Distributed Tracking and Verifying: A Real-Time and High-Accuracy Visual Tracking Edge Computing Framework for Internet of Things

Purva Makarand Mhasakar
pmhasakar@cs.stonybrook.edu
Stony Brook University
Stony Brook, New York, USA

Kevin Doshi
kdoshi@cs.stonybrook.edu
Stony Brook University
Stony Brook, New York, USA

Ning Wang
wangn@rowan.edu
Rowan University
Glassboro, New Jersey, USA

Shen-Shyang Ho
hos@rowan.edu
Rowan University
Glassboro, New Jersey, USA

Haibin Ling
hling@cs.stonybrook.edu
Stony Brook University
Stony Brook, New York, USA

ABSTRACT

We observe that accurate and fast tracking in Internet of Things (IoT) devices is still a challenging problem. Several deep learning models have emerged which provide higher accuracy scores in object detection and tracking, however, due to their computationally expensive nature they are not useful in enabling real-time tracking at IoT devices. Correlation filters have emerged to show better speed in real-time tracking and provide good tracking results in cases of occlusion, rotation, illumination and other distractions. To get better speed as well as accuracy we use combination of correlation filter and deep learning methods. We propose a distributed tracking and verifying (DTAV) framework. Specifically, we run two object tracking algorithms, one on the client and another on the server. The algorithm run on the client is referred to as the Tracker, which is based on correlation filter and runs easily in real-time. The server hosts the verifier algorithm which performs high accuracy verification. Thus, while the client performs fast object tracking, the server's tracking algorithm verifies the output and corrects the server whenever required to maintain the accuracy of the model. We present our edge computing-based framework and discuss the motivation, system setup and series of experiments performed for the framework and present our experimental results. DTAV achieved 7.78% improvement on accuracy and 15% improvement in FPS.

CCS CONCEPTS

• **Human-centered computing** → *Ubiquitous and mobile computing systems and tools*; • **Computing methodologies** → **Tracking**; • **Computer systems organization** → *Client-server architectures.*

ACM/IEEE SEC'23, December 06–09, 2023, Wilmington, DE

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

KEYWORDS

Edge Computing, Distributed Inference, Visual object tracking for IoT

ACM Reference Format:
Purva Makarand Mhasakar, Kevin Doshi, Ning Wang, Shen-Shyang Ho, and Haibin Ling. 2023. Distributed Tracking and Verifying: A Real-Time and High-Accuracy Visual Tracking Edge Computing Framework for Internet of Things. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (ACM/IEEE SEC'23).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Visual object tracking has a major role to play in video analytics. However algorithms observe a trade off between speed, accuracy or real time tracking. This is because object tracking can be challenging due to factors like object illumination, deformation, rotation or pose variation. Object tracking is significantly used in fields like surveillance, human computer interaction, traffic monitoring and robot vision. In a given video we provide bounding box for the object to be tracked in first frame. In real time, our algorithm should be able to track this object across all frames in given video. This brings in challenges due to presence of similar looking objects and background clutter. Some deep learning methods have attempted to solve this issue and have obtained significantly higher accuracy. However, these methods face challenges in real time tracking due to computationally expensive property of deep learning models. IoT devices are often unable to host such computationally expensive algorithms, since they have limited processing capacity, limited memory and power supply. Moreover latency is also a issue. Researchers are exploring light weight algorithms which can be deployed on IoT devices. This helps the algorithms to be deployed in embedded systems. However, the obstacle occurs in providing both accuracy and speed.

To address real-time and high-accuracy dilemma in tracking, we propose a distributed tracking and verifying (DTAV) framework, inspired from the parallel tracking and verifying algorithm (PTAV) framework [**?**]. DTAV uses a fast discriminative scale space tracking (fDSST) algorithm [**?**], called *tracker*, running in the IoT device, and a more accurate Siamese network, called *verifier*, running at the backend server, to verify the outputs generated by the fDSST.

fDSST uses correlation filters since they are successful representative adaptive discriminative tracker to get real-time tracking results. Emerging edge computing is proposed to address the processing capacity of IoT devices and thus gain better speed and more accuracy.

