A Siamese Network for generating oriented bounding box in object tracking

Yuchen Yang, Dong Huo

yy17@ualberta.ca, dhuo@ualberta.ca

Abstract

We focus on generating oriented bounding boxes for object tracking using Siamese network. Recent deep leaning approaches utilize the Siamese network have achieved promising results with high speed and accuracy. Noticed that some traditional methods are capable of generating a bounding box with orientation, for deep learning methods, this remains an unsettled problem with only few research have discussed. In this research, we try to utilize the Siamese network to estimate an orientated bounding box that better describe the tracking object. We explore several naïve pipelines include: (1) using the Siamese network to directly estimate the warp matrix for the transformation between the object in the template and the search window. (2) directly estimate the 4 coordinates of the tracking target in the searching window. (3) estimate the warp indirectly by warping the template corner coordinates to the target area in search windows. Though our observation, the third one is trained and tested well while others fail in different phases. Due to limited time and resource, we train our network in VOT 2016 and test on a subset of VOT 2014. The performance is evaluated by the expected average overlap (EAO). Our method shows potentials to compete with other advanced approaches.

Key words: *Object Tracking, Oriented bounding box, Deep learning, Siamese Network, Warp matrix, Coordinates estimation*

1. Introduction

Object tracking is an important field of study in Computer vision. Given a tracking object (template), it requires an algorithm to give out its locations in consecutive frames in a video sequence. It is useful in many applications such as robot vision and security cameras. The challenges for a tracking algorithm differ in the areas of applications. In this paper, we mainly focus on tracking the general objects, which indicates non-rigid transformations are involved when tracking the target. Methods [1,2,3,4,6] use Siamese Network for tracking is recently developed. They utilize the strong representation power of deep learning and achieved great speed and accuracy on datasets such as VOT. However, these methods can only generate axis-aligned bounding boxes for objects, and commonly cannot describe an object well. The recent developed SiamMask [5] is the method that can generate orientated bounding box, it utilizes the segmentation label provided from the dataset and use it to estimate a segmentation mask in the Siamese network, then they generate the orientated box based on the min and max directions of the mask. Though it can generate the orientated box and achieve great performance by utilizing the multi-tasks learning, the segmentation mask for a dataset is usually not easy to obtain. In this research, we explore the possibility to obtain an oriented bounding box with only rotated bounding box labels provided (see

VOT 2016). We believe this might open an opportunity for a more accurate tracker using Siamese network.

In our research, we explore the use of Siamese network in generating the oriented bounding box. We explore three pipelines in learning the orientated bounding box. First, given the inspiration of recent holography estimation using deep learning [9,10], we apply the Siamese network to directly estimate the warp matrix between the target in the template and in the search window. Second, we regress on the relative coordinates in the search window without the concern on the transformation and the constraints between coordinates. This approach is much similar to the object detection. Third, we estimate the warp from the network but we regress on the transformed corners from template to their real coordinates in the search image by using this warp. The first method fails on learning the parameters (training phase). The second one has a significant small loss during training phase but fails to locate the target in testing phase. The third pipeline, though it is still a naïve approach and has its defections, we found it outperforms the other pipelines and has the potential for improvements. Details and discussions of these pipelines will be described in the following sections.

2. Related work

Neural networks have been used for tracking objects for several years. Recently, the developments of approaches that use the Siamese network is proved to be both accurate and fast. The first paper in this pipeline [6] use a Siamese network to compare the features from a previous frame to current frame, and regress on the final target in the current frame. It reaches 100FPS on VOT 2014 dataset. Paper [2] is the first method that combines the similarity learning and sliding window approach, which is a milestone for Siamese network-based tracking method. Afterward, paper [3] proposed to borrowed the advanced region proposal network from the field of object detection while using paper [2] as its basic structure. Paper discussed the reduction of the distractors in tracking using Siamese network by introducing semantic negative pairs and data augmentation strategies. Paper [5] proposed Siammask network which is the first deep learning method that is capable of generating rotated bounding box. However, it requires segmentation labels during training phase. These papers show the advantage of using Siamese network for tracking.

