Deep Reinforcement Template Matching

Onkar Krishna[1] Go Irie[1] Xiaomeng Wu[1] Takahito Kawanishi[1]
Kunio Kashino[1]

Abstract

Template matching is a fundamental problem in computer vision where the task is to find a region that matches a given query. The exhaustive nature of the sliding window approach has encouraged works to reduce the run time by pruning unnecessary regions. However, such a pruning-based approach still needs to evaluate the non-ignorable number of candidates, which leads to a limited improvement of the balance between speed and accuracy. In this paper, we propose a deep reinforcement learning approach to predict efficient search paths from data. Our approach uses both of the localization-based reward and feature matching quality to train a CNN-LSTM network, which allows us to jointly learn the search paths together with deep image features for matching. Evaluation results demonstrate that our approach achieves significantly better accuracy than existing template matching methods, with highly comparable search speed.

1. Introduction

The task of template matching is to find a part of a reference image that matches a query image, which is essential in a wide variety of applications in computer vision research field such as registration, verification, tracking, compression, and stitching. A desirable algorithm should be *robust* and *fast* enough; it can find the correct matches under the potential variations such as background clutter, illumination changes, occlusions, and geometric transformations and deformations, within a reasonable time budget.

The straightforward approach would be exhaustive search with a certain similarity measure such as normalized cross-correlation (NCC). However, it is often prohibitive due to the vast number of candidates needed to be evaluated. Many efforts have been made for improving the search speed without loosing accuracy. For example, several studies [6], [9], [12] propose to skip unlikely candidate windows or pixels that will not change the final result as much as possible. [5] represents an image by a linear combination of binary images called slices to enable fast matching. [3] proposes to use best-buddies similarity (BBS) that measures the similarity between two image regions based on a small subset of pixel pairs. An extended version of this approach is presented in [11]. Overall, these existing methods still need to evaluate a large number of undesirable candidate regions, which leads to limited improvement of the balance between speed and accuracy.

In this paper, we propose a novel approach to template matching. Our idea is to predict efficient search path from data, i.e., instead of skipping unlikely windows or pixels as done in most of the existing methods, we use machine learning to pick and evaluate only the highly prospective regions of the reference image. Our method uses a CNN-LSTM network model to sequentially search promising regions over the reference image that are expected to match the query. The network is trained through reinforcement learning in an exploratory manner; explicit supervised information such as the location of the target regions or class labels is not necessary. Furthermore, our model jointly learns image features in an end-to-end manner to improve matching accuracy. We evaluate our method on MNIST and FlickrLogos-32 datasets and show that it gives remarkable matching accuracy with comparable to or faster search speed than existing methods.

2. Method

Suppose we are given a pair of a query and a reference images denoted by $\mathcal{Q}$ and $\mathcal{R}$, respectively. Our task is to localize the target region on $\mathcal{R}$ that matches $\mathcal{Q}$. We assume that the pose of the target region is specified by a linear transformation A_g on $\mathcal{R}$ and denote by $\mathcal{R}(A_g)$ the target region of $\mathcal{R}$ determined by A_g. In this paper, We restrict $\mathcal{A}_g$ to be affine transformation, hence $A_g \in \mathbb{R}^{2\times 3}$.

We approach to this task by sequential search. More specifically, our goal is to determine the sequence of the poses of the search window $\{A_t\}_{t=0}^{T}$ on $\mathcal{R}$ so that it can correctly localize the target region $\mathcal{R}(A_g)$ with small T, where T is the length of the sequence. We propose a machine learning approach using a CNN-LSTM neural network. The schematic overview of our model is illustrated as in Fig. 1. Our model consists of two major modules which we call *feature extraction module* and *localization module*, respectively. The basic behavior of our model at each time step t can be summarized as follows. The feature extraction module first extracts the image features of $\mathcal{Q}$ and $\mathcal{R}(A_t)$ which are denoted by $f(\mathcal{Q})$ and $f(\mathcal{R}(A_t))$, respectively. The similarity between $\mathcal{Q}$ and $\mathcal{R}(A_t)$ is measured by using these features. Then the two image features $f(\mathcal{Q})$ and $f(\mathcal{R}(A_t))$ as well as the current pose of the search window A_t are fed

