applied sciences

MDPI

Article

Siamese Tracking with Adaptive Template-Updating Strategy

Zheng Xu [1,2,3,4,5,*], Haibo Luo [1,2,4,5], Bin Hui [1,2,4,5], Zheng Chang [1,2,4,5] and Moran Ju [1,2,3,4,5]

1 Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China
2 Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110016, China
3 University of Chinese Academy of Sciences, Beijing 100049, China
4 Key Laboratory of Opto-Electronic Information Processing, Chinese Academy of Science, Shenyang 110016, China
5 The Key Lab of Image Understanding and Computer Vision, Shenyang 110016, China
* Correspondence: xuzheng@sia.cn; Tel.: +86-024-2397-0757

Received: 9 August 2019; Accepted: 2 September 2019; Published: 6 September 2019

Abstract: Recently, we combined a contour-detection network and a fully convolutional Siamese tracking network to initialize a new start-up of vehicle tracking by clicking on the target, generating a contour proposal template instead of using a fixed bounding box. Tests on the OTB100 and Defense Advanced Research Projects Agency (DARPA) datasets proved that our method outperformed the state of the art and effectively solved the partial-occlusion problem. However, the current Siamese tracking method uses the target in the first frame as a template during the whole tracking period, and leads to the failed tracking of target deformation. In this paper, we propose a new template-update method and reconstruct the whole tracking process with a template-updating module. To be specific, the innovative adaptive template-updating module is comprised of a neural contour-detection network and a target-detection network. Experiment results on the DARPA dataset prove that our new tracking algorithm with the template-updating strategy prominently improved tracking accuracy regarding the deformation condition.

Keywords: object tracking; Siamese network; template-updating; deep learning

1. Introduction

Visual-object tracking is one of the most fundamental problems in computer vision and has attracted much attention in recent years. A wide range of applications relies on tracking methods such as human-motion analyses [1], robotic services [2], human–computer interactions [3], and surveillance [4]. In this paper, we focus on single-object tracking with the target given in the first frame, which is the most common scenario of tracking.

There are two aspects to improve target-locating precision and tracking robustness in existing tracking methods. One is to design effective algorithms of either generative [5–8] or discriminative [9–15] models; the other is to optimize target representation by extracting both conventional handcrafted features and deep convolutional features. As a result, complicated tracking algorithms or features improve performance but simultaneously increase computational complexity. However, the most difficult and significant problem in tracking application is to balance performance and speed. Unlike state-of-the-art DCF methods such as LADCF [16] and MFT, which fuse multiresolution and multifeature to improve performance on rank and sacrifice algorithm speed, Siamese tracking-network [17–23] approaches obtain balanced accuracy and speed. To be specific, GOTURN [18], SiamFC [19], and RASNet [20] transfer traditional tracking tasks from a target-searching and regressing process into a similarity-comparing function and complete tracking in each frame within a fixed amount of time. DSiam [7] adopts online target-variation transformation and

background-suppression transformation, which make DSiam adapt target changes while excluding background interference.

There are three problems that should be addressed, even though these Siamese tracking approaches have balanced speed and accuracy: first of all, most of these Siamese network trackers are not able to update the templates. On the one hand, their simple fixed models [17–20,23] guarantee high tracking speed; on the other hand, they sacrifice better adapting to target-appearance deformation. Second, the performance of most Siamese tracking networks cannot be guaranteed, especially with cluttered backgrounds, because of the interference of semantic backgrounds. In other words, conventional bounding-box inputs are not able to separate background distractions. Last, out-of-view and full-occlusion conditions are still great challenges due to the local-search strategy that is employed by most Siamese trackers.

Our contributions are threefold: (1) We designed and implemented the whole Siamese tracking system with an adaptive template-updating strategy. Single-point initialization, on the one hand, took advantage of our contour templates over conventional bounding-box templates; on the other hand, it achieved the function of our start-up in the system. (2) We designed and combined a target-and a contour-detection network to compose our template-update network that was able to adaptively update templates according to different conditions. (3) Our method achieved leading performance regarding vehicle video images in both the OTB100 and the Defense Advanced Research Projects Agency (DARPA) dataset.

2. Related Work

In this part, we introduce our related work from the following aspects: Section 2.1, Siamese Network; Section 2.2, Template Update; and Section 2.3, Shape-Adaptive Template.

2.1. Siamese Network

Recently, Siamese networks have been widely utilized in many visual-tracking tasks. A Siamese network has two branches, and each of them encodes different features into another representation domain and are then compared with each other by fusing with a specific tensor. Finally, a single output is produced to display results such as similarity. Siamese networks have balanced accuracy and speed because they transfer traditional visual tracking from searching tasks to comparison tasks.

2.2. Template Update

In real-world environments, many recent algorithms have achieved great success in visual-object tracking [24–31]. These algorithms were able to solve the problems of several types of appearance changes in the target, for instance, pose variations, occlusion, and illumination changes. However, if these changes are very extreme, most of the methods may fail to track the target. An example of false tracking caused by vehicle turning is shown in Figure 1. Test images are from the DARPA dataset's video sequence "egtest03".

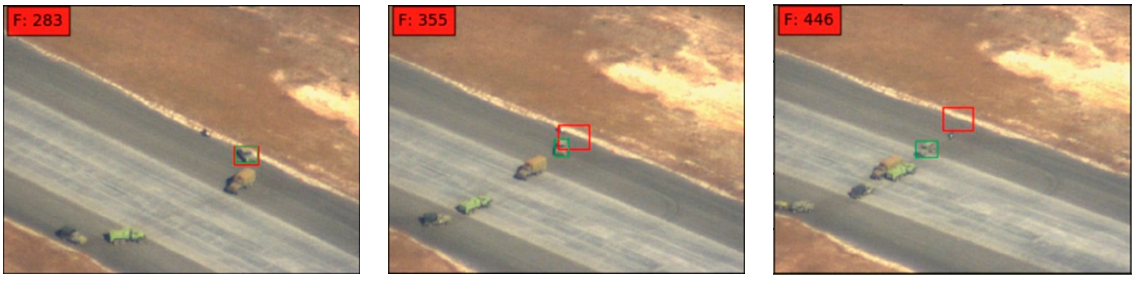


Figure 1. Example of false tracking due to target deformation: (**a**–**c**) are images to show tracking results for three frames in DARPA dataset, green boxes, ground truth; red, tracking results.

In order to track targets with severe appearance changes, some methods [5,32,33] update the target template over time to improve tracking accuracy. The most common template-updating strategies either update every single frame, or update with a fixed number of frame intervals. These template-updating tracking algorithms assume that the current updated template can describe the target better than any other templates in previous frames. However, this assumption sometimes may face the following problem. Typically, the template is updated based on tracking results according to the current frame. While trackers are not able to precisely track the target, tracking results include errors that force the template to be incorrectly updated. As a result, the updated template is not able to accurately describe the current target and eventually causes the tracker to fail.

Obviously, the aforementioned simple template-update methods are not able to satisfy our needs in tracking applications. For one case, when a tracking object is partially shielded by other disturbing objects, such as a tank shielded by a tree, traditional bounding-box template updating treats part of the tree over the tank as a tracking object and introduces interference. To avoid background clutter and template pollution, two template-updating strategies are required. First, instead of simply updating every fixed frame, we need to let the tracker know the most appropriate opportunity to update the template. This strategy can not only reduce template-pollution risks, but also reduce computational complexity of frequent useless template updating. Another beneficial strategy is to reserve backup templates; once the current template is not able to match the target, the backup ones may play a significant role. This strategy can effectively improve the robustness of tracking algorithms.

