Robust visual tracking via combinative deep learning

Jia Wen1,2,3,4, JiWei Zhao1,2,3, PengFei Liu1,2,3 and Hongjun Wang1,2,3

1 The Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province, Yanshan University, Qinhuangdao, P.R.China
2 Information Science and Engineering College, Yanshan University, Qinhuangdao, P.R.China
3 State Key Laboratory of Software Engineering of Hebei Province, Yanshan University, Qinhuangdao, P.R.China
4 E-mail: wjyanyuan@aliyun.com

Abstract. Object tracking is a hot topic in computer vision. In recent years, a large number of trackers has been proposed, in which the deep learning tracker has achieved excellent performance. The real-time capability of the deep learning tracker is not good enough due to the high-complexity of the network structures. This paper proposed an innovative tracking method to solve this problem. There are three important differences between this tracker and the other deep learning trackers. Firstly, the overcomplete basis in the deep learning tracker results in heavy computational cost. In order to reduce the complexity of the network, fewer units are used in the first hidden layer to replace the overcomplete basis. Secondly, a training method combining two observation models is used in the tracking process. The denoising automatic encoder is used in the first layer and the backpropagation is used in the other layers. This can avoid the diffusion of gradients which is caused by BP and adapt to the change of the targets easier. Thirdly, this tracker using adaptive particle filter to track targets. The number of particles is dynamic changes in tracking process. In this paper, we use different kinds of unlabelled datum to train network and initialize observation model. The observation model uses the samples collected in the tracking to adjust dynamically so as to adapt to the target appearance and complex environment. Compared with the existing methods, the results of experiments in different video sequences show that this tracker has a higher speed and the similar accuracy compared.

1. Introduction

Object tracking is one of the most important topics in computer vision and has been applied in many fields. Object tracking has become a challenging task due to the effects of occlusion, illumination variation, out-of-plane rotation, in-plane rotation, fast motion and background clutters. Although lots of researches have been done in recent years, it still need be improved.

In general, observation model is used to describe the target [1]. The observation model can be divided into generative model and discriminative model. The purpose of the generative tracking methods is to learn a model that represents the appearance of object and identifies the best matched candidate from many observations. One of the problems with the generative model is that it ignores the background information. Examples of generative method are incremental visual tracking (IVT) [2], visual tracking decomposition (VTD)[3], L1[4–6] and local sparse appearance model [7]. IVT is based on principal component analysis (PCA). IVT learns an adaptive linear subspace online to simulate the
appearance of object. The effect of IVT is less when dealing with heavy occlusion and non-rigid distortion. The observation model of VTD is decomposed into multiple basic observation models that are constructed by sparse principal component (SPCA) of a set of feature templates. Each basic observation model covers a specific appearance of the object. All basic trackers are then integrated into one compound tracker through an interactive Markov Chain Monte Carlo (IMCMC) framework. L1 assumes that the tracked object can be represented by a sparse combination of overcomplete basis vectors. Some extensions of this method increase speed and accuracy. Baiyang Liu et al. proposed a tracker based on local sparse model, which makes use of histograms of sparse coefficients and the mean shift algorithm for object tracking. In general, generative model gets more accurate results in simple circumstances due to richer image representation.

In complex environment, the discriminant model compares the generated model with background information. Discriminative model regards tracking as a classification problem, which can distinguish the background and target. Examples of discriminative method are ensemble tracking [8], Online algorithm boosting (OAB) [9], tracking learning detection (TLD) [10] and struck [11]. In ensemble tracking, a set of weak classifiers are trained and combined to distinguishing the background and target. The features that ensemble tracking used might contain redundant and irrelevant information which affects the classification performance. Helmut Grabner et al. developed an online boosting algorithm to select features for tracking. However, it only uses one positive sample and few negative samples when model is adjusted. When the model is updated with noisy and potentially misaligned examples, this often leads to the tracking drift problem. TLD adds a detection stage after tracking phase, which can recover the tracker from failure, even the object is occluded for a long time. Struck presented a framework for adaptive object tracking based on structured output prediction. The performance is better when there are occlusion and appearance deformation because discriminative tracker takes background information into consider and distinguish between background and target.

