Video Tracking Using Learned Hierarchical Features

mingzailao

2016-9-11

Outline

ASLSA(adaptive structural local sparse appearance model) [1]

Tracking System Overview

Briefly Introduction of the Tracking System

Suppose we have an observation set of target $x_{1:t} = \{x_1, \cdots, x_t\}$, a corresponding feature representation set $z_{1:t} = \{z_1, \cdots, z_t\}$, the target state y_t can be calculated by:

$$y_t = arg \max_{y_t^i} p(y_t^i | z_{1:t}) \tag{1}$$

where y_t^i denotes the i^{th} sample in the t^{th} frame.

Tracking System Overview

Briefly Introduction of the Tracking System

The posterior probability $p(y_t|z_{1:t})$ can be inferred by the Bayes theorem as follows:

$$p(y_t|z_{1:t}) \propto p(z_t|y_t) \int p(y_t|y_{t-1})p(y_{t-1}|z_{1:t-1}) \quad (2)$$

where $z_{1:t}$ denotes the feature representation, $p(y_t|y_{t-1})$ denotes the motion model and $p(z_t|y_t)$ denotes the appearance model.

Tracking System Overview

Briefly Introduction of the Tracking System

The representations $z_{1:t}$ can simply use raw pixel values. [1] In there , we use the learned hierarchical features from raw pixels for tracking.

Learning Features for Video Tracking

Offline Learning

- Adopt the approach proposed in [2] to learn features From a auxiliary dataset.
- We further use a domain adaptation method to adapt pre-learned features according to specific target objects.

Learning Features for Video Tracking

Algorithm

- Input: the previous tracking state y_{t-1}, the existing feature learning parameter $\hat{\Theta}$ and the exemplar library.
- Apply the affine transformation on y_{t-1} to obtain a number of tracking states y_t^i and the corresponding candidate image patches x_t^i.
- Extract feature representations z_t^i from the candidate image patches x_t^i under the existing feature learning parameter $\hat{\Theta}$.

Learning Features for video Tracking

Algorithm

- Calculate the posterior probability $p(y_t|z_{1:t})$ according to Equation (3).
- Predict the tracking state by $y_t = arg\max_{y_t^i} p(y_t^i|z_{1:t})$.
- Update the feature learning parameter and the exemplar library every M frames.
- Output: the predicted tracking state y_t, the up-to-date feature learning parameter Θ and the up-to-date exemplar library.