2.3. Shape-Adaptive Template

Visual-target detection and tracking are two basic and challenging problems in computer vision. They are highly correlated in most tasks, and the input provided by detection can guide tracking and enhance its performance. At the same time, the precise solution of a detection problem provides reliable observation results for tracking.

There are a variety of target-detection methods with a bounding-box output. Some recent algorithms are introduced in the work of Pont Tuset et al. and Hosang et al. [34,35]. These methods are inspired by the requirements of target detection or tracking, and have been widely used in many applications. However, compared with object-contour detection, the disadvantage of these bounding boxes is that results are inaccurate, and targets cannot be precisely located. In contrast, the results of object-contour detection clearly show the shape and precise position of the target. Using target contour instead of a bounding box to detect or track not only helps improve accuracy, it also tells the system what the target might be according to its shape.

Figure 2 shows an example of the results of the bounding-box and the object-contour recommendation methods: the results of the bounding-box recommendation method are a rough window, while the results of the object-contour recommendation method are fine pixels. Obviously, the results of the object-contour method contain most target information and least background information compared with the results of the bounding-box method. In the process of feature extraction, background information in the bounding box is tracked as target information.

(**a**) (**b**)

Figure 2. Results of (**a**) bounding-box and (**b**) object-contour recommendation methods.

In our system, cameras are on aircrafts or Unmanned Aerial Vehicles (UAVs) that are far away from the target, so the initial target is very small in most cases. In addition, most applications

currently used for real-time tracking start with the manual cropping of the bounding box on the target, and it is impossible to generate appropriate boundaries as ground truth in public datasets. In this case, we needed to make our tracker smarter to distinguish between targets and backgrounds. Therefore, we propose a method [36] to extract shape-adaptive templates from targets and track them using a fully convoluted Siamese network. Unlike most startups that use initial bounding boxes in tracking, our requirement was to select the target in real-time video by clicking on the target on the monitor screen. Once someone clicks on any part of the object, tracking information is automatically learned by the system. In particular, our system automatically detects and segments the contours of selected objects at the semantic level. Then, target features are extracted from the contour model without background information. In other words, we let the tracker know which object to track, not the bounding-box area.

3. Our Approach

Our main idea was to design a target-tracking network based on the artificial selection of objects in real-time video. Unlike most current tracking methods that run high scores in public datasets based on accurate ground-truth initialization, our network needed to achieve optimal performance in our real-time systems. In this section, we introduce our approach in three parts: system design, adaptive template-updating strategy, and Siamese tracking network.

3.1. System Design

We propose utilizing our method in an intelligent real-time tracking system that can be remotely operated and monitored. In this system, images are taken in a timely manner by cameras on aircraft in the sky, and our targets are vehicles on land. Once the system is started, users can manually select the tracking target by clicking on it on the monitor screen. Then, the system automatically learns which object to track by detecting contours and abstracting semantic information according to the position of the initial point. After extracting target features, our system begins to track targets through features in the following frames and saves target features as a high-confidence template. If the similarity between template and tracking object is smaller than a threshold due to target deformation or occlusion, which suggests that the initial template is not able to precisely match object features in the current frame, our system updates the template with strategies that are discussed further down. While the tracker fails or finishes processing all images in a video sequence, the system finishes its task as well. A brief flowchart of the whole system is shown in Figure 3.

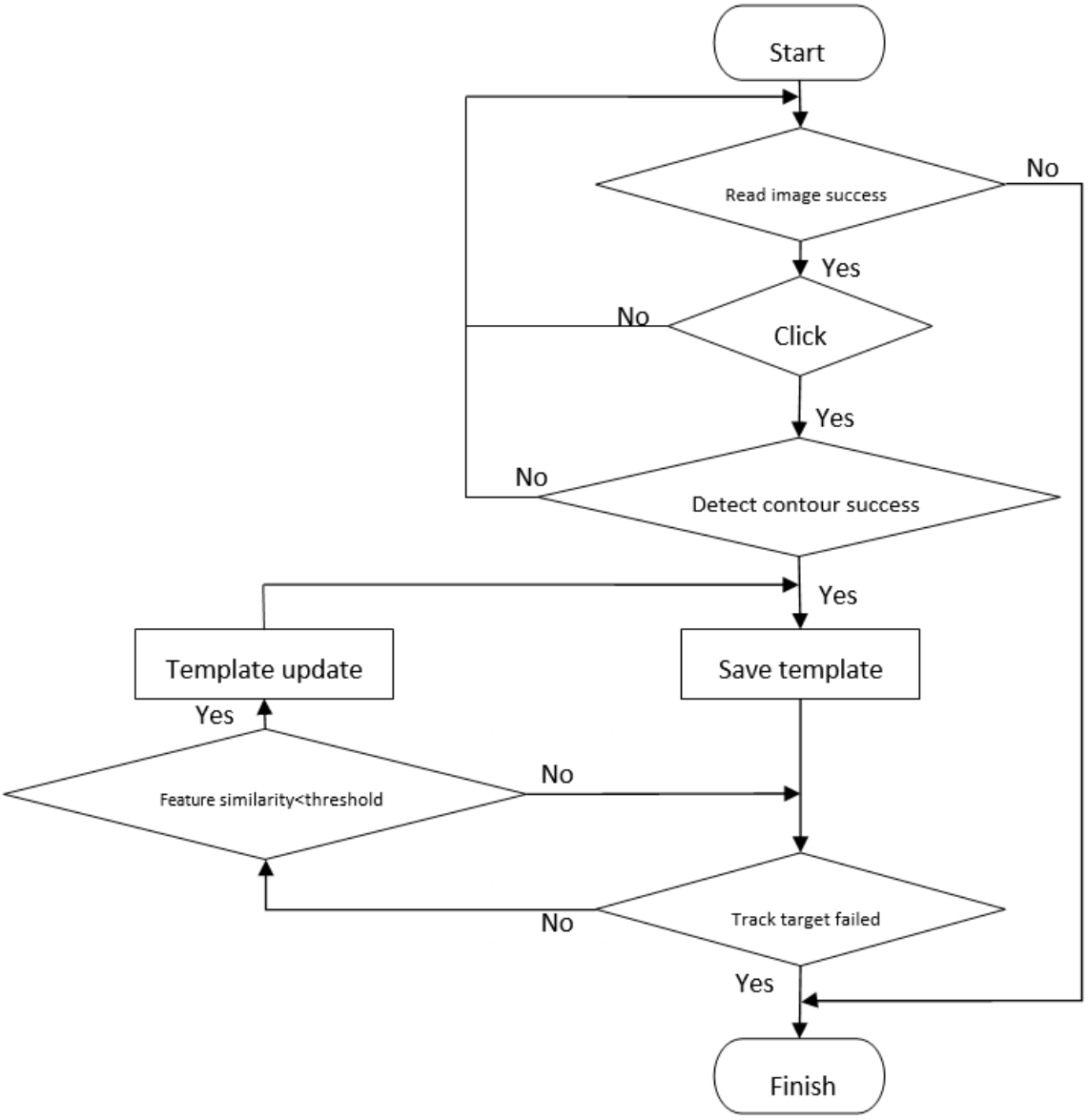


Figure 3. Flowchart of whole system.

