost. Furthermore, in order to reduce the number of L_1-norm minimization, ridge regression is employed to exclude the substantive outlying particles according to the reconstruction error. Finally the high-dimension sparse representation is put into the classifier and the candidate with the maximal response is considered as the target. Both qualitative and quantitative evaluations on challenging benchmark image sequences demonstrate that the proposed tracking algorithm performs favorably against several state-of-the-art algorithms.

1. Introduction

Visual tracking has long been playing a critical role in numerous applications such as surveillance, military reconnaissance, motion recognition and traffic monitoring, to name a few [1]. While much progress has been made within the last decades, it still remains challenging in many scenarios including pose variation, illumination change, partial occlusion, motion blur, background clutter and so on.

In the past few years, variation and extension of L_1-norm minimization have been applied to many computer vision tasks, including face recognition, image super-resolution, denoising, inpainting and image classification [2]. Inspired by the success of sparse representation in face recognition [3], many researchers develop a robust visual tracking framework by casting the tracking as a sparse approximation on the codebook [4]. A thorough review can refer to [5]. The sparse coding visual tracking algorithms can be classified into two categories, generative model and discriminative model. Both of them require obtaining the sparse representation firstly. And the approach to sparse representation is a L_1-norm minimization problem, which can be solved by homotopy method, gradient projection method, iterative shrinkage-thresholding method, interior-point method and so on [6]. As we know sparse coding is a competitive method given sufficiently large codebooks [7]. However, sparse coding is computationally expensive and the computational cost increases sharply with the size of the codebook. So its power is mostly limited by the size of the codebook in practice, especially for discriminative sparse coding tracking algorithm. So many researchers have to make a trade-off between the speed and the discriminative ability. Given a proper computational cost, how to enlarge the codebook to improve the discriminative power is urgent to be solved.

In this paper we propose a robust product sparse coding tracking algorithm. And the codebook size is increased in product manner at a lower computational cost than direct operation on the Cartesian product of two sub-codebooks [7]. The original sparse coding problem is decomposed into two sub sparse coding problems. Each codeword in the codebook is divided into two equal parts. Then the sparse representation of the candidate can be obtained on the two sub-codebooks simultaneously. And the final sparse representation can be calculated via the product of the two obtained sparse coding coefficients. Finally the high-dimension sparse representation is input into the SVM classifier and the candidate with the maximal score is regarded as the target. In order to reduce the number of L_1-norm minimization, ridge regression is adopted to exclude the candidates with big reconstruction error at a lower computational cost. After that, tracking is led by the Bayesian state inference framework in which a particle filter is used for propagating sample distributions over time. Numerous experiments on various challenging sequences show that the proposed algorithm performs favorably against state-of-the-art methods and the tracker based on product sparse coding is superior to the original sparse coding tracker under the same condition.

The rest of the paper is organized as follows. In Section 2, we begin with summarizing the related work on sparse coding tracking. In Section 3, we offer the details of the sparse representation based on product sparse coding. Section 4 is the initialization and generalization analysis of the SVM classifier used in our paper. The integration of our proposed model in particle filter framework for tacking is described in Section 5. Qualitative and quantitative evaluations of our

☆ This paper has been recommended for acceptance by A. Fernandez-Caballero.

* Corresponding author. Tel.: +86 29 84787724; fax: +86 29 84787724.
E-mail address: huanghongtu@sina.cn (H. Hong-tu).

http://dx.doi.org/10.1016/j.patrec.2015.01.014
0167-8655/

tracker on a number of challenging videos are presented in Section 6. Section 7 is the conclusion of our paper.