Our objective is to provide real-time and high-accuracy visual tracking with the proposed DTAV framework in IoT device with emerging edge computing. In specific, we address the optimal configuration optimization in various edge computing scenarios. If the algorithms are not implemented in an optimised manner, edge computing will lead to sub-optimal results. Therefore, to gain speed in real time tracking, but at the same time we choose optimum parameters which provide boost to the accuracy scores. We perform experiments by varying different network parameters in order to reach the optimum network setup and accuracy. We optimize the client server settings by optimizing the communication network parameters like bandwidth, verification intervals and frames per second (FPS).

Our contributions are summarized as follows.

- We present a distributed tracking and verifying framework to visual object tracking by edge computing, which achieves high accuracy and high speed visual tracking framework for IoT at the same time.
- We present a scalable solution. Our framework can be extended to IoT devices such as mobile devices or drones. Thus, we present a solution to achieve higher accuracy without having to compromise on the speed in real time tracking.
- We validated the proposed DTAV approach, which obtained accuracy of 86.90% and showed improvement of 7.78%.

2 RELATED WORKS

In IoT applications [**?**], visual tracking has been widely employed as cloud computing is not always feasible for real-time inference due to poor latency and coverage. Generally, there is a trade-off between speed/frame rate, accuracy, robustness and generality in tracking algorithms.

In [**?**], the authors assessed the state-of-the-art in single object tracking in a video, with an emphasis on the accuracy and the robustness of tracking algorithms. In short, for visual object tracking, simple algorithms work effectively when the target is moving at slow speed, and the background noise and clutter is significantly less. However, in cases where the target moves fast or there are distractions in background. Using deep learning algorithms which have strong discriminative power might help increase the robustness of the algorithm but greatly impacts the speed and hampers real-time performance. Therefore, providing the benefits of both simple and computationally expensive algorithms became a hot area recently.

In recent years, there have been advances in improving the deep learning inference time by splitting the Deep Neural Network (DNN) between client and server or by distributing the DNN on an IoT device cluster. Papers like Neurosurgeon [**?**], Deep Things [**?**], DADS [**?**], QDMP [**?**], Auto-Split [**?**], DNN schedule [**?**], etc., have proposed different splitting/distribution techniques to perform this edge cloud partitioning like layer-based splitting, graph-based splitting, fused tile partitioning etc. The underlying motivation is the same across all techniques - reduce total inference time and possibly bandwidth consumption by implementing some form of dynamic partitioning framework over the network.

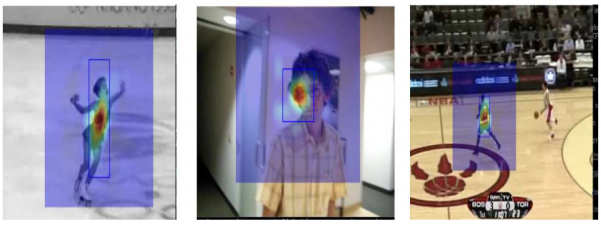

Figure 1: Object tracking performance comparison of fDSST and Siamese network results on OTB 2015 dataset, where the big bounding box is the fDSST algorithm result, and the small bounding box is the Siamese network result.

Parallel Tracking and Mapping (PTAM) algorithm [**?**] by splitting the role of tracking and its corresponding mapping algorithm into two parallel threads. The mapping does not take place for every frame. Taking inspiration from this work, PTAV framework was further developed to perform verification only on certain frames. Our work is closely related to and inspired by the PTAV framework in [**?**], with the following major difference. We aim to implement the framework in a distributed fashion - with a client-server architecture and consisting of IoT/edge devices. Because we need to communicate between client and server, another parameter of network bandwidth gets introduced which wasn't the case in PTAV as it was run on two threads on the same machine.