3.2. *Adaptive Template-Updating Strategy*

Our template-update network comprises two separate networks: a contour- and a target-detection network. The main framework is shown in Figure 4.

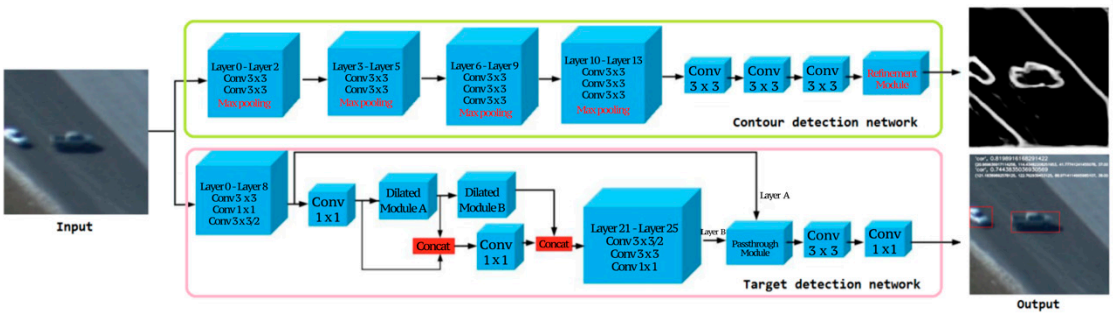


Figure 4. Framework of template-update network.

The contour-detection network is also individually used in the shape-adaptive template-generating process in the first frame. In detail, a filling algorithm started with a manually selected point is used to build the template at the output of the contour-detection network. Once the system needs to update the template in the following frame according to the result of the template-update network, the output of the contour-detection network can be reused for further processing. Specifically, the filling algorithm takes the central point of the tracking result in the current frame as the starting point to complete the contour generation of a new template.

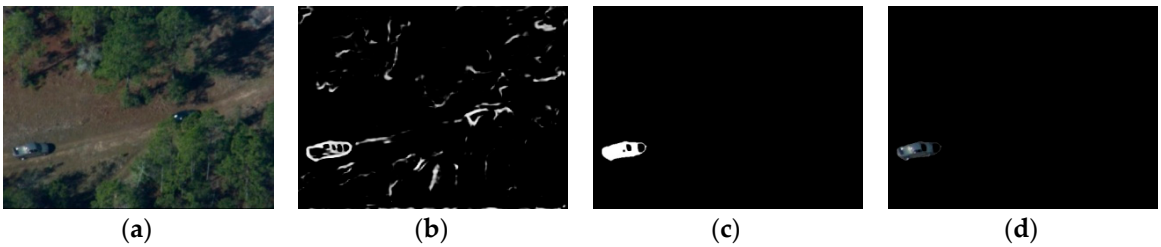
3.2.1. Contour-Detection Network

The VGG-16 network has high depth and density (16 convolution layers and stead-1 convolution kernels), and it is widely used in classification tasks because of its easy training and fast convergence. Therefore, we used the VGG-16 network structure to extract contour features. In order to detect edges, we made some modifications to our contour-detection network. First, we removed all fully connected layers. Next, we deleted the last max pool layer. Third, the output terminal was connected with a refinement module. In detail, conv1–conv5 with max pooling in the VGG-16 network were taken as our front end, which is good at extracting features; then, we modified the back end to enable our network to extract contour information. This idea was generated from the five side-output layers of a normal holistically nested edge-detection (HED) network: the lower layer captures more spatial details, but lacks sufficient semantic information; on the contrary, the deeper layer encodes more abundant semantic information but lacks spatial details. In our task, we needed to reduce the interference of useless background information and abstract target contours. Therefore, we employed the deeper-layer features to formulate our template. However, especially in the deeper layers, side outputs suffer more from the problem of thick boundaries in HED networks. Consequently, output boundaries require refinement in order to generate clear and accurate object contours. In addition, the final outputs need to be resized to their original size by up-pooling and deconvolution. To get the best performance in convolution process, we chose the smallest convolution filter (3×3) with the stride of one pixel that can capture left/right, up/down, and central motion; max pooling was performed with stride 2 over a 2×2 pixel window. The configuration of each convolutional layer and the max pooling layer in our contour-detection network is summarized in Table 1.

Table 1. Receptive field and stride size of each layer. RF, receptive field; C, convolution; P, pooling.

Layer	C1	P1	C2	P2	C3	P3	C4	P4	C5
RF size	5	6	14	16	40	44	92	100	196
Stride	1	2	2	4	4	8	8	16	16

The output of our contour-detection network was a binary image with edge information. We first used the flooding-filling method in OpenCV to generate a connected region to represent the location of the target we chose from the contour map. Next, we could further generate object masks on the basis of the previous stage. The mask set background information to zero, and the selected object was segmented from the original image. Its feature could be extracted as the target template without background information. Finally, the drift problem of the bounding box in complex or deceptive backgrounds was successfully solved. The benefits of using shape-adaptive templates instead of conventional bounding-box region templates were discussed in Section 2.2, and the shape-adaptive template-generating process is illustrated in Figure 5.



(**a**) (**b**) (**c**) (**d**)

Figure 5. Shape-adaptive template generating process: (**a**) original image; (**b**) contour map; (**c**) target mask; (**d**) target template.

3.2.2. Target-Detection Network

Target detection is another hotspot issue in computer vision. For our real-time system, we only focused on one-stage target-detection algorithms, which have a higher speed. As one of the most advanced target-detection methods, YOLOv3 [37] can detect targets of different scales with high accuracy. Tiny YOLOv3 sacrifices detection accuracy, especially for small targets through channel pruning, in order to improve detection speed. However, in our task, most targets were small. To avoid loss of contextual information, we utilized a "dilated module" inspired by dilated convolution [38] to expand the receptive fields. The comparison of the receptive fields between a convolution network and a dilated convolution network is summarized in Table 2.

Table 2. Comparison of receptive fields, kernel sizes, and strides between convolutional network and dilated convolutional network of each layer.

Convolutional Network Layer	**0**	**1**	**2**	**Dilated Convolutional Network Layer**	**0**	**1**	**2**
RF size	3	5	7	RF size	3	7	15
Kernel size	3 × 3	3 × 3	3 × 3	Kernel size	3 × 3	3 × 3	3 × 3
Stride	1	1	1	Stride	1	1	1
				Dilated rate	1	2	4

The "passthrough module" was added to combine target location information in the lower layers and semantic information in the higher layers. To be specific, the features of the lower layers contained more fine-grained information, which helps to locate small targets. Therefore, we used a passthrough layer with stride 2 that transformed the feature map from 2N × 2N × C to N × N × 4C. The function of the passthrough layer is explained in Figure 6.