[1] NTT Corporation

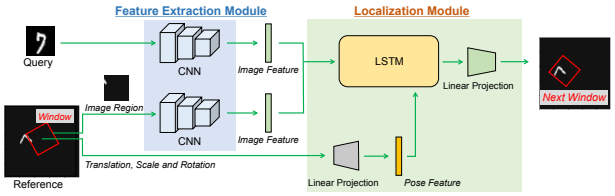

Fig. 1 Overview of our proposed model. Our model has two major modules: feature extraction module and localization module.

to the localization module to determine the next pose of the window A_{t+1}. These two sub-processes are repeated sequentially until the maximum number of iterations T is reached. We detail the two modules in Sec. 2.1.

Training of our model is somewhat tricky. Since the localization is performed in a sequential manner and the current decision at a time is made depending on all the past decisions, the quality of the decision at each time step cannot be evaluated independently from the others. Therefore, typical supervised or unsupervised learning methods that assumes i.i.d. samples cannot be applied to our case. Fortunately, the process can be modeled as a partially observable Markov decision process (POMDP), so the training can be performed based on reinforcement learning. In this paper, we propose a reinforcement learning method that can jointly learn image features as well as search paths (sequence of the window poses) in a unified framework. Our algorithm is inspired by the recurrent attention models which are proposed for image recognition [1], [2], [8]. Unlike these, our model is customized to template matching task and designed to learn image features effectively for similarity matching. Another important difference is that ours does not require any class label information for training. Furthermore, unlike our recent method [7] that can only learn and estimate the position of the target region at a fixed scale, the model presented in this paper is extended to consider the full affine transformation of the target. We give the detail of our algorithm in Sec. 2.2

2.1 Details of Modules

Fig. 1 shows the brief overview of our model. It is basically a CNN-LSTM model and uses the CNN for *feature extraction* and LSTM for *localization*. We hereafter give the details of these two major modules one-by-one.

Feature Extraction Module. This module extracts the image features from $\mathcal{Q}$ and $\mathcal{R}(A_t)$. It consists of two identical fully-convolutional CNNs with the parameters shared; one for $\mathcal{Q}$ and the other for $\mathcal{R}(A_t)$. The CNN is designed to have a sequence of five Conv-ReLU layers (ReLU activation after 2D convolutions) followed by a global average pooling.

Localization Module. The main component of the localization module is LSTM that sequentially predicts the next pose of the window A_{t+1} based on three external inputs including the two image features $f(\mathcal{Q})$ and $\mathcal{R}(A_t)$ and the current pose of the window A_t. These three inputs are concatenated to form a single vector and then fed to the LSTM. The dimension of the pose parameters (transformation) A_t is often far smaller than that of the image features (typically > 128). In order to mitigate this imbalance, A_t is first encoded by a linear projection into a *pose feature* which has the same dimension as the image feature and then concatenated. Another linear projection is applied to the hidden state h_t of the LSTM to predict the next pose of the location $\hat{A}_{t+1}$. We assume that A_{t+1} is a stochastic variable that follows a Gaussian distribution whose mean is given by $\hat{A}_{t+1}$. More specifically, A_{t+1} is obtained as a sample from the distribution $\mathcal{N}(\hat{A}_{t+1}, \lambda I)$ as $A_{t+1} \sim \mathcal{N}(\hat{A}_{t+1}, \lambda I)$, where I is the identity matrix and λ is a hyperparmeter.

The initial pose of the window A_0 is determined by a similar way used in [1], [2]. Specifically, we first extract the image features of the (down-sampled) global reference image, $f(\mathcal{R})$, and then feed it to another linear projection which is analogous to the context network used in [1], [2] to generate A_0.