3 PROBLEM STATEMENT

The objective of this paper is to build a real-time and high-accuracy visual tracking system that works on IoT devices. Our proposed method consists of two tracking algorithms. A fast but low-accurate tracking algorithm called *tracker* does inference in real-time A high-accurate but slow tracking algorithm, called *verifier*, verifies and corrects the results of the tracker. As we see in Figure 1, the results obtained from the tracker may sometime need correction. This is where the verifier contributes by providing its feedback. The entire framework would run as a client-server based system, with the tracker running on the client, which is an IoT/edge device and the verifier running on a server. Tracking and verifying run asynchronously as verifying is more expensive computationally compared to tracking, so verifying is usually done at regular intervals. The main challenges in such a system is to dynamically adjust its configuration in different configurations like verification interval, speed at which the client and server perform inference, network bandwidth and observe their impact on the FPS and accuracy of our framework.

The main challenges in such a system is to dynamically adjust its configuration in different configurations like verification interval, speed at which the client and server perform inference, network bandwidth and observe their impact on the FPS and accuracy of our framework.

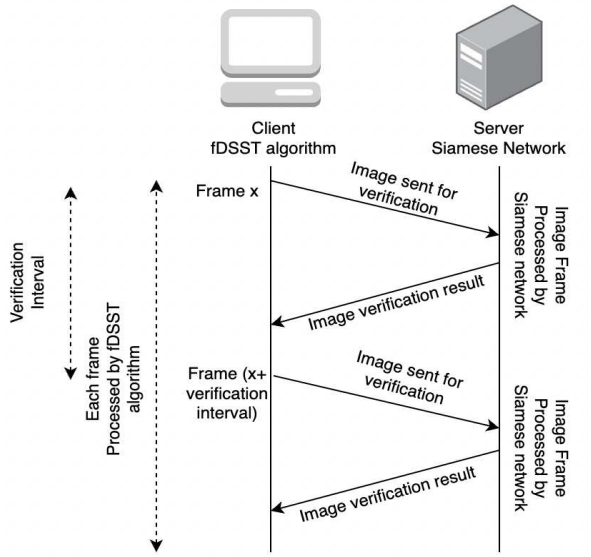

Figure 2: DTAV Overview

4 DISTRIBUTED TRACKING AND VERIFYING (DTAV) METHOD

The overall design of DTAV is shown in Fig. 2. In DTAV, the tracker maintains a buffer of frames which makes the tracing back efficient. The verifier uses Siamese network to perform object tracking. We initialise a threshold T_1. If the verification result from the tracker is less than this threshold, then verifier will consider this as a tracking failure. Now, the verifier performs object tracking and generates its results. The frame is passed to the trained Siamese network which tracks the object and produces output score. If the score for this result is less than the threshold T_2, then the tracking result is not changed. Instead we decrease the verification interval. This helps in increasing the local region to track the target. This process is repeated until the verification score is greater than the threshold T_2. These T_1 and T_2 values are taken as 1.0 and 1.6 respectively. The original verification result is restored after this. If the object tracking results generated by the fDSST algorithm on the client side are close to the score generated by the Siamese network, then the Server sides provides a positive feedback. In the other case, the server's feedback contains the corrected coordinates for the given input frame.

However, deploying DTAV in an IoT setup is challenging because we need to consider parameters such as heterogeneous client-server speed, network environment etc. We perform a series of experiments to decide the optimal parameters for our communication channel. In a client-server setup it is critical to determine the network parameters like communication frequency, bandwidth and FPS.

First, the correct verification interval for communication between the client and server is very important. It is critical to determine the optimal verification interval for the setup. Larger verification intervals can lead to decrease in the accuracy of the model. This is because if an object is moving relatively faster in the video frames, then having a larger verification interval may lead to skip the object tracking verification. For example, consider a scenario

Algorithm 1 DTAV: Client side process

```
for every the x-th frame in video do
    tracker algorithm processes xth frame
    if msg received from server side : then
        if verifier needs to correct tracker result for a frame : then
            trace back to the frame
            correct results
            resume tracking
        end if
    end if
    if x%verification interval==0 : then
        send the client side result and frame to server side for verification
    end if
end for
```

Algorithm 2 DTAV: Server side process

```
if request received from client side then
    process the frame by verifier algorithm
    if verification passed : then
        send positive feedback to client
    end if
    if verification not passed : then
        send correct tracking results as feedback to client
    end if
end if
```

where client side verifies output with the server at every 20th frame, but a tracker object appears and disappears from the frame before 20th frame. Thus, this object would skip the verification from the server side. Additionally, a smaller verification interval would lead to decrease in FPS which could affect the accuracy. It would decrease the speed of execution because smaller interval means more frequent communication between client and server.