In recent years, Naiyan Wang proposed deep learning tracker (DLT) [12]. The stacked denoising autoencoders (SDAE) [13-14] and overcomplete basis are used to train the network offline with a large number of images to learn generic features. And then, BP algorithm is used to adjust the observation model to adapt to target. The observation model is retrained with samples when the appearance changes obviously to make observation model adapt to the appearance deformation. Jason Kuen et al. proposed deep slow tracker (DST) [15]. They added temporal slowness constraint to autoencoder to learn invariant representations. They adopt stacked convolutional autoencoder to learn more complex invariant features. Hong et al. employed CNN and online SVM to get saliency map to track target [16]. They applied pre-trained CNN to extract features and used online SVM as discriminative appearance model to classify samples. Then target-specific features are identified by SVM weights and positive samples. The target-specific features are back-projected though CNN to get target-specific saliency map. The map can be exploited to obtain generative target appearance models and locate the target. In the experiments, they have higher accuracy compared with other trackers, but the complex structure retards the tracking speed.

This paper proposed deep combinative tracker (DCT) to overcome the problem of consuming huge computation. The first hidden layer utilizes a denoising autoencoder (DAE) to prevent inhomogeneities of gradient in the deep network to adapt to a particular object in tracking. In higher layers, BP is used to optimize the performance of classification. DAE is used in first hidden layer instead of BP. The weights of the first layer can be trained more sufficiently in this way and the extracted features will be regarded as the input of higher layers. BP is used to adjust the network in the overall perspective. In the process of particle filter, a filter is added to control the number of particles adaptively. This filter speeds up tracking greatly. It can be divided to two situations. In this filter, a classifier is added to the third layer to filter particles.

2. Method
In this paper, we use a special tracker called DCT to track objects. The framework of this method is shown in figure 1. The framework can be divided into two parts: (1) Offline training. Different
categories of unlabelled images and SDAE are used to pre-training and the network which can adapt to specific object will be get. (2) Online tracking. The observation model is initialized with the weights of the network and trained with several specified samples to adapt to target the goal of tracking initialization. In the initialization, the first hidden layer of the network is trained by DAE and BP is applied to the higher layers. And then the particle filter is used to tracking objects. The adaptive filter which can reduce the number of particles in some cases can be applied to accelerate tracking. This filter divide the tracking into two situations, each situation corresponds to a strategy. The observation model will be retrained when the appearance of target changes obviously. Finally, the particle which has highest confidence will be regard as the target in every frame. The details of the key section are as follows.

2.1. Offline training
We use tiny image dataset and SDAE to train network in offline training. SDAE is an unsupervised algorithm whose basic component is DAE. DAE which is composed of three layers is the extension of classic autoencoder. The network can be trained by corrupted samples to reconstruct the input image and learn specific structure from the datum because the units in hidden layer are less than the input. This can be regard as a method of dimensionality reduction. During training, \(x_i\) represents the original data and \(\tilde{x}\) represents the corresponding corrupted data. Corrupted data could be additive Gaussian noise, masking noise, or salt-and-pepper noise. The process is as follows:

\[
\begin{align*}
    y &= f_\theta(W\tilde{x} + b) \\
    z &= g_{b'}(W'y + b')
\end{align*}
\]

Here \(W\) and \(W'\) denote the weights of encoder and decoder, \(b\) and \(b'\) denote basis terms, \(f\) and \(g\) are nonlinear activation functions which are the logistic sigmoid function generally, \(y\) is the activation of first hidden layer. \(z\) is the activation of the second hidden layer. DAE can extract features by adjusting parameters of optimization problem:

\[
\min_{W,W',b,b'} \sum_{i=1}^{k} \|x_i - z_i\|^2_2 + \lambda(\|W\|^2_f + \|W'\|^2_f)
\]

Here \(\lambda\) is the parameter of weight penalty term which balances the reconstruction loss and weight penalty terms, \(\|\| \) denotes the L2 norm (Frobenius norm).