Another topic of research we think is related to our research is warp estimation using deep learning. Methods [9,10] use deep neural network to estimate the warp between two images. These networks are usually trained on a synthetic dataset in which they apply randomly transformations on original image and feed the network with the original images. And the network is used to regress on the randomly generated transformation matrices. One can see the resemblance with the training stage of Jurie & Dhome tracking method [11] in which they also apply the same dataset generation step on image. However, current methods focus on the homography estimation are not performed well if we directly apply them to track object (see Chen's report [13]). We argue that this might because the main content is not changed when we apply this training set generation, while the realistic task enriched with rigid and non-rigid changes and background changes, which is far more complicated. In our research (first pipeline), we consider the possibility that if we can estimate the warp between frames by changing the training set generation.

3. Methodology

In this part, we demonstrate our methodology. First, we will introduce our overview pipeline. Then, we described the network structure we use and three training strategies we explore.

3.1 Overview of warping pipeline (Dong Huo)

The overview pipeline can be seen in Figure 1 and Figure 2. Similar to any tracking methods, we have a template image and a search window. Notice that normally in other deep learning methods, they crop a rectangle area in the first frame based on the label (coordinates of four points) of ROI (region of interest) as the template image. Here we consider the given template is not axis-aligned but simply 4 points coordinates as we mentioned before. This fits better with the dataset that has orientated box label such as the VOT and TMT dataset, and it is potentially better to describe the tracking target since it includes less background information, which is not a part of the tracking target. To unify all scenarios, given 4 points coordinates of template area, we warp them to 4 corners of a rectangle area with size of 224x224 to generate a new image, and use this image as our template image. It is distorted sometimes because the shape around four points are not a square, but the impact of this distortion can be ignored. We annotated this warp as W_1. Not only crop the template image on the first frame, we crop it on all of the frame and save it in the same folder. In this case, we can train a more robust model. The reason why we crop the search window is that original images are too large, and we want the model to focus more on the ROI rather than other background pixels. Methods of generating template images are the same both in training process and testing process, but the way of generating search windows are different.

As shown in Figure 1, for training part, a search window image is generated by cropping and expanding 50% area on the height and width based on the target coordinates on the last frame which are extracted from label file, then warp to the 224x224 rectangle size as well. Note that if the expansion is beyond the border of the image, we pad the image with the average value of the search window to the area outside of it (same techniques in papers [2,3,4,5]). We can use warp W_2 to denote this process. In practice, we see that simply crop search windows with 50% of extension area is not enough, because sometimes the object is moving really fast and size changed a lot between adjacent frames. In order to train a more robust model, we add two parameters *scale* and *translate* to change the size and position of the bounding box, it is a way to augment the data. After cropping the search windows, we warp it to 224x224 as well.

As shown in Figure 2, for testing part, we crop the first search window from the second frame based on the coordinates on the first frame as well and record the warp matrix W_2. After we construct the template and search window images, we feed them into the Siamese network to get the relative coordinates of the target in search window, we can denote this process inside Siamese network as a warp estimation W_{sn}. Hence, a forward pass for generating the final absolute coordinates in current frame is to simply calculate by warping:

$$\mathrm{coordinates}_{abs} = {W_2}^{-1} * (\mathrm{W}_{sn} * \mathrm{meshgrid}(1,224))$$

Then we regard it as the true coordinates to crop next search window from the next frame. After this warping operation, we are able to collect the absolute coordinates for the target in current frame and generate the search window for next frame. Repeat the same work until we

generate the coordinates on all of the frames. In this part, we do not need to scale or transform the bounding box.