2.2 Model Training

Let $\Theta = \{\theta_f, \theta_l\}$ be the parameters of the whole model, where θ_f and θ_l are the parameters of the feature extraction module and the localization module, respectively. We use reinforcement learning to tune Θ. For notational simplicity, we use $s_{t-1} = \{\{A_\tau\}_{\tau=1}^{t-1}, \mathcal{Q}, \mathcal{R}\}$. The *policy* of our model then can be represented as a conditional distribution $\pi(A_t|s_{t-1};\Theta)$. Our goal is to maximize the total reward $R = \sum_{t=1}^{T} r_t$ w.r.t. Θ, where r_t is the reward value at time t. The expected value of the total reward is given as

$$J(\Theta) = \mathbb{E}_{p(s_T;\Theta)}[R], \tag{1}$$

where $p(s_T;\Theta)$ is the probabilistic distribution of s_T which depends on the policy. Although the gradient w.r.t. Θ is non-trivial, it can be approximately computed by sampling the sequences of $\{A_t, s_{t-1}\}_{t=1}^{T}$ from the policy in a similar way to Monte-Carlo approximation, which gives

$$\nabla_\Theta J(\Theta) \approx \frac{1}{M} \sum_{i=1}^{M} \sum_{t=1}^{T} \nabla_\Theta \log \pi(A_t^i|s_{t-1}^i;\Theta) R^i. \tag{2}$$

where M is the number of sample sequences. By using this, Θ can be iteratively updated through gradient ascent.

Training our model only with reinforcement learning often makes the prediction results unstable. Hence, we involve another loss function to improve the stability of the learning process. Specifically, we require the image features extracted by the feature extraction module to correctly measure the similarity between $f(\mathcal{Q})$ and $f(\mathcal{R}(l_t))$, i.e., the distance between the two features should be small for matching pairs and large for non-matching pairs. To this end, we impose a loss function which resembles contrastive loss [4] on the feature extraction module.

$$L(\theta_f) = \sum_{t=1}^{T} r_t d^2 + (1 - r_t) \max\{0, m - d\}^2, \tag{3}$$

where $d = ||f(\mathcal{Q}) - f(\mathcal{R}(l_t))||$ and m is a margin. Unlike the original loss function used in [4], ours uses reward r_t to determine the loss value. This is piecewise differentiable w.r.t.

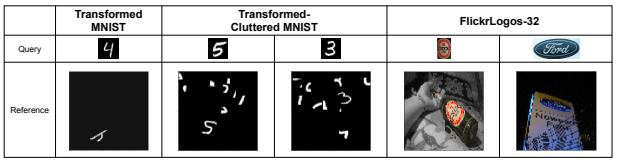

Fig. 2 Examples of the query-reference pairs used in our experiments.

θ_f and so can be easily optimized with gradient descent.

3. Experiments

3.1 Datasets

We use two benchmark datasets, MNIST[*1] and FlickrLogos-32[*2] [10], in our experiments for evaluation. Examples of the query-reference pairs are shown in Fig. 2.

MNIST. For our experiments, we prepare two noisy variations based on the original MNIST dataset. The first one is **Transformed MNIST**, in which each reference image is generated by applying a random affine transformation (translation, scaling, and rotation) to each MNIST training image to be of size 80×80. The second one is **Transformed-Cluttered MNIST** in which each reference image is generated by adding 9×9 patches randomly extracted from other MNIST training images to random locations of the Transformed MNIST reference image. All query-reference pairs are prepared for these two MNIST variants by assigning a 28×28 query image of the same digit to each of 80×80 reference images. The query image is selected from a master set consisting of 10 original MNIST images. We follow the official split for obtaining training and testing sets.

FlickrLogos-32. This dataset consists of $2,240$ images of 32 different logos with 70 images per logo. In our experiments, the training and the testing subsets are composed of $2,000$ and 240 query-reference pairs, respectively. Each pair is generated by considering a logo image in the dataset as the reference and a tightly cropped logo of the same brand as the query. In total, we have 32 query images, each corresponding to an individual logo. All the reference images were resized to 80×80 and the corresponding ground truth bounding box coordinates are also resized.

For both of the datasets, the query and the corresponding ground truth region are of the same category (digit or logo) but are not exactly the same, which is more realistic compared to the traditional template matching scenarios.

3.2 Experimental Setup

Performance Metrics. We evaluate our approach in terms of accuracy and speed. Given a query and a reference image, our model outputs a predicted pose of the window corresponding to a region in the reference image, which is used to evaluate the accuracy. In particular, an image matching is considered as being successful if the intersection over union (IoU) between the predicted pose of the window and the ground-truth window is greater than 0.5.We report the success rate which represents the ratio of the number of image pairs with correct matches to all the pairs. The efficiency is evaluated in terms of two measures. One is the number of windows evaluated for matching, and the other is run time required for processing each query-reference pair.