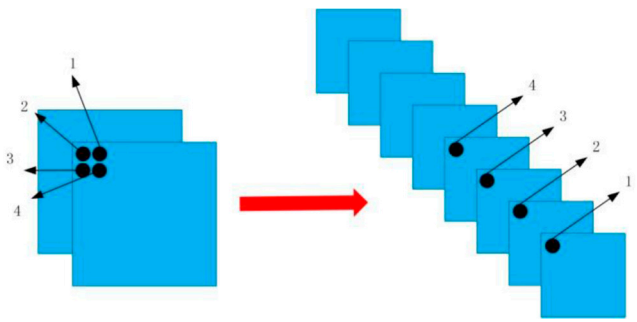


Figure 6. Passthrough-layer function: black circles with numbers indicate pixel in feature map before and after going through passthrough layer.

Finally, we obtained similar accuracy with YOLOv3, and similar speed with tiny YOLOv3. The details and experiment results are available in previous work [39] from our team.

3.2.3. Template-Update Strategy

We set the greatest response on the heat map as the criterion for strategic decision. When the greatest response was smaller than our threshold, it indicated that the similarity between the tracked target in the current frame and the template was low. There were mainly two kinds of conditions in our application scenarios that could lead to this phenomenon: deformation and occlusion. When the target is deformed, such as when a vehicle makes a turn, updating the template is necessary because previous templates may not be suitable for the same object from another view angle; on the contrary, when a target is temporarily occluded, such as when a vehicle passes through the jungle and is shielded by trees, updating the template during occlusion is not needed and may cause template pollution because of shield interference. Therefore, our system needs to differentiate between these two conditions and decides whether to update the template or not. Examples of deformation and occlusion in tracking are shown in Figure 7.

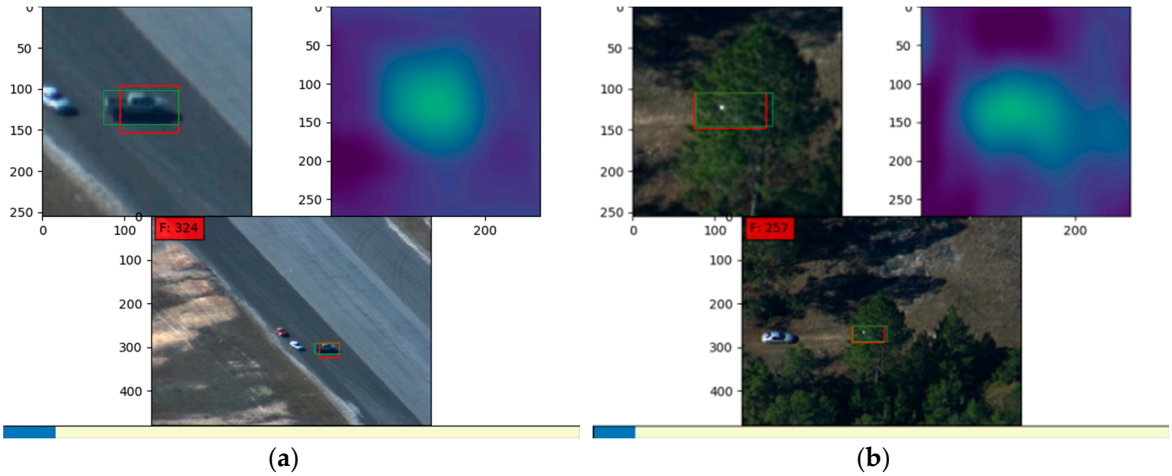


Figure 7. Results during (**a**) deformation and (**b**) occlusion. Bottom, tracking result in origin frame; top left, cropped image showing partial enlarged detail of tracking result; top right, corresponding heat map denoting similarity. Green boxes, ground truth; red, tracking results.

After analyzing the characteristics of both conditions, we proposed an updating strategy that decided by the results of two separate networks: a contour- and a target-detection network. If the target had obvious appearance changes from one frame and activated the template-update network, both networks would output corresponding results within a neighboring target area, and a new appearance model would be generated. It needs to be denoted that our tracking method with shape-adaptive templates is able to deal with partial-occlusion conditions because there is less background interference compared with the traditional bounding-box template. Only severe occlusion may cause similarity low enough to activate the template-update network. Under severe occlusion conditions, the contour- and target-detection networks are not able to get consistent results in most cases. Figure 8 shows outputs from our template-updating network regarding the deformation and occlusion conditions.

(**a**) (**b**)

Figure 8. Outputs from template-update network under (**a**) deformation and (**b**) occlusion: left of each group, output from contour-detection network; right output from target-detection network.

In Figure 8a, which was under deformation, the target vehicle in the center of image and its contour were simultaneously detected, while in Figure 8b, which was under occlusion, neither the more severe shielded vehicle nor its contour were detectable. In order to improve the robustness of our system and avoid unknown conditions that may lead to false tracking, we preserved the template in the first frame, which was manually selected and had the highest confidence. This highest confidence template was utilized as a backup template to match with the search region during each template update except for the first time.

3.3. Siamese Tracking Network

For most Siamese tracking networks, such as GOTURN and SiamFC, the first step is to zoom in on and crop input target images in the absence of target-feature information. In this process, when a target moves out of the bounding box, it is beneficial to add a margin to the context, but at the same time, this may lead to target-size reduction and the introduction of more background information. Siam FC is better than GOTURN for three reasons: first, its fully convoluted network structure makes offline training data highly identifiable; second, SiamFC uses a strictly supervised heat map rather than a recommended regression. Finally, the accuracy of the fusion tensor is improved by using a correlation layer as the fusion tensor, to which the success of correlation-based methods on visual-target tracking is owed. GOTURN uses regression results in the previous frame and updates the template in each frame, while SiamFC generates the template from basic facts in the first frame that in our system can be manually selected by operators. Feature similarity between template and search area is calculated by correlation operation, and a similarity heat map is generated. The highest response in the heat map represents the location of the target center in the next frame. In summary, the system realizes the tracking function from frame to frame.

Figure 9 shows the architecture of our Siamese tracking network: First, the full convolutional network takes template and search area as inputs, respectively, and outputs features for the template and search area. Then, the heat map is generated through correlation of the similarity between template and search-area features. Last but not least, the target is tracked in the search area according to the highest response on the heat map.

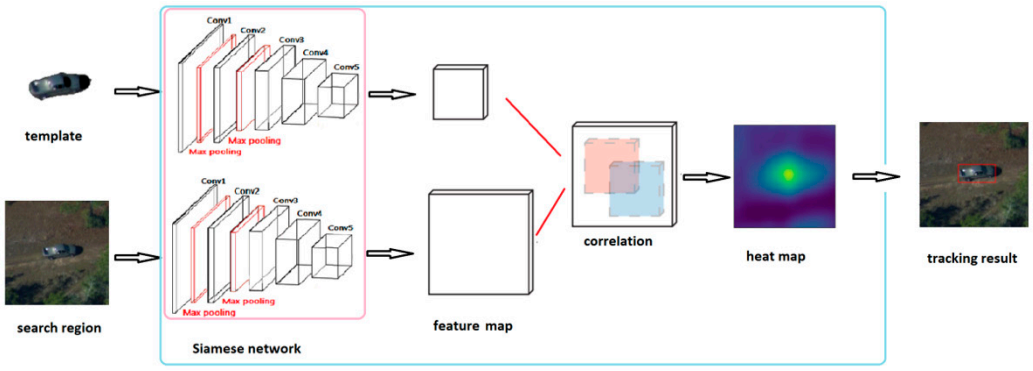


Figure 9. Architecture of our Siamese tracking network.