Second, how to update the verification interval is very important. This is because, when the verifier provides its feedback to the client. But, if the verification interval is too large, the verifier would not be able to provide the correct score. Thus, we prefer to make the verification interval adjustable. Depending upon the verification score, the interval is kept on decreasing if required, till the score generated by the verifier passes threshold T_2. This concept is further discussed in the experimental setup section.

5 PERFORMANCE EVALUATION

In this section, we begin by describing our system implementation followed by discussing the experiments performed. In order to understand the efficiency of our approach, and to fit the correct parameters we run the experiment in different scenarios and pick the most optimum values.

5.1 System Implementation

We have implemented this client-network architecture using gRPC, an open-source high-performance Remote Procedure Call (RPC)

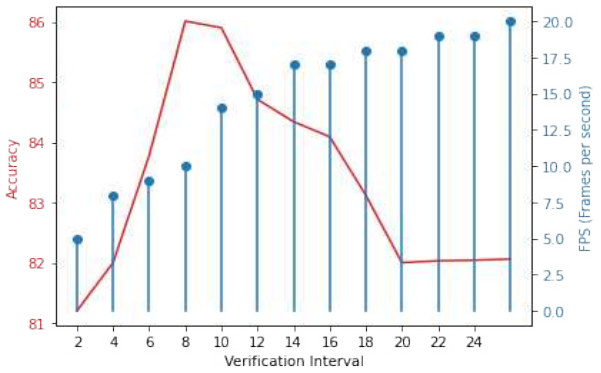

Figure 3: Verification Interval vs FPS vs Accuracy

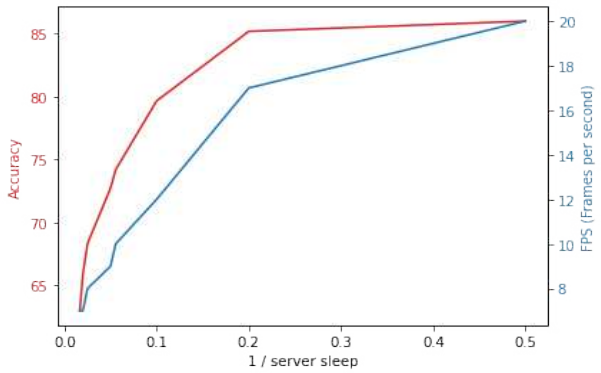

Figure 4: Simulating different server speed

framework developed by Google that can run in many environments such as C, Go, Python, Java etc. For simplicity and ease of implementation, we have chosen a Unary RPC approach where the client sends a single request and gets back a single response. The client invokes a stub (server) method with a Request message object. Once the server has the client's request message, it does whatever work is necessary to create and populate a response. The response(if successful) is then returned to the client which completes the call on the client side. Once the gRPC setup is done, we need to tune our setup with optimum bandwidth, verification interval and client and server speeds.

5.2 Experimental Setup

We performed a series of experiments to determine the optimum network parameters to get best performance for the PTAV client server setup. We varied the verification interval and observed the changes in accuracy and bandwidth. Through these experiments we were able to chose the optimum verification interval values. Moreover, we run the experiment in different bandwidth settings to simulate the performance of the network setup in different devices. Further we also compare the performance of our proposed PTAV with the default PTAV in client server setup.