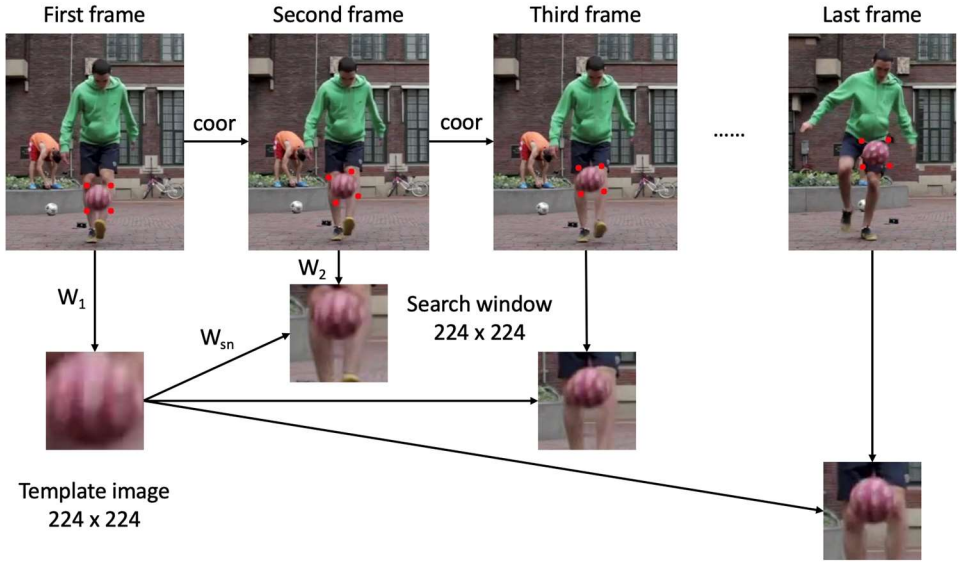

Figure 1. The warping operation overview of training process.

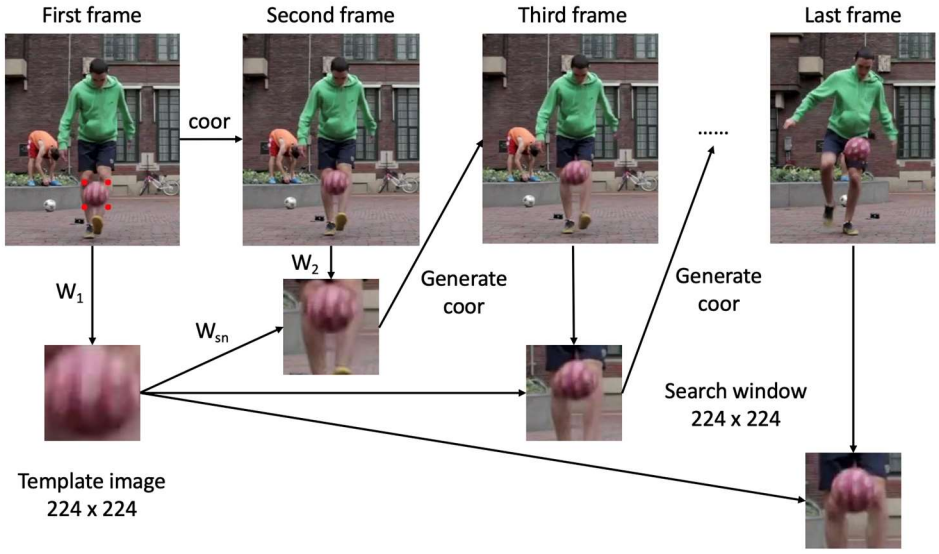

Figure 2. The warping operation overview of testing process.

To train a Siamese network to either learn the warp or the relative coordinates, we can utilize the W_1, W_2 and calculate the true W_{sn} by DLT algorithm based on the 4 points correspondence in the template and the target labeled location in search window. Besides the warp calculation from label information, we applied few augmentations as we mentioned above on the search window includes scale variation and translation variation.

3.2 Siamese network structure (Yuchen)

An overview of our model can be viewed in Figure 2. The network structure can be viewed as a shared weights CNN backbone (with two branch), a feature concatenate operation and fully connected layers for regressing the target. Given a template image with a size of 224x224, and a search window image with a size of 224x224, each branch accepts the image and generate a vector with a length of 256 * 6 * 6. We call this step as feature extraction, since it transfers the image into a fixed length vector. Then the concatenate process simply attaches these two vectors together to a new vector with a length of (256 * 6 * 6) * 2. The final FC layers forms a multilayer regressor that regress the target vector to the given label. Here, we do not give out the length of the final target vector (the 'len' in Figure 2). The details of the learning target are given in the next section.