Configurations on kernel sizes and strides of convolutional layers in the Siamese network are summarized in Table 3. It needs to be noted that there are response-normalizing and intervening pooling layers connected to the outputs of the first two convolutional layers.

Table 3. Kernel sizes and strides in our Siamese network.

Layer	C1	C2	C3	C4	C5
Kernel size	$11 \times 11 \times 3$	$5 \times 5 \times 48$	$3 \times 3 \times 256$	$3 \times 3 \times 192$	$3 \times 3 \times 192$
Stride	4	8	16	16	16

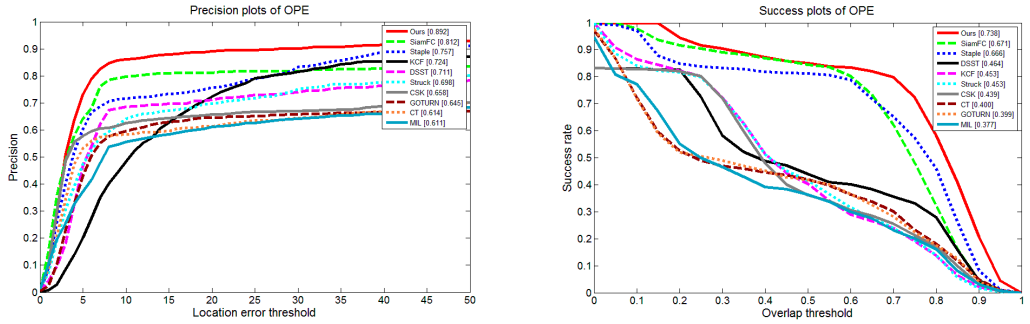
4. Experiments and Results

Our template-updating network and Siamese tracking network were implemented based on the TensorFlow framework for Python with OpenCV 2.7. We had already run our Siamese tracking network on public dataset OTB100 and compared it with nine other tracking methods. In this part. we selected eleven vehicle videos in the OTB100 dataset: Car1, Car2, Car4, Car24, BlurCar1, BlurCar2, BlurCar3, BlurCar4, Suv, CarScale, and CarDark. Following the instruction of the OTB100 benchmark, the precision plots and success plots of one-pass evaluation (OPE) were plotted. Then, our Siamese tracking network with an adaptive template-updating strategy was tested on images taken at Eglin Air Base during the DARPA Video Verification of Identity (VIVID) program. This experiment set was performed by comparing the performance of our Siamese tracking network with or a without template-updating strategy. The image resolution in this experiment set was 640 x 480. From five video sequences (egtest01, egtest02, egtest03, egtest04, and egtest05), we chose 5000 images constituting our datasets for testing. All the above experiments were tested on a server with a TITAN X GPU and 3.5 GHz CPU.

4.1. OTB100 Results

We created a group label for these vehicle videos and modified the initialization part of our Siamese tracking network for tests because our requirement for the task was different from that in the OTB100 challenge. In detail, we used the center point of each bounding box according to the ground truth instead of our initialization point.

According to the requirement of our system, we only tested the OPE metric. Figure 10 shows the precision plots and the success plots of OPE.



(**a**) (**b**)

Figure 10. Results of different trackers on vehicle videos in OTB100: (**a**) one-pass evaluation (OPE) precision plots; (**b**) OPE success plots.

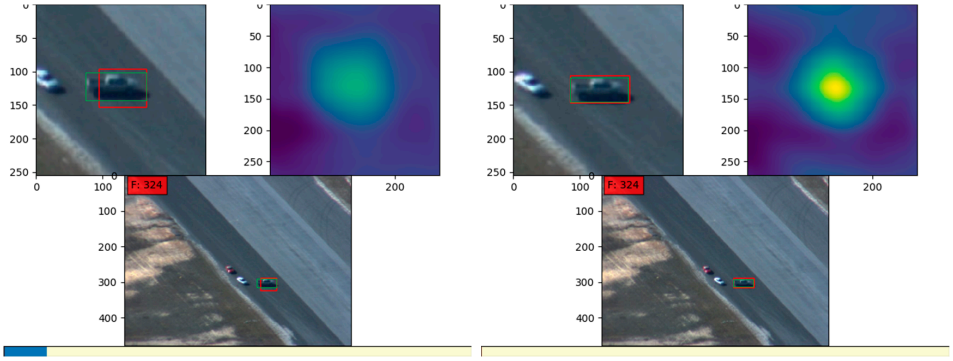
As can be seen from Figure 10, our shape-adaptive template method achieved the best performance between the 10 trackers. The distance accuracy and overlap accuracy in OPE are shown in Figure 10a,b, respectively. It is noteworthy that we compared the central accuracy scores of the success rate under the 20-pixel threshold because, in our system center, accuracy is the most important, and most target sizes in our scenarios were about 20 to 40 pixels. In the experiment, our method achieved a success rate of 0.892 on distance accuracy of the 20-pixel threshold, and 0.738 on overlap accuracy. Our method achieved the best performance for both methods: in terms of distance accuracy, our method was 9.9% better than SiamFC, which ranked second, and in terms of overlap accuracy, our method was better by 10.0% as well.

4.2. DARPA Dataset Results

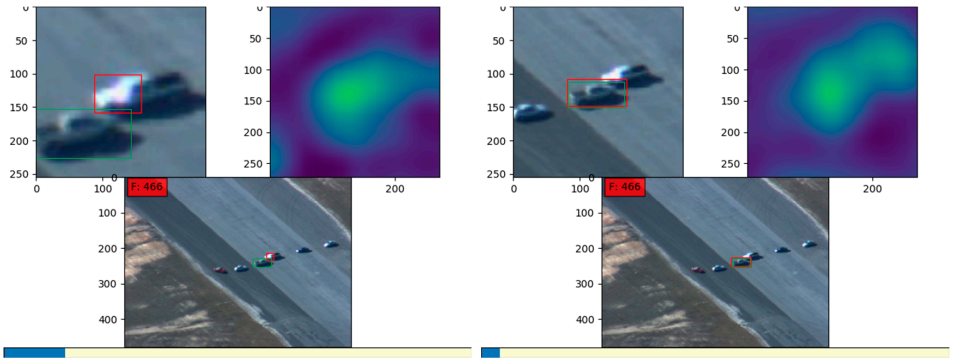
4.2.1. Qualitative Evaluation

As mentioned earlier, the images in our program were taken from airborne-sensor platforms that were far away from targets. This led to a low resolution and small-size targets. Therefore, similar datasets collected by DARPA were utilized for our testing. Images on the left-hand side below are results for the Siamese tracking network without template updating on the DARPA datasets; ones on the right-hand side are results for our Siamese tracking with an adaptive template-updating strategy on the same frames.