5.3 Experimental Results

Effect of Verification Interval on Accuracy: As we discussed in the previous section, the client sends frame to the server on regular intervals which are predefined. Verification interval has a significant role to play in determining the accuracy. This is because the algorithm hosted by server, corrects and verifies the tracking results obtained from the algorithm hosted by the client. Figure 3 shows the effect of verification interval on accuracy. We observe that if the verification interval is too large, for example more than 20, then some of the real time tracking results will not be accurate. Else, if the verification interval is small, for example less than 5, then this would have effect on the real time tracking. FPS and hence the real time tracking would be affected. We empirically derive the optimum accurate verification interval to begin with, which is 10 from Figure 3.

Effect of Verification Interval on FPS: Verification interval has effect on speed of the framework. Figure 3 shows the effect of verification interval on FPS. We observe that smaller verification interval, has smaller FPS. This is because smaller verification interval causes the algorithm to be computationally expensive. Larger verification interval implies less communication between the client and server which leads to faster FPS, but this reduces the accuracy. From the graph, we also see that, FPS increases up to a certain value depending upon the system limitations. In our case the highest FPS achieved was 20 and then the value is constant, this is because of system limitations.

Adjust Communication frequency vs non-adjustment: Client and server communicate at fixed interval and exchange the object tracking information. This fixed interval communication (non-adjustment) has disadvantages. Running DTAV in non-adjusting setup gave accuracy of 82% while running DTAV in adjusting setup gave accuracy of 86.90%. This is because if the target appearance is changing quickly, then a fixed verification interval will not be able to adjust itself to capture it. Hence, a threshold is defined which determines from accuracy score whether the verification interval needs to be reduced. In such cases, the verification interval is decreased till the accuracy score is above the threshold. This helps in sharing more details of the local region with the server (verfier). After we obtain a valid result, the verification interval is restored. Hence, the adjusting verification interval is implemented in the algorithm.

Impact of Server Speed: We highlight the impact of server speed on the accuracy. We simulate different client speeds, by using sleep functionality. In Figure 4 we plot accuracy on y-axis and plot the inverse of server sleep time on x-axis. As we expected, we observe that efficient server can help achieve better accuracy. This is because we run the Siamese network on the server side, which is computationally heavier than the correlation based fDSST tracker used on the client side.

Comparison of PTAV and DTAV: We compare our results with the results of PTAV framework. While PTAV framework achieves accuracy of 79.12% on OTB2015 dataset with highest 17 fps, our framework obtains accuracy of 86.90 % with 20 fps. These results were achieved with 18.8 Mbps bandwidth. Further, we compare the PTAV approach versus the performance of DTAV. We observe that the values and patterns derived from the series of experiments have resulted in significant increase in the FPS and accuracy. Further, our framework would be useful in deploying this algorithm on embedded devices in different network setups. Different embedded devices operate on different bandwidths. We choose to setup the

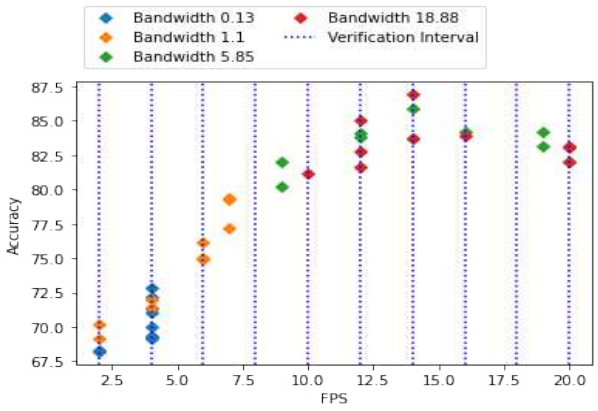

Figure 5: Accuracy vs FPS for different Verification intervals and different bandwidths

different bandwidths as CAT1 0.13 Mbps, 3G 1.1 Mbps, 4G 5.85 Mbps, Wi-Fi 18.88 (Mbps). We simulate different bandwidths to check the performance of our algorithm on such devices. Moreover, it is generally seen that the client speed is generally lesser than the server speed. To setup such real world scenarios, we reduce the speed of our client by 0.25x by adding latency in the Siamese network verifier algorithm. In the Figure 5, we observe how the change in bandwidths affect on the verification interval and accuracy. Moreover, in Figure 5 we also display the variations in FPS with respect to the Verification interval. We observe that bandwidths 5.85 Mbps and 18.88 Mbps give nearly the same values in the graph. For comparative study between the behavior of default client server PTAV setup versus our proposed client server PTAV setup, we compare their behaviors in 18.88 Mbps and 1.1 Mbps bandwidth as shown in Table 1. From Table 1, it is clear that DTAV outperforms DTAV in both tracking accuracy and tracking speed. In particular, DTAV can improve 33% FPS in the best scenario.