Note that we choose this concatenation on features (used in papers [1,6]) rather than cross-correlation (used in papers [2,3,4,5]). We call the Siamese network with concatenation as the naïve approach. The main reasons for choosing such model are: (1) This framework works for paper [1,6] in object tracking task. We are confident that it can perform well with a well-designed loss function. (2) The network can be simply implemented, since it takes the same size of inputs for both template and search window branch. The downside for choosing this naïve network is that it lacks constraints on the weights of the first fully connected layer. It is possible to lead to a situation where the weights that correspond to the vector generated from template branch might be very small (close to zero), hence affect the generalization ability of the Siamese network. More advanced papers [2,3,4,5] utilize fully convolution layer and cross correlation which mitigate this problem. However, the paper [2,3,4,5] that use the cross-correlation is slower than the naïve approach (typically decrease the speed from 100 fps to 30fps). Based on our limited resource and time, we choose the naïve approach with AlexNet as our backbone structure for the Siamese network since it costs less memory and has faster speed when running. We copy the weights of the first five convolution layers (feature extraction layers) of the model trained from ImageNet and use it as our initial weights. The fully connected layers are initialized randomly.

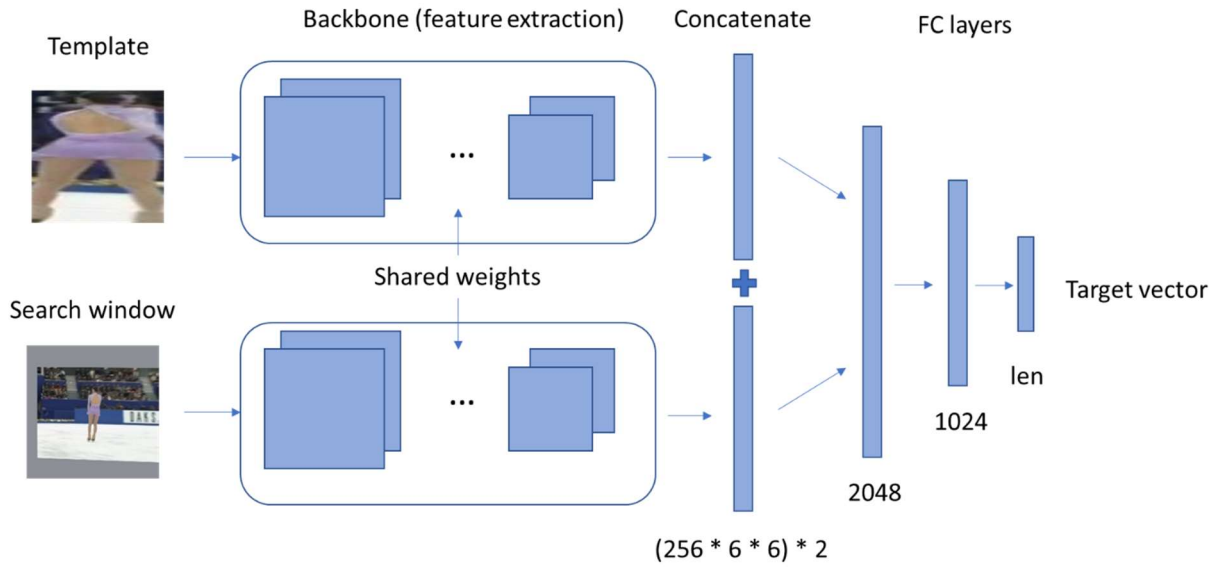

Figure 2. The Siamese network structure overview.