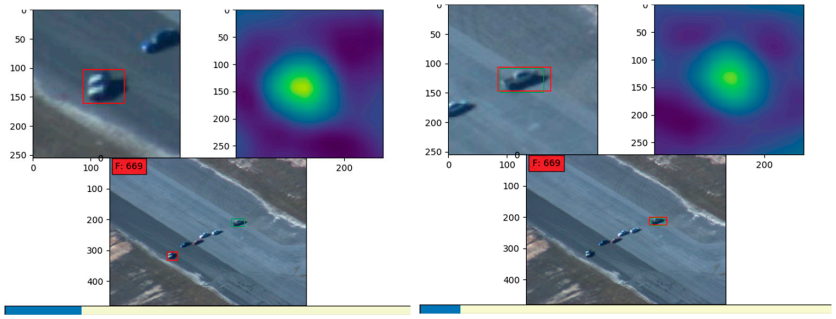
Figure 11 are the results of two different sequences of video images from the DARPA dataset. To be specific, images in Figure 11a–c are from video sequence “egtest02”, and images in Figure 11d–f are from video sequence “egtest03”. In Figure 11a,d, both of the tracking vehicles were making a turn and this lead to target deformation. These appearance changes could not match previous templates with high similarities according to the heat maps. As a result, the left images in Figure 11b,c,e,f show that the tracker without template-updating failed to track the target, while the right images in the corresponding group prove that our latest method was able to solve the problem through template updating.



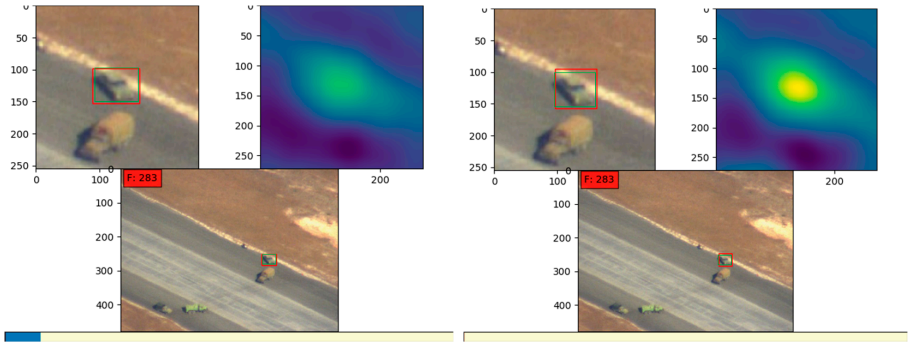
(**a**)



(**b**)

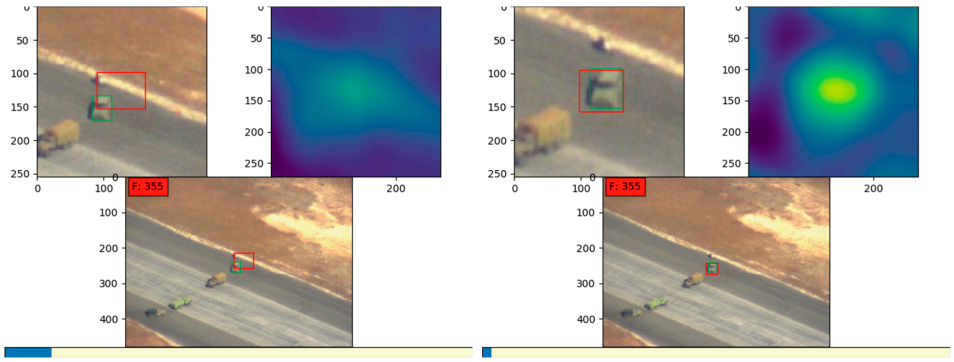


(**c**)

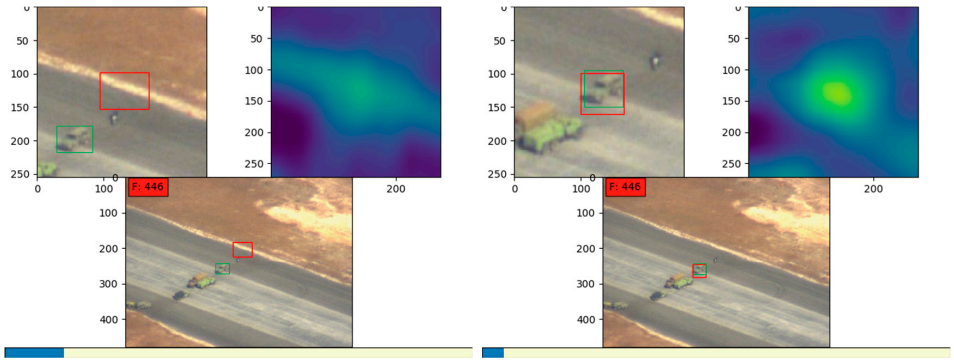


(**d**)

Figure 11. *Cont.*



(**e**)

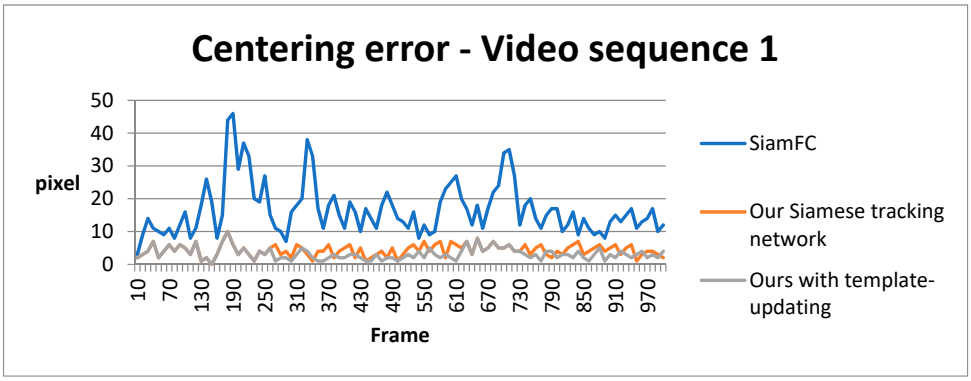


(**f**)

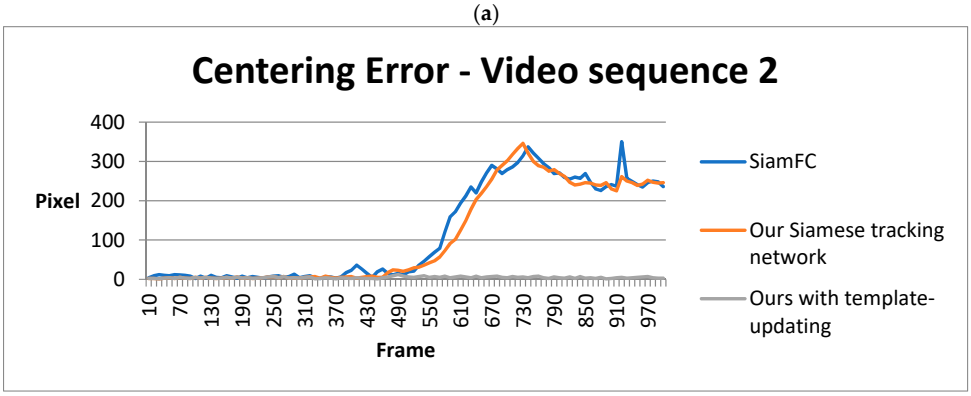
Figure 11. Results on images in datasets taken by the Defense Advanced Research Projects Agency (DARPA): (**a**–**f**) are six groups of comparison results on the DARPA dataset between the Siamese tracking methods without or with adaptive template-updating strategy. Bottom image in each group, tracking result in origin frame; top left, cropped image showing partial enlarged detail of tracking result; top right, corresponding heat map denoting similarity. Green boxes, ground truth; red, tracking results.