2. Related work

The computational complexity of the sparse coding visual tracking based on L_1-norm minimization is $o(Nm^2n^{3/2})$, where N is the number of the candidates, m is the codeword dimension, n is the number of codeword in the codebook [5]. So two kinds of method can be utilized to reduce the computation complexity. One is to reduce the computational cost of each L_1-norm minimization. The other is to reduce the number of L_1-norm minimization. Instead of reducing the dimension by down sampling the cropped images, Liu et al. proposed a feature selection method to choose low dimension but more discriminative features [8]. In Wu et al., to reduce the dimension of the linear representation, they proposed the covariance matrix to represent the target and the candidates [9]. In Li et al., they exploit the restricted isometry property to reduce the dimension of the feature by multiplying hashing matrix simultaneity [10]. In Zhong et al., they try to get the projection matrix through the samples and corresponding labels to select more discriminative features [11]. Afterward the training samples and the candidates are projected to the selected discriminative feature space using the projection matrix. In Yan et al., they propose a weighted lasso to handle occlusion instead of using identity pixel to represent the errors caused by occlusion [12]. To reduce the number of L_1-norm minimization, in Zhang et al. [13], they use the candidates instead of the target to construct the codebook, so the number of L_1-norm minimization is required only once in theory to identify the target. In Mei et al. [14], the number of L_1-norm minimization is reduced by exploiting the fast computational lower bound of the reconstruction error to exclude the unimportant particles. In Liu et al. [15], they integrated a motion model into the sparse representation. Starting from the target initial state, a gradient based optimization procedure is iterated to find the sparse representation and the corresponding gradient vector. Although the computational cost is reduced to some extent using the above methods, the codebook size does not increase in essence. Thus the sparse coding based tracking can appear less competitive due to the limited codebook size, typically when the accuracy of other encoding methods can be improved by simply enlarging the codebook.

3. Product sparse coding based appearance mode

3.1. Sparse representation based on product sparse coding

In the visual tracking based on sparse coding, the codebook can be constructed by the hand-designed or learning method [16]. In this paper we choose the hand-designed method due to its efficiency and simplicity. The codebook is constructed by the SIFT descriptors of the target and its 8 neighbors without trivial templates. For candidate $\boldsymbol{Y}_i$, the patches SIFT descriptors $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k$ [18] are densely extracted by a 16×16 sliding window with step of 8 pixel, where k is the number of patches in one candidate. In the case of two sub-codebooks, each SIFT descriptor is denoted as $\mathbf{x} = [\mathbf{x}_1^{\mathrm{T}}, \mathbf{x}_2^{\mathrm{T}}]^{\mathrm{T}}$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ are the first and second half of $\mathbf{x}$. Similarly, any codeword in $\boldsymbol{D} \in \mathbb{R}^{m\times n}$ can be represented as $\mathbf{d} = [\mathbf{d}_{1p}^{\mathrm{T}}, \mathbf{d}_{2q}^{\mathrm{T}}]^{T}$, where $\mathbf{d}_{1p}$ is the pth codeword in $\mathbf{D}_1 \in \mathbb{R}^{\frac{m}{2}\times n}$ and $\mathbf{d}_{2q}$ is the qth codeword in $\mathbf{D}_2 \in \mathbb{R}^{\frac{m}{2}\times n}$, for $p, q = 1,2,\ldots,n$. We denote the coefficient of this codeword as β_{pq}. Then the objective function can be written as

$$\beta = \arg\min_{\beta} \frac{1}{2} \left\| \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} - \sum_{p,q} \begin{bmatrix} \mathbf{d}_{1p} \\ \mathbf{d}_{2q} \end{bmatrix} \beta_{pq} \right\|_2^2 + \lambda ||\beta||_1 \quad \text{s.t.} \quad \beta \geq 0 \tag{1}$$

We introduce two vectors α_1 and α_2, whose entries are defined as $\alpha_{1p} = \sum_q \beta_{pq}, \alpha_{2q} = \sum_p \beta_{pq}$. The first term of (1) can be expanded as $\frac{1}{2}||\mathbf{x}_1 - \mathbf{D}_1\alpha_1||_2^2 + \frac{1}{2}||\mathbf{x}_2 - \mathbf{D}_2\alpha_2||_2^2$. Since $\beta \geq 0$, the vectors α_1 and α_2 are subject to the constraint $\sum_{p,q}|\beta_{pq}| = \sum_p |\alpha_{1p}| = \sum_q |\alpha_{2q}|$, i.e. $||\beta||_1 = ||\alpha_1||_1 = ||\alpha_2||_1$. To give a separate form of β, we introduce two parameters $\lambda_1 (0 < \lambda_1 < \lambda)$ and $\lambda_2 = \lambda - \lambda_1$, and we can define the PSC problem as [7]

$$[\alpha_1, \alpha_2] = \arg\min_{\alpha_1,\alpha_2} \begin{pmatrix} \frac{1}{2}||\mathbf{x}_1 - \alpha_1||_2^2 + \lambda_1||\alpha_1||_1 + \\ \frac{1}{2}||\mathbf{x}_2 - \alpha_2||_2^2 + \lambda_2||\alpha_2||_1 \end{pmatrix} \quad \text{s.t.} \quad \alpha_1 \geq 0, \alpha_2 \geq 0; \quad |\alpha_1||_1 = ||\alpha_2||_1 \tag{2}$$