Sparsity constraint can be added to DAE to extract more effective features. The unit is active when the output of the unit is close to 1 and the unit is inactive when the output value is close to 0. The
sparsity constraint constrains the units to be inactive at most of time. \( \rho \) represents the target sparsity level which is a number close to 0. \( \hat{\rho}_j \) represents average activation of hidden units \( j \).

\[
\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} [a_j(x^{(i)})]
\]  

(4)

here \( a_j \) is the activation of hidden units \( j \). The sparsity constraint can be described as follows:

\[
KL(\hat{\rho} || \rho) = \sum_{j=1}^{m} \left [ \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j} \right ]
\]  

(5)

due to \( \rho \) is a small value, the sparsity constraint can constrain the average activation.

2.2. Adaptive observation model

The weights obtained from offline training will be used to initialize observation model for tracking. In order to achieve better tracking performance, observation model needs be modified in tracking. On one hand, the observation model needs to adapt to target before tracking. On the other hand, the observation model needs to adapt to the target which appearance may change obviously in tracking. Our observation model is discriminative model which needs positive and negative samples in the training process. The targets which are confirmed could be regarded as positive samples. When the observation model needs to be modified, negative samples can be obtained in the background around the last positive sample. The adaptive of model could be achieved by supervised training.

2.2.1. Samples collection: The target is specified by bounding box in the first frame. In initialization, positive samples can be collected by translating one pixel in two directions and this method can get 10 samples in total. Negative samples can be collected from background at a short distance from target and updated in tracking, the tracking result will be added to positive samples and replaces one of them. The specified samples will be fixed in positive samples to prevent samples from drifting in tracking.

2.2.2. Online training: In initialization, a logistic classification layer will be added to the network. Observation model only uses the encoder part of SDAE obtained from offline training. The logistic classification layer will give the confidence of each particle after making a forward pass through the network.

When a new frame arrives, the particles will be drawn by particle filter and the confidence of each particle is determined by the observation model. The particle which has maximum confidence is the target. After training, the reconstruction error rate of DAE will be reduced to a small range. Then, the invariant features are regarded as input of higher network.

When a new frame arrives, the particles will be drawn by the particle filter and the confidence of each particle is determined by the observation model. The particle which has the maximum confidence is the target. When the maximum confidence below threshold \( \tau \), the appearance of the target had changed obviously and the observation model needs to be retained using the collected samples to adapt to the appearance of the target. The BP network results in gradient diffusion that the error term is gradually reduced and the weight of the lower level is updated slowly, which can be solved by reducing the units of the first hidden layer and using DAE to train the network. Concretely, the number of units in the first hidden layer is reduced from 2560 to 768 and the DAE allows the first hidden layer to fully adapt to the appearance change of the target. By this way, this model can extract the invariant features of different objects and speed up the tracking and show the accuracy which similar to the existing methods. After training, the reconstruction error rate of DAE will be reduced to a small range, and then the invariant features are regarded as input of higher network.

The training will be stopped when the mean squared error on training set is less than threshold \( \alpha \) or the training times is greater than threshold \( \beta \) to achieve real-time performance. When the reconstruction error rate raised, it indicates that the network has been adjusted sufficiently. The deeper network also uses the same method. The observation model needs to be adjusted more sufficiently in initialization, so the threshold \( \beta \) is larger than tracking. In tracking, the observation model just needs fine tuning and the speed of tracking is also ensured. When the maximum confidence is still larger than
threshold after several frames, observation model needs to be adjusted at the same to adapt to the change of background. The threshold needs to be selected carefully, a suitable threshold needs to prevent adjusting frequently while ensure the correct rate. The learning rate and the momentum of the network should be suitable for adjustment of tracking. Meanwhile, sparse coding and weight penalty term will be added to the network. Figure 1 shows the structure of the online tracking network.