Table 1 Success rate.

Dataset	Transformed MNIST	Transformed - Cluttered MNIST	Flickr Logos-32
Ours	0.89	0.85	0.34
MTM [5]	0.51	0.18	0.10
BBS [3]	0.56	0.20	0.31

Baselines. We compare our method with two representative template matching methods, namely BBS [3] and MTM [5]. Furthermore, the conventional key-point matching methods such as SIFT and BRISK can also be considered for baseline in future extension of this work.

Learning Configurations. We trained the proposed model from scratch using Adam with a batch size of 64 for MNIST and 1 for FlickrLogos-32. The learning rate was kept in the range $[10^{-4}, 10^{-3}]$ with an exponential decay. The variance hyperparameter of the Gaussian λ, which is used for sampling out the next location, is fixed to 0.22. In the current implementation, the maximum number of iteration T is fixed to 6. In future extension, we are targeting investigate the balance between the efficiency and the number of skipped windows by adaptively learning T for each input pair of query and reference image

Regarding the reward r_t in our algorithm is determined based on the success or failure of the localization at time t. For MNIST where we have the exact ground truth pose (A_g), we first decompose the affine matrix into translation, scale, and orientation factors, and evaluate each separately by checking whether the absolute error is within a certain threshold or not; we give $+1$ if and only if the error is within the threshold, and 0 otherwise. The thresholds used are 0.5, 0.3, 0.3 for translation, scale, and orientation, respectively. For FlickrLogos-32 where we could not know the exact ground truth pose, we determine the reward based on the IoU value and give $+1$ if and only if it exceeds 0.5.

3.3 Results

For all the datasets, the success rate, the number of windows evaluated, and the run time are reported in Tables 1, 2 and 3, respectively.

Results on Transformed MNIST. First, as can be seen in the Table 1, the success rate of our method is the best among all the methods. The gain of our method over the baselines reaches 0.33 to BBS and 0.38 to MTM. Second, as shown in Table 2, our model evaluates only 6 candidate windows to achieve the accuracy, which is clearly fewer compared to those of the other two baseline methods. Looking at the run time reported in Table 3, although it is not directly proportional to the number of windows processed since each

[*1] http://yann.lecun.com/exdb/mnist/
[*2] http://www.multimedia-computing.de/flickrlogos/

Table 2 Number of windows evaluated to localize a query.

Dataset	Transformed MNIST	Transformed-Cluttered MNIST	Flickr Logos-32
Ours	6	6	6
MTM [5]	2809	2809	2809
BBS [3]	6400	6400	6400

Table 3 Average run time in milliseconds.

Dataset	Transformed MNIST	Transformed-Cluttered MNIST	Flickr Logos-32
Ours	3.8	4.0	26.2
MTM [5]	1.1	1.0	5.2
BBS [3]	90.1	90.3	110.6

methods have different computation requirement for every pixel, ours is competitive to MTM and is much faster than BBS, while yielding significantly better matching accuracy.

Results on Transformed-Cluttered MNIST. As shown in Table 1, our method is the best among all the methods in terms of success rate in matching. BBS and MTM performs poorly on Transformed-Cluttered MNIST. MTM first decomposes an image into a set of binary images and measures the similarity based on these binary bases, which may be difficult to distinguish the target from the noisy background. In BBS, the matching between query and candidate windows is evaluated according to the consistency of the distributions of pixels; two windows were determined as a matching pair if their pixel distributions in (x, y, R, G, B) space are similar. This strategy is not effective on cluttered MNIST where the noise may have the same underlying distribution as the target. Unlike these two, ours can accurately localize the query object in just 6 candidate windows.

Results on FlickrLogos-32. This is the most challenging dataset because the targets are different from the queries due to a wide variety of scale changes, viewpoint changes, and deformations. Table 1 shows that our method outperforms all the baselines in success rate. Although MTM is faster than ours, the accuracy is far from satisfactory. When ours is compared with BBS, they are competitive in terms of success rate but ours is sufficiently faster than BBS. These results suggest that our method better balance between speed and accuracy, and template matching based on reinforcement learning actually works well on such a real dataset.

