MOBILE AUGMENTED REALITY OBJECT DETECTION APPLICATION

Jan Strnad[1], Jaromír Landa[1]

[1]Computer science department, Faculty of Business and Economics, Mendel University in Brno, Zemědělská 1, 613 00 Brno, Czech Republic

ABSTRACT

This article proposes a Mobile Augmented Reality (MAR) application for object detection. The application can detect predefined objects in the camera stream and display information about them. Object detection poses many challenges, and a common approach is to perform it remotely on a server. However, this requires an active internet connection. Alternatively,detection can be performed locally using a model stored on the device. How- ever, not all devices have the capability to perform real-time detection. We have created a Mobile Augmented Reality app that can detect objects in the camera stream. The app can perform detection locally or remotely, depending on the device's configuration. Sec- ondly, the app's ability to perform detection locally or remotely makes it versatile. The paper has two main contributions. Firstly, the proposed application architecture can be applied to any similar MAR app. The application was tested on multiple Android devices to determine the minimum configuration required for local object detection.

Keywords: object detection, Mobile Augmented Reality, Tensor Flow

JEL Code: L63, L86

1 INTRODUCTION

Machine vision, including artificial intelligence, plays a crucial role in various mobile ap- plications. One of the most common groups of applications is Mobile Augmented Reality (MAR), which allows virtual objects to be visualised in real-world environments. MAR ap- plications are typically operated on users' mobile devices, such as their phones (Zhou and Zhao, 2022). MAR applications utilise object detection to identify real-world objects in a camera stream and the augmented reality principle to visualise information specific to the detected object(s). However, object detection presents several challenges. There are three main approaches to performing object detection with mobile devices: a) on the server, b) on the device, or c) a combination of both (Ghasemi et al., 2022). Ghasemi et al. (2022) published a review that clearly demonstrates the prevalence of processing on a remote server. In this method, the camera image from a device is sent to the server, where object detection is performed. This approach has a clear advantage in that it utilises the compu- ting power of the server rather

https://doi.org/10.11118/978-80-7509-990-7-0159

than relying on the device (Ghasemi et al., 2022). However, it also presents several challenges, with communication latency being the most signifi- cant. Edge computing can partially eliminate this issue by performing detection on a local server connected to the same Wi-Fi network as the device (Liu and Han, 2018). However, as the computing power of mobile devices increases every year, many applications now use local models to perform necessary tasks. Some applications use a combination of server and device, with the device used to find the ROI and the server used to recognise specific objects (Knez and Šajn, 2020).

This paper presents a system for object detection that can be performed locally or remotely on a server. However, it is important to note that object detection is just one aspect of the entire MAR process. The model is first trained and then prepared for local or remote object detection using the TensorFlow framework. Once the object is detected, information about it is visualised using augmented reality. The Android platform is used to test the proposed methods.

The contribution of this work is twofold. First, we have proposed and developed an object detection and AR visualisation system. The system decides whether the device is capable of fast local object detection. If so, the detection is performed locally. Otherwise, the image is sent to a remote processing server and the object location information is sent back to the device. Secondly, we carried out performance tests. We tested object detection on several Android devices and compared the results with remote object detection using two different client-server communication technologies: REST API and Web Socket.

The rest of the paper is organised as follows. In the next chapter, we review recent articles on object detection. The ‚Methods and Materials' section outlines the proposed system and includes the test methodology. The results of the tests are presented and discussed in the ‚Results' section. Finally, concluding remarks are given in the ‚Discussion and Conclusions' section.

2 LITERATURE REVIEW

Object detection is the process of identifying a specific object or class of objects in a com- puter image. There are various algorithms available for object detection, with the most common being Convolutional Neural Networks (CNN). These algorithms can be classified into two main categories: two-stage and one-stage detectors (Martinez-Alpiste et al., 2022). Two-stage detection involves dividing the image into separate parts. The algorithm takes each input part and, after passing through convolution and pooling layers, outputs the object classes (Ghasemi et al., 2022). In contrast, one-stage object detection algo- rithms like SSD (one-stage single shot detectors) or YOLO (You Only Look Once) identify objects with just one pass through the image. The output of all detectors is the location of the object's bounding box in the image (Xiong et al., 2021). Two crucial requirements for object detection are speed and accuracy. Typically, there is a trade-off between the two, where higher speed results in lower accuracy. Object detection can be performed in three ways: on the server, on the client device, or a combination of the two (Ghasemi et al., 2022).