Pre-Learning Generic Features from Auxiliary Videos

Deep Learning model

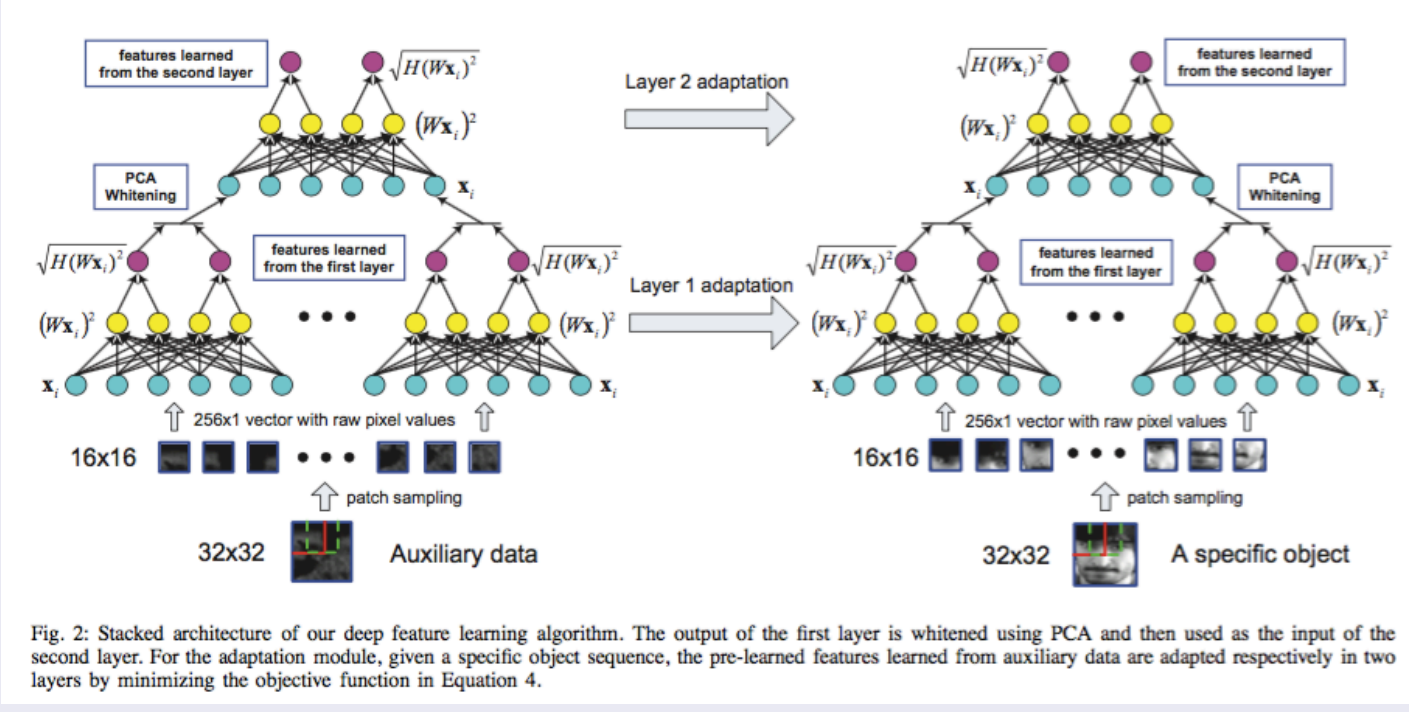

Fig. 2: Stacked architecture of our deep feature learning algorithm. The output of the first layer is whitened using PCA and then used as the input of the second layer. For the adaptation module, given a specific object sequence, the pre-learned features learned from auxiliary data are adapted respectively in two layers by minimizing the objective function in Equation 4.

Figure: The Structure of the Network

Pre-Learning Generic Features from Auxiliary Videos

Input

Offline training patch x_i from the i-th frame,

Corresponding Learned Feature

$$z_i = \sqrt{H(Wx_i)^2}$$

- H : pooling matrix.
- $(Wx_i)^2$: elementwise square on the output of the linear network layer.

Pre-training Generic Features from Auxiliary videos

How to compute the parameter W:

The feature transformation matrix W is learned by solving the following unconstrained minimization problem:

$$\min_{W} \lambda \sum_{i=1}^{N-1} ||z_i - z_{i-1}||_1 + \sum_{i=1}^{N} ||x_i - W^T W x_i||_2^2 \qquad (3)$$

- z_{i+1} denotes the learned feature from the $(i+1)$-th frame.
- N is the total length of all video sequences in the auxiliary data.

Pre-training Generic Features from Auxiliary videos

Apply the W^{L_1} to the 32×32 patches

The input of the first layer is the raw pixel values of smaller patches (16×16).

- We can learn the feature transformation matrix W^{L_1} for the first layer by Equation (3).
- Then, we apply W^{L_1} to convolve with the larger patches (32×32).

The larger patch is divided into a number of sub-patches (16×16) We use W^{L_1} to conduct feature mapping for each sub-patch and concatenate features of all the sub-patches to represent the larger patch.

- Next, PCA whitening is applied to the concatenated feature vector.

Pre-Training Generic Features from Auxiliary videos

Finally

- Using the whitened feature vector of the larger patch as the input to the second layer and learn the feature transformation matrix W^{L_2} for the second layer.
- We concatenate features from two layers as our generic features.

Domain Adaption Module

The Adapted Feature

$$z_i^{adp} = \sqrt{H(Wx_i^{obj})^2}$$

- x_i^{obj} : the object image patch in the i-th frame of the training data for adaptation.
- W : feature transformation matrix to be learned.

Domain Adaption Module

Adding Regularization Term

$$
\begin{aligned}
W_{adp} &= \arg\min_{W} \lambda \sum_{i=1}^{N-1} ||z_i^{adp} - z_{i+1}^{adp}||_1 \\
&+ \gamma \sum_{i=1}^{N} ||Wx_i^{obj} - W_{old}x_i^{obj}||_2^2 \\
&+ \sum_{i=1}^{N} ||x_i^{obj} - W^T W x_i^{obj}||_2^2
\end{aligned} \tag{4}
$$

- W_{old} : the pre-learned feature transformation matrix.

Domain Adaption Module

Analysis

The second term refers to the adaptation module and aims to make the adapted feature close to the old one for the sake of preserving the pre-learned features robustness to complicated motion transformations.

Optimization and Online Learning

Notes

Denotes the objective function of the adaptation module as $f(X; \Theta, \hat{\Theta})$

- X : a number of training images of object regions for the adaptation.
- $\Theta = \{w_{ij} | i, j = 1, \cdots, N\}$:the parameter set representing all entries in the transformation matrix W.
- $\hat{\Theta}$: W_{old}.

Optimizier

L-BFGS

Optimization and Online Learning

BFGS

The Quasi-Newton methods, such as BFGS algorithm, need to update the approximate Hessian matrix B_k at the i-th iteration to calculate the search direction $p_k = B_k^{-1} \nabla f_k$, where ∇f_k is the derivative of the objective function f w.r.t. Θ_k at the k-th iteration.

L-BGFS

Reference

Xu Jia, Huchuan Lu, and Ming-Hsuan Yang.
Visual tracking via adaptive structural local sparse appearance model.
In *Computer vision and pattern recognition (CVPR), 2012 IEEE Conference on*, pages 1822–1829. IEEE, 2012.

Will Zou, Shenghuo Zhu, Kai Yu, and Andrew Y Ng.
Deep learning of invariant features via simulated fixations in video.
In *Advances in neural information processing systems*, pages 3212–3220, 2012.