3.3 Three learning strategies (Yuchen)

As we discussed earlier, we tested three pipelines in learning the location of the template image in the search window. First, we want to see if we can directly estimate the warp matrix between the 4 corners of the template and the true 4 coordinates in the search window. This means learning the parameters of any following warp matrices:

$$W_{similarity} = \begin{bmatrix} p1 & -p2 & p3 \\ p2 & p1 & p4 \\ 0 & 0 & 1 \end{bmatrix},\ W_{affine} = \begin{bmatrix} p1 & p2 & p3 \\ p4 & p5 & p6 \\ 0 & 0 & 1 \end{bmatrix},\ W_{homography} = \begin{bmatrix} p1 & p2 & p3 \\ p4 & p5 & p6 \\ p7 & p8 & 1 \end{bmatrix}$$

Hence the target length in Figure 2 is automatically adjusted to fit for the number of parameters to learn. We apply the l2 Euclidean distance between these parameters and the true warp parameters W_{true} calculated from the DLT algorithm as the loss function:

$$Loss = \left\| W_{true} - W_{affine} \right\|_2$$

This loss fits with the description in paper [9] in homogenous estimation using deep learning. Since the template and the search window potentially has a higher degree of freedom transformation, we only give attention to learning the latter two matrices. What we observe is that when we train the parameters directly is that it does not seem to converge well during our several training attempts. Interestingly, by extracting the learned target, we see that only the p3 and p6 in the affine and homography parameters are well learned while the other parameters are far away from its true value (calculated from DLT). We argue that the parameters of the rotation and scale might not be easily obtained by minimizing this Euclidean distance directly.

The second strategy we use is to directly learn the coordinates in the second image. This strategy is similar to the SiamRPN, in which they apply the RPN [7] from object detection to regress on the final coordinates. However, it learns the center of an aspect-ratio specified anchor and the relative height and width of that anchor, which means the structure of RPN itself restrict the bounding box as in axis-aligned bounding box style. In order to regress an orientated bounding box, we simply regress the 4 coordinates $Coords_r = \{X1, Y1, X2, Y2, X3, Y3,\ X4, Y4\}$ in the search window. We find the l2 loss during the training phase can be significantly small, however, when we apply this to real video sequence rather than the pairs of images in the training set, it fails completely on tracking the target area. This suggests severe overfitting problem.

The third pipeline is proposed based on the observation of the STN [8]. In STN, a warp is learned in the network indirectly. It was originally proposed to be used in classification task. The learned warp is applied to the feature/image itself and normalize the locations of all incoming features/images to a rectified position for a single category. The warp is indirectly learned inside the network guided by the final classification loss only (the cross-entropy loss). Paper [1] utilizes the STN for tracking objects. It minimizes the subtraction value between the warped search window and the template. This minimization is similar to the basic lucas-kanade method, they share the same assumption that the color of the area is not significantly changed. However, in a non-rigid motion, this subtraction is not necessarily valid. Thus, our third pipeline is to learn the warp indirectly. and we generate the warp parameters as the target vector of the network then we

warp the 4 corners of the template coordinates to the search window. We minimize the l2 loss between the true coordinates and the warped coordinates. That is to learn the $Coords_r$ in:

$$\begin{bmatrix} p1 & p2 & p3 \\ p4 & p5 & p6 \\ 0 & 0 & 1 \end{bmatrix} \quad * \quad \begin{bmatrix} 0 & 0 & 223 & 223 \\ 0 & 223 & 0 & 223 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad = \quad \begin{bmatrix} X1 & X2 & X3 & X4 \\ Y1 & Y2 & Y3 & Y4 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$W_{affine} \qquad 4\ corner\ Coordinates\ of\ the\ template \qquad Coords_r$$

Another way of viewing this is that we put constraints to these parameters. If we take another step on the left side on this equation, we can see the regression is actually learning:

$$\begin{aligned} p3 -> X1, \qquad & p6 -> Y1, \\ 223 * p2 + p3 -> X2, \qquad & 223 * p5 + p6 -> Y2, \\ 223 * p1 + p3 -> X3, \qquad & 223 * p4 + p6 -> Y3, \\ 223 * (p1 + p2) + p3 -> X4, \qquad & 223 * (p4 + p5) + p6 -> Y4. \end{aligned}$$