2.3. Dynamic model
The observation model needs to combine with the dynamic model which is used to model the states and state transition of the target to determine the location of the target. The dynamic model we used is the particle filter [15]. The particle filter is easy to implement multi-object tracking due to its non-linear and non-Gaussian assumptions. In the particle filter, a large number of particles which contribute to achieve better effects in the frames which the appearance changes sharply are usually used to ensure precision, but in most case the speed of the particle filter slows down in the tracking process. Add a classifier to the network to establish an adaptive method that can adaptively determine the number of particles, which can effectively improve the tracking speed.

The particle filter which is based on Bayesian filter and applies hidden Markov model (HMM) is used in the tracking process. At time t, \( x_t \) indicates the latent state and \( y_t \) indicate the observed state. The latent state \( x_t \) indicates the location of target which can be represented by six parameters: translation, scale, aspect ratio, rotation and skewness. At time \( t-1 \), we assume that the probability density function (PDF) \( p(x_{t-1}|y_{1:t-1}) \) is known. The PDF can be used to calculate the posterior probability distribution of \( x_t \). The formula is as follows:

\[
p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1}
\]

(6)

At time t, we can get the new observation \( y_t \). The posterior probability distribution of \( x_t \) could be updated as follows:

\[
p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}
\]

(7)

Where \( p(y_t|y_{1:t-1}) = \int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t \)

In general, the calculation of \( p(x_t|y_{1:t}) \) is difficult. We adopt sequential Monte Carlo importance sampling method to approximate it. The particles with weights can be used to approximate this posterior distribution. The number of particles is setting to \( n \). Every particle \( s^i_t \) corresponds to a weight \( w^i_t \). Particles should be drawn from an importance distribution \( q(s_t|x_{1:t-1},y_{1:t}) \). The weight of particle can be updated as follows:

\[
w^i_t = w^i_{t-1} \frac{p(y_t|x^i_t)p(x^i_t|y_{1:t-1})}{q(s_t|x_{1:t-1},y_{1:t})}
\]

(8)

Since the observation values are independent of the state transition in first-order Markov process, importance distribution function \( q(s_t|x_{1:t-1},y_{1:t}) \) could be simplified to \( q(s_t|x_{t-1}) \). In formula (8) , \( q(s_t|x_{t-1}) = p(s_t|x_{t-1}) \), the weights are updated as \( w^i_t = w^i_{t-1}p(y_t|x^i_t) \). \( x^i \) represents 6 affine parameters. Each dimension of \( q(s_t|x_{t-1}) \) is normal distribution. The tracking result of each frame is the particle whose weight is maximum. The main difference is the formulation of observation. \( p(y_t|x^i_t) \) represents the probability of obtaining observation value from the particle. In our method, this probability is exponentially related to the confidence of classification.

\[
p(y_t|x^i_t) \propto e^{conf}
\]

(9)

Here \( conf \) is the confidence of the observation model. In practice, we set weight \( w^i_{t-1} = \frac{1}{n} \), \( n \) is the number of particles. The 6 parameters are transformed randomly and particles are drawn from frames to build a matrix. Using observation model to get the confidence of particles, \( conf.f \) \( p(y_t|x^i_t) \) can be obtained by \( conf \) due to \( p(y_t|x^i_t) \propto e^{conf.f} \). \( w^i_t \) can be calculated by \( w^i_t = w^i_{t-1}p(y_t|x^i_t) \). The maximum \( w^i_t \) is our target.