2.1 Server-side object detection

Server-side object detection is the most common type. Images are sent to a remote server where detection takes place. This type of detection requires high accuracy and low end- to-end network latency. However, low latency can significantly reduce accuracy due to changes in the user's view (Liu, Li and Gruteser, 2019).To optimize this process, several techniques can be employed, such as edge computing, federated learning, or software- defined networks (Xiang, Seeling and Fitzek, 2021). Edge computing reduces latency by performing computations closer to the source device on a local network (Liu and Han, 2018). It is important to note that training can also be demanding on device or server performance, not just object recognition.

Zhou and Zhao (2022) propose the use of Fed- erated Learning, which allows each device to train a shared model collaboratively without sharing local data with others.

2.2 Client-side object detection

The primary benefit of client-side object detection is its offline capability and privacy. The model is stored locally on the device where the detection is performed, eliminating the need for a remote server. As the processing is done locally, privacy is also a significant advantage. Personal images are not sent to a remote service (Savchenko, Demochkinb and Grechikhinb, 2022). Several frameworks are used for local object detection, including TensorFlow Mobile (TFM), TensorFlow Lite (TFL), OpenCV, and Qualcomm Snapdragon (Martinez-Alpiste et al., 2022). However, local detection can be energy-intensive (Apicharttrisorn et al., 2019) and demanding on devices with limited computational power (Martinez-Alpiste et al., 2022). There are several ways to optimize the process. Cai et al. (2020) propose a cooperative scheme between the GPU and the CPU for processing.

2.3 Combination of server-side and client-side object detection

Object detection requires a tradeoff between speed and accuracy. To partially solve this issue, Li et al. (2022) proposed a parallel offloading scheme that combines server-side and client-side processing. The mobile device is used to detect large objects and regions of interest (ROIs) containing small objects, while small objects are detected on a remote server. Wang et al. (2022) also employ this principle for autonomous mobile vision. Small object detection is delegated to the edge. Both Li et al. (2022) and Wang et al. (2022) demonstrate significant improvements in accuracy.

3 METHODOLOGY AND DATA

This section outlines the proposed object detection system, which aims to provide fast and accurate object detection on all devices, regardless of their processing power. The system has been designed to detect objects either on a remote server or on a local device. The minimum requirements for Android devices to perform local detection are specified. If a device is not powerful enough to perform local detection, the remote server is used. The second objective is to evaluate the effectiveness of REST API and Web Sockets for real-time object detection. To test the proposed system, both a remote server and an An- droid application were developed. The application captures camera feed, performs object detection either on the server or on the device, and presents information about the de- tected objects in augmented reality using the SceneForm framework.

3.1 Remote server-based object detection

The goal of the server-based approach is to offload the performance-intensive object de- tection from the device to a remote server. The device captures an image from the camera and sends it to the remote server. The server receives the image, processes it and per- forms the detection. Objects are detected using the TensorFlow library. This library re- quires trained models in the tflite format. This format was chosen for its known compati- bility with mobile devices and can also be used for server-side detection. The tflite format is used by many projects, such as Chilukuri, Yi and Seong (2022) and Azzo, Taqi and Mila- nova (2018).

Once the detection process is complete, the server sends a list of detected objects (label, confidence and bounding box) back to the device in JSON format. The device pro- cesses the list and then renders the detected objects in augmented reality directly into the camera stream.

