Assignment 2: Single Object Tracking

Abdul Bhutta
Electrical and Computer Engineering
Toronto Metropolitan University (TMU)
Toronto, Canada
abdul.bhutta@torontomu.ca

I. Introduction

Object detection and tracking are not just crucial tasks in computer vision and image processing, but they are the backbone of these fields. Their role in identifying, classifying, and tracking objects in images or videos is pivotal, given the intricate nature of objects and the environment. These tasks have been a focal point of research for many years, and the emergence of deep learning has significantly enhanced their accuracy and efficiency. Their wide array of applications, including autonomous driving, surveillance, and medical imaging, underscores their significance in the field of computer vision. Object tracking is a subset of object detection that involves tracking and classifying objects simultaneously. Although detection and tracking are two different tasks, they are closely related and are often used together in applications. It is a complex task approached through two main techniques: Single-Object Tracking (SOT) and Multiple-Object Tracking (MOT). Single-Object tracking (SOT) involves tracking a single object in a video using a bounding box. It begins with detecting the object in the first frame and then tracks it in subsequent frames using various trackers. On the other hand, Multiple-Object tracking (MOT) involves tracking and detecting multiple objects with distinct characteristics throughout the video. However, in recent research papers, many have focused on MOT, which can be modified to be used as a single object tracker. Both approaches face numerous challenges, including occlusion, scale variation, and illumination changes, which underscore the complexity of object tracking. This report introduces NanoTrack, a top-performing single object tracker that offers high efficiency and can be run on resource-constrained devices such as mobile phones and embedded devices. NanoTrack, a powerful tool for accurate real-time object detection, is built on SiamBAM and LightTrack architecture. Its practicality and real-time capabilities make it a valuable tool for a wide range of applications, outperforming other state-of-the-art trackers.

II. NanoTrack Algorithm

NanoTrack, a lightweight and efficient high-speed single-object tracker, is a reliable choice for devices with limited resources. Its efficient design ensures smooth real-time object tracking, which is built on the solid foundations of SiamBAM and LightTrack architectures. All Siamese-based networks have two identical subnetworks that share the same weights. The input image is passed through one subnetwork in which the user has selected an object to track (exemplar). The other image is a search image passed to the other network. The purpose is to use the search image to find the object, and it compares the object in different regions of the search image while creating a similarity score for each region. A few Siamese-based networks, such as SiamFC, SiamRPN, SiamMask, and SiamBAN, have been introduced. SiamFC introduced a new correlation layer, combining feature maps to extract features while displaying excellent performance [1]. SiamRPN introduced a region proposal network to generate and integrate regional proposals with the network to address the issue of scale variation [2]. The SiamMask framework introduced a mask branch to generate segmentation masks and to perform object tracking and segmentation simultaneously, leading to more accurate and robust tracking [3]. Siamese Box Adaptive Network (SiamBAN) was designed to eliminate box designs and parameters by implementing a single fully convolutional network for classification and bounding box regression [4].

LightTrack is a lightweight tracker designed for less resource-constrained devices such as mobile phones and embedded devices. It uses a neural architecture search (NAS), an automated process rather than a hand-crafted method, to design the network architecture and has been shown to outperform them. It deploys an optimal and efficient network architecture from a search space of many layers, filters, and connections. It can track objects in real time and has shown excellent results in various benchmarks compared to other state-of-the-art trackers [5]

NanoTrack, a reliable tracker, combines SiamBAN and LightTrack to create a robust architecture. It features three main components: backbone, head, and neck. The backbone network uses transfer learning and allows the selection of two different backbones: AlexNet and MobileNet. A pre-trained model, trained on a large dataset, enables the network to extract features from the image, eliminating the need to train the network from scratch. The backbone feature maps are passed to the neck module, which generates additional feature maps. The adjusted feature maps are then passed to the head module, which consists of the Box Adaptive Network (BAN). The BAN uses these adjusted feature maps to generate two important outputs: the 'cls' output, which classifies the object, and the 'bbox' output, which provides the bounding box coordinates of the object. These outputs are the results of the object-tracking process in NanoTrack.

III. Loss Function

The NanoTrack implements three different loss functions to train the network: classification loss, regression loss, and total loss. The classification loss (cls) calculates the difference between predicted and actual classes. It is computed from the head module and passed through the SoftMax function to generate the probability distribution. The cls_loss is calculated using the output from the function and passed through the cross-entropy loss. The classification loss is calculated using the equation below,

$$cls\_loss = loss\_pos * 0.5 + loss\_neg * 0.5 \tag{1}$$

where $loss\_pos$ is the positive loss and $loss\_neg$ is the negative loss which are calculted using the the negative log-likelihood loss as shown in equation 2.