Adaptively regulating the number of particles can speed up the tracking. Setting two thresholds based on confidence to determine the number of particles. One is a filter threshold and the other is an adaptive threshold, meanwhile the filter threshold is lower than the adaptive threshold. When the
confidence is higher than the filter threshold, the influence of decreasing the particles suitably can be neglected and we can decrease several particles appropriately. When the confidence is lower than filter threshold, the number of particles is maintained in set value to ensure the precision. When the confidence is higher than the adaptive threshold, using a classifier to select particles. This particle filter can be treated as adaptive filter. The extra classifier is set in the third layer to reduce computational cost. Particles can be divided into two categories according to the confidence. The particles with higher confidence are called positive particles where the target particle is picked from, and others are negative particles. The purpose of using the classifier is not to find the best particles but to exclude the effect of negative particles, thus screening out the approximate range. The structure of this particles filter is shown in figure 2.

**Figure 2.** The structure of our particles filter.

3. Experiments

In this section, we will describe the implementation details and parameters setting. And then, CVPR2013 Visual Tracker Benchmark [17] was selected for the test data set, which contains 50 sequences and various challenges, such as illusion variation, occlusion, out-of-plane rotation. DCT is compared with the 29 state-of-the-art trackers on this dataset. The CPU of our computer is Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz 3.30GHz, operating system is windows 8.1 64bit. And the program is implemented by matlab and C.

3.1. Implementation details

In this section, the specific parameters of the algorithm in the previous chapter are given. The parameters are set according to the experiences.

3.1.1. Offline training setting. The dataset which is used in offline training is tiny image dataset [18-19]. The dataset is collected from web by 7 search engines and covers many of objects and scenes in the real world, which contains 79302017 images stored by a binary file and the size of each image is 32 * 32. A million images randomly selected can train the network more effectively. The images are transformed to grayscale and represented by 1024 dimensional vectors. These vectors compose a matrix which is used to train the network by SDAE. The pixels of grayscale are linearly scaled to the range [0, 1]. The number of units is set to 768 in the first hidden layer and the number of units will reduce to 256 when a new layer is added.

In offline process, a tiny image dataset is used to train and the learning rate is set to 0.05. The momentum is set to 0.9 due to the network learns different categories of objects. The Gaussian noise is set to 0.0004 to generate the corrupted input. Sparsity target is set to 0.05 to keep average activation at a low level. Weight penalty is set to 0.0001 to prevent over-fitting. Mini-batch is set to 100.

3.1.2. Online training setting. In the process of tracking, the learning rate of the first hidden layer is set to 0.08. Because the appearance of the deformation compared with the initial state is not obvious, the momentum is reduced and set to 0.7. Sparsity target is set to 0.002 and noise is set to 0.003. The
learning rate of BP is set to 0.09 and the weight penalty item is 0.002. The size of mini-batch is set to 10 and threshold is set to 0.8. Particle filter draws 1000 particles.

3.2. Comparison
In this comparison, we use two performance metrics: success rate(SR) and center-of-location(COL) error. The calculation formula of SR is:

\[
\text{SR} = \frac{\text{area}(R_T \cap R_G)}{\text{area}(R_T)}
\]

where RG is the ground truth of object, RT is the tracking result. COL error is the Euclidean distance between RG and RT. The two metrics form the success plots and the precision plots. The propose three kinds of evaluation strategies: one-pass evaluation (OPE), temporal robustness evaluation (TRE) and spatial robustness evaluation (SRE). In this paper, using the OPE to evaluate our algorithm. The speed of tracking is compared in frames per second(fps). In figure 3, we compare the speed in different attributes (illumination, out-of-plane, scale, occlusion, deformation, motion blur, fast motion, in-plane rotation, out of view, background clutter, low resolution and average) by fps and our tracker is obviously outperform the DLT in all attributes. In the all attributes, the performance of DCT is almost twice than the performance of DLT.

![Figure 3. Speed plot.](image)

In figure 4, DCT shows the best performance in success plot and precision plot. In precision plot, DCT gets the best performance in the attributes of deformation, in-plane rotation, low resolution, out-of-plane rotation and out-of-view and outperforms DLT by 0.086. In success plot, DCT gets the best performance in the attributes of deformation, in-plane, low resolution, occlusions, out-of-plane rotation and out-of-view and outperforms DLT by 0.084.