4.2.2. Quantitative Evaluation

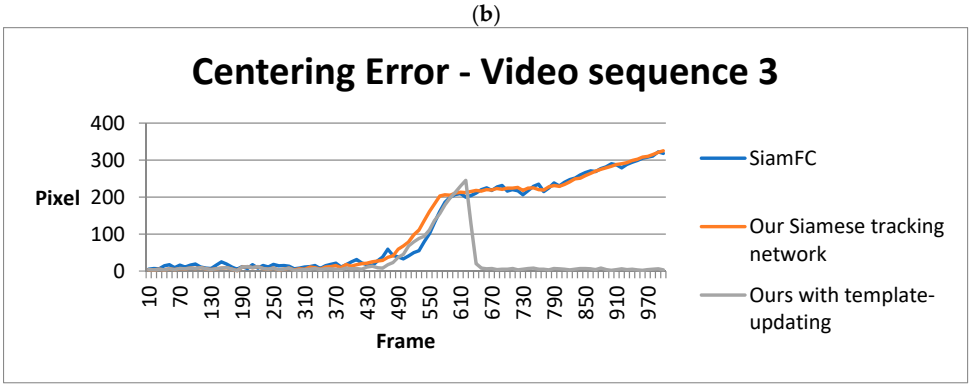
We calibrated and recorded the result deviations of SiamFC, our Siamese tracking network, and our Siamese tracking network with a template-updating strategy, respectively. In this set of experiments, we only compared the centering error of the different methods in each video sequence because, in our application, center precision was the most significant and could have had serious consequences. We calculated the centering error between the tracking results and the calibrated ground-truth centers every 10 frames for the three different methods, outlined in Figure 12. It is worth noting that the target size in the first four video sequences was about 20 to 40 pixels, and in the video sequence, 5 was about 100 pixels. In most frames, the central error of our method was less than 10 pixels. This deviation could be tolerated as long as the detected target center was still in the actual target location. When faced with challenges such as occlusion or deformation, our Siamese tracking with adaptive template-updating strategy had the least mistakes during the tracking process. The 'success of updating templates in our latest method generated some spikes that can be seen in the results of video sequences 3 and 5. Overall, the average centering error of our latest method was much lower than that of the others.



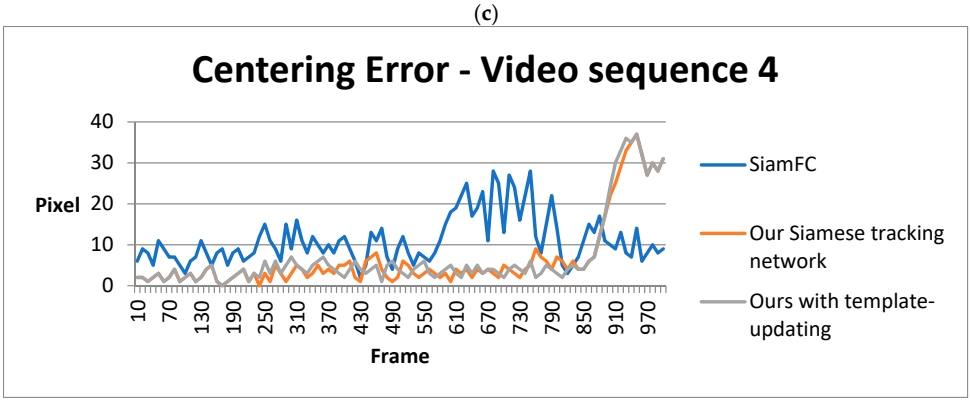
(**a**)



(**b**)

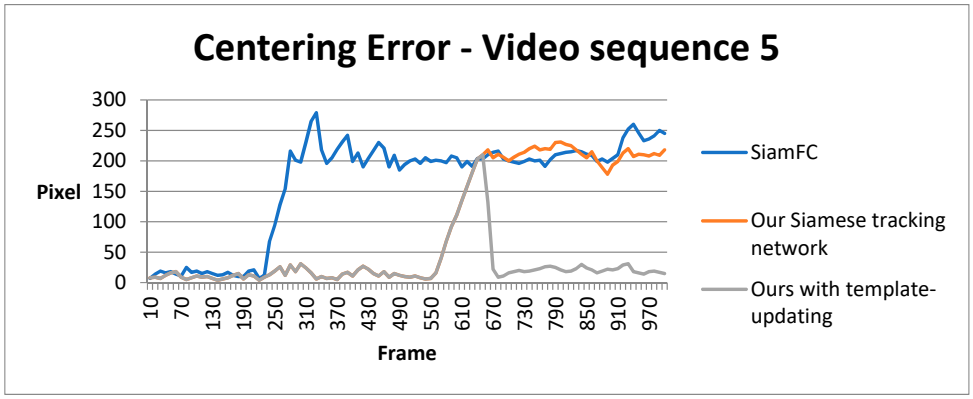


(**c**)

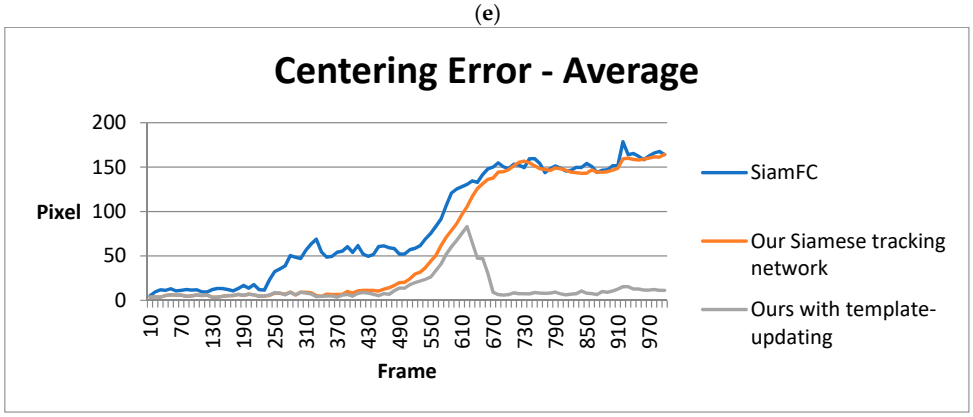


(**d**)

Figure 12. *Cont.*



(**e**)



(**f**)

Figure 12. Statistical results of SiamFC, our Siamese tracking network, and our Siamese tracking network with template-updating strategy: (**a**–**e**) centering error of three methods in five different video sequences and (**f**) average centering error.

5. Conclusions

In this paper, we first analyzed the advantages and drawbacks of recent tracking methods and argued that Siamese tracking approaches, balancing accuracy and speed, are the most suitable for our requirements. After that, we designed and implemented the whole system with an innovative template-update network by modifying and combining a contour-detection network and a target-detection network to track in real time using contour templates instead of a conventional bounding-box template. Experiment results showed that our latest Siamese tracking with adaptive template-updating strategy was able to improve performance regarding occlusion and deformation problems in object tracking. The whole system is suitable for single-rigid-object tracking in real time. For multiple objects or nonrigid-object real-time tracking, there is still further work to be done.