$$negative\_log\_loss = -\sum_{i=1}^{M} y_i \log(p_i) \tag{2}$$

The loc_loss is an IoU loss that is used to calculate the difference between the predicted bounding box and the ground truth bounding box, which is computed using the Intersection over Union (IOU) metric as shown in the equation 3,

$$loc\_loss = 1 - \frac{Areaof Overlap}{Areaof Union} \tag{3}$$

Both loss functions are meticulously designed to optimize the network's output, ensuring maximum accuracy and minimum error. The total loss is computed by combining the classification and localization losses. It allows the network to adapt, correctly identifying the bounding box and classifying the object. Each loss is multiplied by a weight, with the starting value for each weight set at 1.0. The starting values for each weight are crucial, as they determine the initial state of the system and $total\_loss$ is calculated using equation 4.

$$total\_loss = cls\_loss * cls\_weight + loc\_loss * loc\_weight \tag{4}$$

IV. Training, Validation, and Testing

The tracker was trained on the practical and widely used object-tracking dataset, GOT-10k. This dataset, consisting of 10,000 video segments, over 560 object classes, and 87 total motion patterns, including animal, vehicle, person, passive motion object, and object part, is a valuable resource for various applications. It provides a diverse range of objects and motion patterns, not limited to fast motion and occlusion. The dataset includes labels for frame occlusion, frame absence labels, and motion class labels. It is split into a training, validation, and test set with the properties of each dataset [6] detailed in Table 1.

TABLE I
GOT-10k Dataset Properties

Dataset	Number of Videos	Number of Objects	Total Motion Patterns
Train	9335	480	67
Validation	180	150	15
Test	420	84	31

V. Results

A. Training Dataset Results

The NanoTrack model was trained using the GOT-10k dataset, a task that presented numerous technical challenges. The results are shown in the table below. It's important to note that no validation dataset was used during the training process; a decision was made to focus solely on the training. The training process was executed on a single GPU (Nvidia GeForce RTX 2080Ti) with a batch size of 32 and a starting learning rate of 0.001. The process was expected to run for 50 epochs, but unfortunately, due to a kernel crash, it was only trained until 37 epochs. The training results are shown in Table 2 for five epochs with the classification loss, localization loss, and the total loss at the end of each epoch with a total training time of over 20 hours. The best results were achieved at epoch 33, with a classification loss of 0.0228, a localization loss of 0.1727, and a total loss of 0.1955.

TABLE II
Training Dataset Results

Epoch	Classification Loss	Localization Loss	Total Loss
10	0.2668	0.3979	0.6647
15	0.0508	0.2415	0.2923
20	0.0527	0.2357	0.2884
33	**0.0228**	**0.1727**	**0.1955**
37	0.0603	0.2409	0.3012

B. Test Dataset Results

The tracker was tested on the GOT-10k test dataset using three different metrics: Average Overlap, Success Rate, and Speed. The test results for the three different metrics are shown in Table 3, where the average overlap was 0.607, the success rate was 0.733, and the speed was 99.586, with the success rate plot shown in Figure 1. The results indicate that the tracker performed well on the test dataset and achieved high accuracy.

TABLE III
Test Dataset Results

Dataset	Average Overlap	Success Rate	Speed
Test	0.607	0.733	99.586

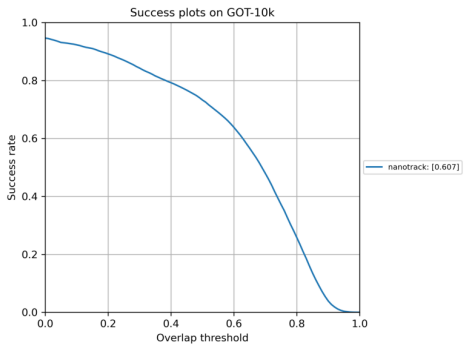

Fig. 1. Success Rate Plot

C. Inference Results

The trained tracker was applied to a demo video provided in the repository on GitHub [7], and the performance was evaluated using the newly trained weights. The user was asked to select an object to track, and both the women's faces in the video were selected, as shown in Figure 2.

Fig. 2. Object Selection

The tracker proved its reliability by consistently tracking the object in the video, even when the objects were in fast motion or obstructed by hands. This is evident in Figure 3 for object one and Figure 4 for object 2, where the tracker maintained high accuracy throughout a significant portion of the video.