6 CONCLUSION

In this paper we present a distributed approach to perform visual object tracking. The distributed framework approach discussed in this paper aims to address speed and accuracy trade off challenge faced by other tracking algorithms. We run fDSST tracking algorithm on the client side which communicates with the Siamese network algorithm on the server to obtain accurate and faster results. We further perform a series of experiments to determine the optimum bandwidth and verification interval for our setup to provide better speed and accuracy. The experimental results show that DTAV achieved 7.78% improvement on accuracy and 15% improvement in FPS. This attributed approach is useful for efficient tracking algorithms to be deployed on IoT devices like smartphones.

ACKNOWLEDGMENTS

The paper is supported by National Science Foundation CPS Program Award Number: 2128341, 2128350 and 2006665.

2*Verification Interval	Accuracy		FPS	
	PTAV	DTAV	PTAV	DTAV
2	72.40	81.20	5	10
4	73.71	81.67	5	12
6	74.12	82.78	8	12
8	74.13	85.01	10	12
10	76.09	83.09	12	14
12	73.89	83.71	12	14
14	73.72	83.94	15	16
16	75.76	**86.90**	15	**20**
18	76.59	83.12	16	20
20	**79.12**	82	**17**	20

2*Verification Interval	Accuracy		FPS	
	PTAV	DTAV	PTAV	DTAV
2	65.49	69.17	5	10
4	66.02	70.23	5	12
6	67.12	71.39	8	12
8	68.98	72	10	12
10	69.11	74.90	12	14
12	71.21	75	12	14
14	72.67	76.13	15	16
16	73	77.21	15	20
18	75.34	**79.32**	16	**20**
20	**76**	79.29	**17**	20

Table 1: Comparison of DTAV method with PTAV method with bandwidth 18.8 Mbps (First Table) and with bandwith 1.1 Mbps (Second Table)

REFERENCES

[1] H. Fan and H. Ling, "Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5486–5494.
[2] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Discriminative scale space tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 8, pp. 1561–1575, 2017.
[3] B. Blanco-Filgueira, D. García-Lesta, M. Fernández-Sanjurjo, V. M. Brea, and P. López, "Deep learning-based multiple object visual tracking on embedded system for iot and mobile edge computing applications," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5423–5431, 2019.
[4] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 7, pp. 1442–1468, 2013.
[5] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *SIGARCH Comput. Archit. News*, vol. 45, no. 1, p. 615–629, apr 2017. [Online]. Available: https://doi.org/10.1145/3093337.3037698
[6] Z. Zhao, K. M. Barijough, and A. Gerstlauer, "Deepthings: Distributed adaptive deep learning inference on resource-constrained iot edge clusters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2348–2359, 2018.
[7] C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic adaptive dnn surgery for inference acceleration on the edge," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 1423–1431.
[8] S. Zhang, Y. Li, X. Liu, S. Guo, W. Wang, J. Wang, B. Ding, and D. Wu, "Towards real-time cooperative deep inference over the cloud and edge end devices," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 4, no. 2, jun 2020. [Online]. Available: https://doi.org/10.1145/3397315
[9] A. Banitalebi-Dehkordi, N. Vedula, J. Pei, F. Xia, L. Wang, and Y. Zhang, "Auto-split: A general framework of collaborative edge-cloud ai." New York, NY, USA: Association for Computing Machinery, 2021.
[10] N. Wang, Y. Duan, and J. Wu, "Accelerate cooperative deep inference via layer-wise processing schedule optimization," in *2021 International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2021, pp. 1–9.
[11] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, pp. 225–234.