Author Contributions: Z.X. and H.L. conceived and designed the experiments; Z.X. performed the experiments; B.H. and Z.C. analyzed the data; M.J. designed the target-detection network; and H.L. supervised this work.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bhattacharya, S.; Souly, N.; Shah, M. Covariance of Motion and Appearance Features for Spatio Temporal Recognition Tasks. *arXiv* **2016**, arXiv:1606.05355.
2. Di Rocco, M.; Sathyakeerthy, S.; Grosinger, J.; Pecora, F.; Saffiotti, A.; Bonaccorsi, M.; Cavallo, F.; Limosani, R.; Manzi, A.; Teti, G.; et al. A planner for ambient assisted living: From high level reasoning to low-level robot execution and back. In *Qualitative Representations for Robots, ser. AAAI Technical Report*; AAAI Spring Symposium: Menlo Park, CA, USA, 2014.

3. Broz, F.; Di Nuovo, A.; Belpaeme, T.; Cangelosi, A. Multimodal robot feedback for eldercare. In Proceedings of the Workshop on Robot Feedback in Human-Robot Interaction: How to make a Robot Readable for a Human Interaction Partner at IEEE International Symposium on Robot and Human Interactive Communication, Boston, MA, USA, 5–8 March 2012.
4. Tang, S.; Andriluka, M.; Andres, B.; Schiele, B. Multiple people tracking by lifted multicut and person reidentification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 3701–3710.
5. Ross, D.A.; Lim, J.; Lin, R.-S.; Yang, M.-H. Incremental learning for robust visual tracking. *Int. J. Comput. Vis.* **2008**, *77*, 125–141. [CrossRef]
6. Kwon, J.; Lee, K.M. Visual tracking decomposition. In Proceedings of the Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, San Francisco, CA, USA, 13–18 June 2010; pp. 1269–1276.
7. Jia, X.; Lu, H.; Yang, M.-H. Visual tracking via adaptive structural local sparse appearance model. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 16–21June 2012; pp. 1822–1829.
8. Zhong, W.; Lu, H.; Yang, M.-H. Robust object tracking via sparsity-based collaborative model. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 16–21 June 2012; pp. 1838–1845.
9. Kalal, Z.; Mikolajczyk, K.; Matas, J. Tracking-learning detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 1409–1422. [CrossRef]
10. Danelljan, M.; Khan, F.S.; Felsberg, M.; Van De Weijer, J. Adaptive color attributes for real-time visual tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 24–27 June 2014; pp. 1090–1097.
11. Hare, S.; Saffari, A.; Torr, P.H.S. Struck: Structured output tracking with kernels. In Proceedings of the 2011 International Conference on Computer Vision, Washington, DC, USA, 20–25 June 2011; pp. 263–270.
12. Hong, S.; You, T.; Kwak, S.; Han, B. Online tracking by learning discriminative saliency map with convolutional neural network. In Proceedings of the 32nd International Conference on Machine Learning (ICML-15), Lille, France, 6–11 July 2015; pp. 597–606.
13. Ning, J.; Yang, J.; Jiang, S.; Zhang, L.; Yang, M.-H. Object tracking via dual linear structured svm and explicit feature map. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 4266–4274.
14. Babenko, B.; Yang, M.-H.; Belongie, S. Robust object tracking with online multiple instance learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *33*, 1619–1632. [CrossRef]
15. Zhang, J.; Ma, S.; Sclaroff, S. Meem: Robust tracking via multiple experts using entropy minimization. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 188–203.
16. Xu, T.; Feng, Z.H.; Wu, X.J.; Kittler, J. Learning adaptive discriminative correlation_filters via temporal consistency preserving spatial feature selection for robust visual tracking. *arXiv* **2019**, arXiv:1807.11348.
17. Tao, R.; Gavves, E.; Smeulders, A.W.M. Siamese instance search for tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 1420–1429.
18. Held, D.; Thrun, S.; Savarese, S. Learning to track at 100 fps with deep regression networks. In Proceedings of the European Conference on Computer Vision, Las Vegas, NV, USA, 27–30 June 2016; pp. 749–765.
19. Bertinetto, L.; Valmadre, J.; Henriques, J.F.; Vedaldi, A.; Torr, P.H.S. Fully-convolutional siamese networks for object tracking. In Proceedings of the European Conference on Computer Vision Workshop, Amsterdam, The Netherlands, 8–16 October 2016; pp. 850–865.
20. Wang, Q.; Teng, Z.; Xing, J.; Gao, J.; Hu, W.; Maybank, S. Learning attentions: Residual attentional siamese network for high performance online visual tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 Jun 2018.
21. Valmadre, J.; Bertinetto, L.; Henriques, J.F.; Vedaldi, A.; Torr, P.H.S. End-to-end representation learning for correlation filter based tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
22. Guo, Q.; Feng, W.; Zhou, C.; Huang, R.; Wan, L.; Wang, S. Learning dynamic siamese network for visual object tracking. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017.

23. Li, B.; Yan, J.; Wu, W.; Zhu, Z.; Hu, X. High performance visual tracking with siamese region proposal network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 Jun 2018; pp. 8971–8980.
24. Godec, M.; Roth, P.M.; Bischof, H. Hough-based tracking of non-rigid objects. In Proceedings of the IEEE International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011.
25. Han, B.; Davis, L. On-line density-based appearance modeling for object tracking. In Proceedings of the IEEE International Conference on Computer Vision, Beijing, China, 17–20 October 2005.
26. Hong, S.; Han, B. Visual tracking by sampling tree-structured graphical models. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014.
27. Sevilla-Lara, L.; Learned-Miller, E. Distribution fields for tracking. In Proceedings of the Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 16–21 June 2012.
28. Danelljan, M.; Bhat, G.; Khan, F.S.; Felsberg, M. ECO: Efficient Convolution Operators for Tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
29. Timofte, R.; Kwon, J.; Van Gool, L. PICASO: Pixel correspondences and soft match selection for real-time tracking. *CVIU* **2016**, *153*. [CrossRef]
30. Kwon, J.; Lee, K.M. Adaptive visual tracking with minimum uncertainty gap estimation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 18–32. [CrossRef]
31. Kwon, J.; Timofte, R.; Van Gool, L. Leveraging observation uncertainty for robust visual tracking. *CVIU* **2017**, *158*, 62–71. [CrossRef]
32. Collins, R.T.; Liu, Y.; Leordeanu, M. Online selection of discriminative tracking features. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 1631–1643. [CrossRef]
33. Jepson, A.D.; Fleet, D.J.; Maraghi, T.F.E. Robust online appearance models for visual tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* **2003**, *25*, 1296–1311. [CrossRef]
34. Pont-Tuset, J.; Gool, L.J.V. Boosting object proposals: From Pascal to COCO. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015.
35. Hosang, J.; Benenson, R.; Doll'ar, P.; Schiele, B. Whatmakes for effective detection proposals? *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *38*, 814–830. [CrossRef]
36. Xu, Z.; Luo, H.B.; Hui, B.; Chang, Z. Siamese tracking from single point initialization. *Sensors* **2019**, *19*, 514. [CrossRef]
37. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
38. Leng, J.; Liu, Y. An enhanced SSD with feature fusion and visual reasoning for object detection. *Neural Comput. Appl.* **2018**, 1–10. [CrossRef]
39. Ju, M.R.; Luo, J.N.; Zhang, P.; He, M.; Luo, H. A simple and efficient network for small target detection. *IEEE Access* **2019**, *7*, 85771–85781. [CrossRef]