Here, we simply rewrite this to minimize the loss:

$$Loss = \left\| Coords_{r\_true} - Coords_r \right\|_2$$

$$Coords_r = W_{affine} * meshgrid(0,223)$$

We think the reasons why this is a success compared with the first two learning strategies is that: the first strategy tries to learn a single warp matrix which actually may not be the best because the DLT method does not guarantee a 'best' warp matrix. And the first strategy let the network learn the parameters directly without knowing the hidden relations between these parameters. The second strategy only pays attention to the four corner points matching, it does not consider the points inside the 4 corner points. All these constraints are considered in the third solution in the way of learning the warp indirectly. In the third strategy, we give certain freedom on generating the warp (do not regress on a specific warp), and certain constraints that consider an area of points (we think the warping process is equivalent to consider the shape changes in an area) rather four independent points. And through our experiments in training and testing this strategy, we see that it converges well and can be applied to the real video sequences in both training and testing set.

4. Experiments

In the following paragraphs, we refer the 'model' as the third strategy we mentioned above (since it is the only one that can be successfully trained and tested). We implement our model using Pytorch. The GPU we use is Nvidia GTX 980m 8G. The CPU is core i7. We train our model on the VOT 2016 dataset [12]. It contains 60 sequences of videos with 21455 frames. We choose this dataset because (1) It contains rotated bounding box notations. (2) The number of categories of the objects in this dataset is relatively rich. (bird, human, car, etc.). We trained our model on the full VOT 2016 dataset using Adam gradient descent with learning rate 0.0001, batch size of 64 and 100 epochs. The warp parameters we learn here follows the affine transformation (6 DOF). We test our method on 18 videos from VOT 2014 dataset that does not appear in the VOT 2016 dataset. The performance is evaluated using EAO (expected average overlap), which calculate the average of IOU (intersection of union) between the true absolute coordinates in the current frame and the estimated absolute coordinates in all frames of a video, then averaged over all videos. The search window is generated based on the tracking result of the algorithm in the last frame by expending

50% area on the tracked area. If the algorithm fails to track the object, no measure will be done. This criterion follows the definition of EAO measurement in VOT datasets.

train (VOT 2016)	test (subset of VOT 2014)
0.7327	0.223

Table 1. The EAO of our trained model tested on VOT 2016 (train set) and subset of VOT 2014 (test set).

	EAO
SiamFC-A	0.235
Color KCF	0.226
SO-DLT	0.221
KCF 2014	0.192

Table 2. The EAO of other methods tested on VOT 2016. The SiamFC-A use Alexnet as backbone as we do. The results are copied from VOT2016 report [14].

	Speed
Template/ Search window loading	0.004s
Siamese network	0.005s
Transform to absolute coordinates	0.0004s

Table 3. The speed of each stage, recorded on NVIDIA GTX 980m, intel core i7 laptop.

Unfortunately, we cannot compare with other methods fairly, the reasons are: (1) All the other Siamese network-based approaches are trained on bigger datasets. We don't have the time and resource to train on these datasets. It involves rearranging other datasets and a powerful GPU. (2) The VOT 2016 is the dataset we can find in MTF library [12] that fits for training our model (in terms of variety of the categories and a large number of frames). Therefore, we report the testing results on the training set and the testing set which we gathered from the VOT 2014. Since the subset of VOT 2014 we choose does not appear in the training set, we consider this as a correct measurement for performance. Besides this, we searched online to find some results on the VOT 2016 (Table 2) of other methods. These results are only for reference.

From Table 1, we can see the big gap between the testing result on train set and test set. This means the model does not have a great generalization ability. By comparing with other methods in Table 2, our method is competitive to recent approaches. Then again, this comparison is not fair and our methodology is a naïve approach that has room to improve. Table 3 shows the speed of our approach. The speed is almost the same with paper [6], since the network structure is very similar. The other Siamese network based methods [2,3,4,5] are slower than this speed.

The visualization of a real testing sequence can be viewed in our demos. From these videos some problems are shown: (1) An object that is similar to the objects appeared in the training set is easily tracked (human, ball), while new objects can hardly be tracked well. (2) It is not very robust in tracking objects that have large scale changes.