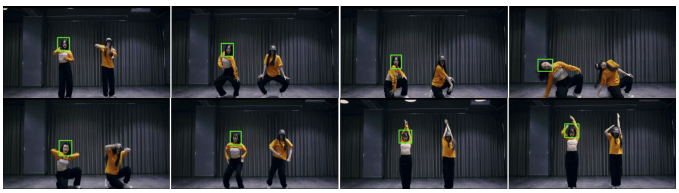

Fig. 3. Object Tracking for Women 1

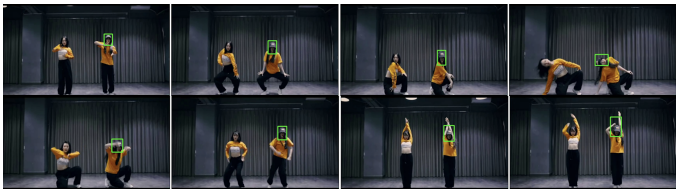

Fig. 4. Object Tracking for Women 2

VI. Occlusion

Occlusion, a common occurrence in object tracking or detection, involves an object partially or completely hidden by another. Object tracking, already a challenging task, becomes more complex with occlusion. A few issues were observed despite the generally good tracking performance throughout the video. The first instance occurred when another object occluded the tracked person, and they turned around, rendering their face partially invisible. As shown in Figure 5, the tracker lost the object and began tracking the second woman in the video. However, as soon as the person was visible again, the tracker would resume tracking. These challenges highlight the potential impact of occlusion in real-world applications.

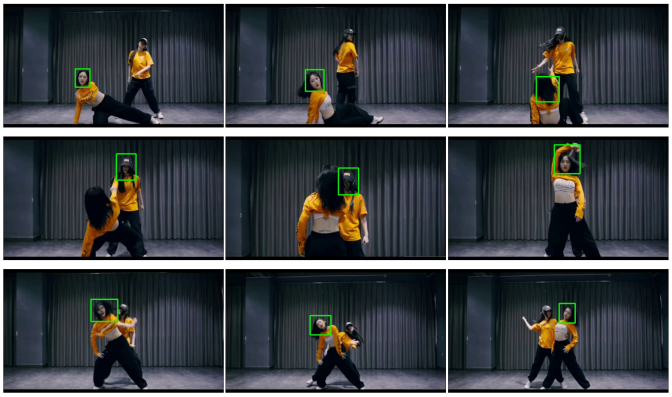

Fig. 5. Object 1 Occlusion

The second instance occurred when another object completely hid the tracked person. The first woman (object 1) entirely blocked the second woman, causing the tracker to lose focus. Even when the person became visible again, the tracker could not resume tracking for the rest of the video and instead tracked the other women, as shown in Figure 6. However, these may be the result of completing only some of the training epochs as the kernel crashed but the training process can always continue from the last epoch or checkpoint, saved after each epoch, to determine whether the tracking can be improved.

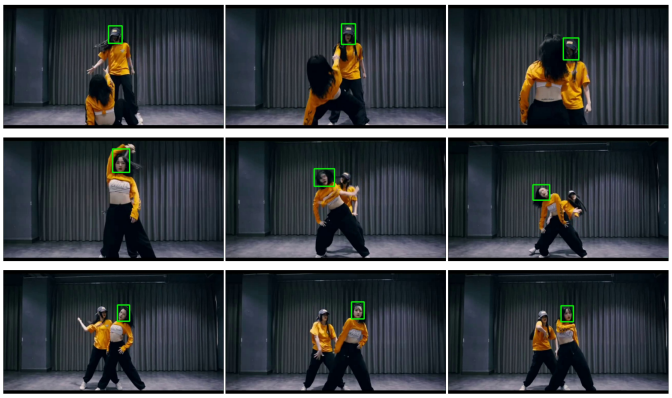

Fig. 6. Object 2 Occlusion

REFERENCES

[1] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, 'Fully-Convolutional Siamese Networks for Object Tracking', arXiv [cs.CV]. 2021 (accessed June 22, 2024)

[2] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, 'High Performance Visual Tracking with Siamese Region Proposal Network', in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 8971–8980. (accessed June 22, 2024)

[3] W. Hu, Q. Wang, L. Zhang, L. Bertinetto, and P. H. S. Torr, 'SiamMask: A Framework for Fast Online Object Tracking and Segmentation', arXiv [cs.CV]. 2022. (accessed June 22, 2024)

[4] Z. Chen, B. Zhong, G. Li, S. Zhang, and R. Ji, 'Siamese Box Adaptive Network for Visual Tracking', arXiv [cs.CV]. 2020. (accessed June 23, 2024)

[5] B. Yan, H. Peng, K. Wu, D. Wang, J. Fu, and H. Lu, 'LightTrack: Finding Lightweight Neural Networks for Object Tracking via One-Shot Architecture Search', arXiv [cs.CV]. 2021. (accessed June 23, 2024)

[6] L. Huang, X. Zhao, and K. Huang, 'GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild', IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 5, pp. 1562–1577, May 2021. (accessed June 23, 2024)

[7] Honglin Chu, "Honglinchu/siamtrackers: (2020-2022) the PyTorch version of SIAMFC, SiamRPN, DaSiamRPN, UpdateNet, siamdw, SIAMRPN++, siammask, SIAMFC++, siamcar, siamban, ocean, lighttrack, TRTR, NanoTrack; visual object tracking based on Deep Learning," GitHub, https://github.com/HonglinChu/SiamTrackers (accessed Jun. 24, 2024).