![Figure 4. Average success plot and precision plot.](image)
3.3. Qualitative Results

We present the results of 12 sequences in figure 5, where original frames with 6 tracking results are illustrated. We can observe that our algorithm demonstrates outstanding performance qualitatively.

![Figure 5. Tracking results.](image)

4. Discussion

In this paper, the proposed tracker is similar to DLT, but there are still some significant differences. Firstly, the structure of the network is changed. DLT uses the over-complete basis to extract features, but the number of units in the first hidden layer of DCT is less than input, which can speed up the tracking. Secondly, we adopt different training steps. In tracking, the first hidden layer applied DAE and the higher layers applied BP, which can avoid diffusion gradient. Thirdly, the adaptive particle filter can adjust the number of particles dynamic, which can further speed up tracking.

5. Conclusions

In this paper, first a network has trained that can quickly adapt to a specific target and has initialized the observation model using the weight of the network. Then the model adapts to target and combine with particle filter for tracking. In the tracking process, the adaptive particle filter can adjust the number of the particles dynamic and the model is adjusted to adapt to the change of environment. The DAE is used to train the first hidden layer, which prevents the gradient diffusion and trains the first layer more efficiently than the BP. The results of experiments have proved that our tracker owns similar accuracy and lower computational cost than DLT on some challenging video sequences. Our method also has advantage in accuracy compared with state-of-art trackers.

Acknowledgments

This work is supported by Research project of young teachers' autonomous research project of Yanshan University (No. 15LGB014)

References

[1] Li X, Hu W, Shen C, Zhang Z, Dick A and Hengel A V D 2013 ACM transactions on Intelligent Systems and Technology 4(4) pp 58-100
[2] Ross D A, Lim J, Lin R S and Yang M H 2008 International Journal of Computer Vision 77(1) pp 125-141
[3] Kwon J and Lee K M 2010 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp 1269-1276
[4] Mei X and Ling H 2009 Proceedings of IEEE International Conference on Computer Vision (ICCV) pp 1436-1443
[5] Mei X, Ling H, Wu Y, Blasch E and Bai L 2011 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp 1257-1264
[6] Bao C, Wu Y, Ling H and Ji H 2012 *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pp 1830-1837

[7] Liu B, Huang J, Kulikowski C and Yang L 2013 *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(12) pp 2968–2981

[8] Avidan S 2007 *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(2) pp 261-271

[9] Grabner H, Grabner M and Bischof H 2006 *Proceedings of British Machine Vision Conference (BMVC)* pp 47-56

[10] Kalal Z, Mikolajczyk K and Matas J 2012 *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34(7) pp 1409-1422

[11] Hare S, Saffari A and Torr P H 2011 *Proceedings of IEEE International Conference on Computer Vision (ICCV)* pp 263-270

[12] Wang N and Yeung D Y 2013 *Proceedings of Advances in Neural Information Processing Systems (NIPS)* pp 809-817

[13] Doucet A, Freitas D N and Gordon N 2001 *Sequential Monte Carlo Methods In Practice* Springer

[14] P Vincent, H Larochelle, I Lajoie, Y Bengio, P A Manzagol 2010 *The Journal of Machine Learning Research* 11 pp 3371-3408

[15] Kuen J, Lim K M and Lee C P 2015 Pattern Recognition 48(10) pp 2964-2982

[16] Hong S, You T, Kwak S and Han B 2015 *Proceedings of International Conference on Machine Learning (ICML)* pp 597-606

[17] Wu Y, Lim J, Yang M H 2013 *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pp 2411-2418

[18] Torralba A, Fergus R and Freeman W T 2008 *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30(11) pp 1958-1970

[19] Li P X, Wang D, Wang L J and Lu H C 2018 *Pattern Recognition* 76 pp 323-338