Note that we actually trained our model in TMT dataset [12] first. It is not well learned and we argue the reason is that although this dataset has more frames available, it only contains few categories of object/patterns (marker, cereal box, mug, etc.) compared to VOT dataset. Therefore, the branch in the Siamese network for the template image always receives a similar input, which raises a concern on training the Siamese network. Also, it is proved in paper [4], to a Siamese network, the variety of the categories of objects/patterns is the key to provide more robust result. The TMT dataset does not have this characteristic.

5. Conclusion

In this research, we explore the possibility of using the Siamese to generate orientated bound boxes that better describe the object. We develop a learning strategy that indirectly learns the warp parameters and achieved a competitive result against advanced algorithms (Though the comparison is not exactly fair).

Several problems we observed:
(1) Dataset with variety of objects and long sequence of labelled frames are needed.
(2) The generalization ability is not great. Besides the small dataset issue, we argue that this might because we choose to follow the pipeline (the concatenation inside the network) in papers [1,6] which actually impose less influence from the template.

More can be done in the future:
(1) Enhance the template influence by introducing negative pairs [4]. Our current framework is trained based on the assumption that an object is always included in the next search window (all pairs are matched and positive when training). Maybe a classification score map can be added to support the negative pairs.
(2) Find a way to integrate this methodology with SiamFC. This is a suggestion that we can consider the sliding window approach and the use of cross-correlation rather than simple concatenation on the output features from the Siamese Network.

Acknowledge:

Work distribution can be divided into:
(1) Yuchen Yang**:** Pipelines design, Model and loss strategies implementation, train/test/eval interfaces implementation. Main report writing.
Section 3.2 3.3

(2) Dong Huo**:** Training set and label generation, Overview of wrapping pipeline implementation, Evaluation criterion implementation.
Section 3.1

Other sections are collaborated work.

Codes used for reference:
https://github.com/chenhsuanlin/inverse-compositional-STN

https://github.com/harveyslash/Facial-Similarity-with-Siamese-Networks-in-Pytorch

Reference

[1] Lu, Xiankai, et al. "Learning Transform-Aware Attentive Network for Object Tracking." Neurocomputing (2019).

[2] Bertinetto, Luca, et al. "Fully-convolutional siamese networks for object tracking." European conference on computer vision. Springer, Cham, 2016.

[3] Li, Bo, et al. "High performance visual tracking with siamese region proposal network." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[4] Zhu, Zheng, et al. "Distractor-aware siamese networks for visual object tracking." Proceedings of the European Conference on Computer Vision (ECCV). 2018.

[5] Wang, Qiang, et al. "Fast Online Object Tracking and Segmentation: A Unifying Approach." arXiv preprint arXiv:1812.05050 (2018).

[6] Held, David, Sebastian Thrun, and Silvio Savarese. "Learning to track at 100 fps with deep regression networks." European Conference on Computer Vision. Springer, Cham, 2016.

[7] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.

[8] Jaderberg, Max, Karen Simonyan, and Andrew Zisserman. "Spatial transformer networks." Advances in neural information processing systems. 2015.

[9] DeTone, Daniel, Tomasz Malisiewicz, and Andrew Rabinovich. "Deep image homography estimation." arXiv preprint arXiv:1606.03798 (2016).

[10] Erlik Nowruzi, Farzan, Robert Laganiere, and Nathalie Japkowicz. "Homography estimation from image pairs with hierarchical convolutional networks." Proceedings of the IEEE International Conference on Computer Vision. 2017.

[11] Jurie, Frédéric, and Michel Dhome. "Hyperplane approximation for template matching." IEEE Transactions on Pattern Analysis and Machine Intelligence 24.7 (2002): 996-1000.

[12] http://webdocs.cs.ualberta.ca/~vis/mtf/

[13] https://github.com/stwklu/DL_SearchMethod/blob/master/428_report.pdf

[14] The Visual Object Tracking VOT2016 challenge results