Qualitative Results. Figure 3 shows some qualitative results. It demonstrates the excellent ability of our model in learning the search path. The results on MNIST datasets show that despite the increased level of search difficulty due to clutter, our model still can successfully localize the query object within a few number of candidate windows evaluated. Similarly, from the FlickrLogos-32 examples, it can be seen that our method can successfully find the target logos even if they are heavily different in their colors and poses. For instance, in the case of the right most example, ours successfully localizes the target despite the large geometric transformation between query and reference images. Furthermore, in the second example from the left, the model can successfully localize the target region, despite of

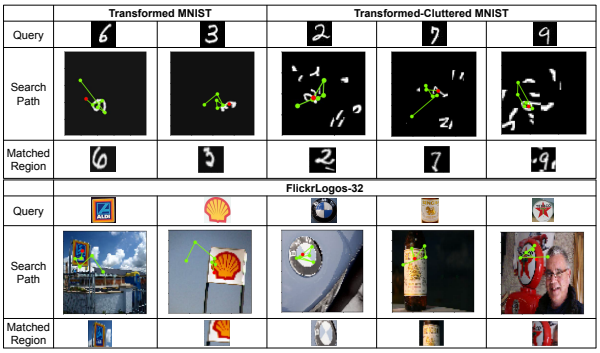

Fig. 3 Qualitative results. For a given query-reference pair, the example shows the search path traced by our model in order to localize the query. The search path is overlaid in green lines. The red dot indicates the predicted location at the last time step. The window cropped from the last attended location is the search output of our model.

starting the search at texture-less region of the image (sky). This can be possible because our model learned to pick the next location randomly until a part of the target region is captured in the search window, once it captured the model converges rapidly to the target location. However, there are a few cases when the model can not capture any part of the reference region within the 6 search iteration and fails to localize.

4. Conclusions

We proposed a reinforcement learning approach to template matching that can jointly learn search path in the form of a sequence of affine transformations as well as deep image features in an end-to-end manner. Evaluation results suggested that our approach can achieve significantly better matching accuracy than existing methods with highly competitive search speed.

References

[1] Ablavatski, A., Lu, S. and Cai, J.: Enriched Deep Recurrent Visual Attention Model for Multiple Object Recognition, *Proc. WACV* (2017).
[2] Ba, J., Mnih, V. and Kavukcuoglu, K.: Multiple Object Recognition with Visual Attention, *Proc. ICLR* (2015).
[3] Dekel, T., Oron, S., Rubinstein, M., Avidan, S. and Freeman, W. T.: Best-buddies similarity for robust template matching, *Proc. CVPR* (2015).
[4] Hadsell, R., Chopra, S. and LeCun, Y.: Dimensionality Reduction by Learning an Invariant Mapping, *Proc. CVPR* (2006).
[5] Hel-Or, Y., Hel-Or, H. and David, E.: Fast template matching in non-linear tone-mapped images, *Proc. ICCV* (2011).
[6] Korman, S., Reichman, D., Tsur, G. and Avidan, S.: Fast-match: Fast affine template matching, *Proc. CVPR* (2013).
[7] Krishna, O., Irie, G., Wu, X., Kawanishi, T. and Kashino, K.: Learning Search Path for Region-Level Image Matching, *Proc. ICASSP* (2019).
[8] Mnih, V., Heess, N., Graves, A. and Kavukcuoglu, K.: Recurrent Models of Visual Attention, *NIPS* (2014).
[9] Pele, O. and Werman, M.: Accelerating pattern matching or how much can you slide?, *Proc. ACCV* (2007).
[10] Romberg, S., Pueyo, L. G., Lienhart, R. and Zwol, R. V.: Scalable logo recognition in real-world images, *Proc. ICMR* (2011).
[11] Talmi, I., Mechrez, R. and Zelnik-Manor, L.: Template matching with deformable diversity similarity, *Proc. CVPR* (2017).
[12] Vinod, V. V. and Murase, H.: Focused color intersection with efficient searching for object extraction, *Pattern Recognition*, Vol. 30, No. 10, pp. 1787–1797 (1997).