If we ignore the constraint $||\alpha_1||_1 = ||\alpha_2||_1$, we can get two separate sub sparse coding problems

$$\alpha_1 = \arg\min_{\alpha_1} \frac{1}{2}||\mathbf{x}_1 - \mathbf{D}_1\alpha_1||_2^2 + \lambda_1||\alpha_1||_1 \quad \text{s.t.} \quad \alpha_1 \geq 0 \tag{3}$$

$$\alpha_2 = \arg\min_{\alpha_2} \frac{1}{2}||\mathbf{x}_2 - \mathbf{D}_2\alpha_2||_2^2 + \lambda_2||\alpha_2||_1 \quad \text{s.t.} \quad \alpha_2 \geq 0 \tag{4}$$

Each is a sparse coding problem. But the codebooks are much smaller. Solving these two sub-problems can be much faster. We introduce the multi-task sparse representation [17] into the two sub-problems. All the sub-sparse coefficients are computed together instead of the independent operation in the original PSC. So we exploit the sparse coefficients relevance among the particles due to the spatial relation, making the sparse representation more robust and stable. In this paper we use a reasonable approximation of PSC, i.e. (i) set $\lambda_1 = \lambda_2 = 0.5\lambda$, (ii) solve the two sub-problems in (3) and (4). Given α_1 and α_2, the following β is a solution satisfying the constraint.

$$\beta = \frac{\mathrm{vec}(\alpha_1\alpha_2^{\mathrm{T}})}{\sqrt{||\alpha_1||_1||\alpha_2||_1}} \tag{5}$$

where $\mathrm{vec}(\cdot)$ rearrange the matrix $\alpha_1\alpha_2^{\mathrm{T}}$ into a vector column-wise.

From the above analysis, we can see clearly that the sparse representation based on the above is an approximation of sparse representation based on the codebook $\mathbf{D}_e \in \mathbb{R}^{m\times n^2}$, where $\mathbf{D}_e$ is the Cartesian product of $\mathbf{D}_1$ and $\mathbf{D}_2$. The precise form of the PSC can be denoted as

$$\hat{\beta} = \arg\min_{\hat{\beta}} \frac{1}{2}||\mathbf{x} - \mathbf{D}_e\hat{\beta}||_2^2 + \lambda||\hat{\beta}||_1 \quad \text{s.t.} \quad \mathbf{D}_e = \mathbf{D}_1 \times \mathbf{D}_2;\ \hat{\beta} \geq 0 \tag{6}$$

So the sparse representation β based on (3)–(5) is a reasonable approximation of $\hat{\beta}$. Obviously the codebook $\mathbf{D}$ is a subset of $\mathbf{D}_e$. So the sparse representation of the candidate $\mathbf{Y}_i$ based on PSC is concatenated by all the patch sparse coefficients together.

$$\rho_i = \left[\beta_1^{\mathrm{T}}, \beta_2^{\mathrm{T}}, \ldots, \beta_k^{\mathrm{T}}\right]^{\mathrm{T}} \tag{7}$$

3.2. Complexity analysis

For codebook $\mathbf{D} \in \mathbb{R}^{m\times n}$, the computation complexity for computing the sparse representation based on lasso is $o(m^2n^{3/2})$ [5]. The comparison of the computation complexity of SC and PSC is shown in Table 1. The computational cost of the PSC consists of two parts, the first part is the cost of the two sparse coding sub-problems, the second part is the cost of the product operation in (4). We can see clearly that the computational cost of PSC is lower than the cost directly computed on the codebook $\mathbf{D}_e$.

Table 1
The comparison of the computation complexity.

	Sparse coding	Product sparse coding
Codebook	$\mathbf{D}_e \in \mathbb{R}^{m\times n^2}$	$\mathbf{D}_1, \mathbf{D}_2 \in \mathbb{R}^{\frac{m}{2}\times n}$
Complexity	$o(m^2n^3)$	$o(0.5m^2n^{3/2} + n^2)$