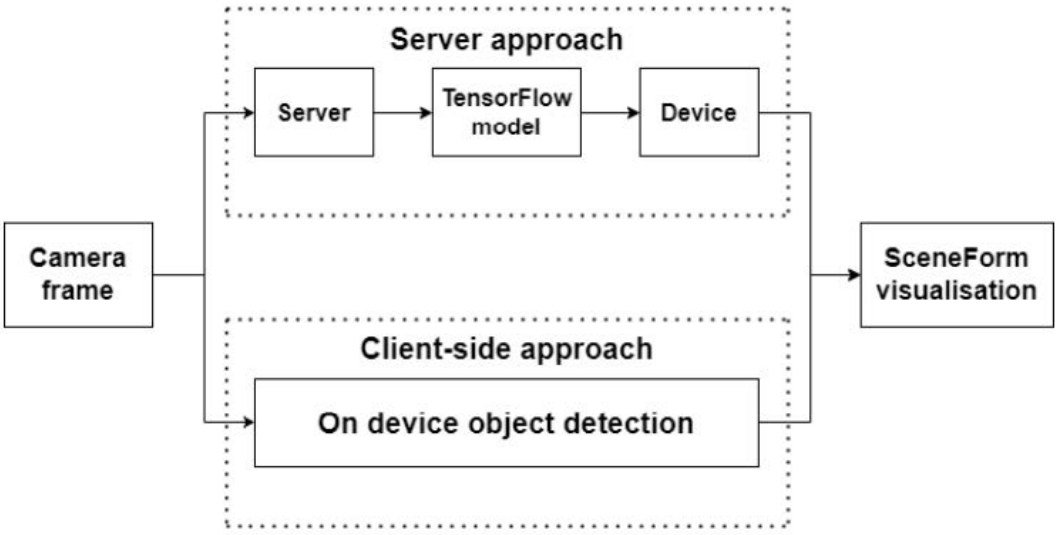

Fig. 1: Object detection process

This process illustrates the drawback of server-based detection. The image from the camera has to be sent to the server, which can take a long time. The two most common communication schemes today are REST API and WebSockets. Both REST API and Web- Sockets have been implemented and tested.

3.2 Client-side object recognition

This approach removes a major disadvantage of the server approach, which is the need for a permanent Internet connection. The whole process can be done completely offline. Google‘s ML Kit library was used for detection. The ML Kit uses the same tflite model for- mat as the previous server-based approach. In the case of client-side detection, an image from the camera is passed directly to the detector, which returns a list of the resulting detected objects.

4 RESULTS

To test the functionality and performance of each approach, we conducted tests on both. In terms of server-side object detection, we compared the time it took to communicate using REST API and Web Sockets. Table 1 clearly shows that Web Sockets have a signifi- cant advantage over REST API. Regarding the REST API, the time it takes for requests and responses to travel between the server and device is slower due to the need to send mul- tiple images per second to the server and receive the resulting objects in a timely manner. The detection speed on the server side remains the same when using the same model and detection method to identify objects.

To compare client-side object detection, we compared seven different mobile phones with varying configurations (refer to Table 2). The tested mobile phone devices vary with different configurations and Android version to test the detection on commonly used de- vices. As this is an offline process, we did not consider the network load of the phones. We established several metrics for comparison, as described above. The results in Table 2 show that the RAM parameter is one of the main attributes affecting the resulting detec- tion time. Moreover, newer processors exhibit significantly improved performance in ob- ject detection, as confirmed by the CPU usage metric. Moreover, newer processors exhibit significantly improved performance

Type	Request	Detection time	Response	Totaltime
RESTAPI	228.451 ms	99.075ms	206.258 ms	533.784 ms
Websocket	34.364ms	99.075ms	25.397ms	158.836ms

Tab. 1 The results of server-side object detection

in object detection, as confirmed by the CPU usage metric. Moreover, newer processors exhibit significantly improved performance in object detection, as confirmed by the CPU usage metric. The more advanced processors are no- ticeably less burdened.

The findings indicate that newer and more powerful devices have a distinct ad- vantage in client-based object detection, as it saves time and eliminates the need for a constant internet connection. However, for older phones, detection time is longer com- pared to the server approach that uses web sockets. In such cases, it is recommended to perform object detection on the server. Real-time use can omit detection using REST API communication.

As previously stated, our system adjusts to current conditions. Through testing, we have determined the necessary requirements for a mobile device to perform local object recognition: a) a minimum of 6 GB of RAM, b) new octa-core processors, and c) Android version 10 or higher. The results show the advantage of using newer mobile phones over older models in both hardware configuration and available Android OS version.

Refer to Fig. 2 for the final visualization of the detection.

Device name	CPU	RAM	Detection time	RAM usage	Android version	CPU usage
Samsung S21 FE	Snap- dragon 888	6 GB	14.37 ms	444 MB	13	29%
Pixel 5	Snap- dragon 765G	8 GB	71.31 ms	600 MB	13	46%
Samsung S8	Snap- dragon 835	4 GB	160 ms	500 MB	9	49%
Pixel XL	Snap-dragon 821	4 GB	213 ms	700 MB	10	44%
Xiaomi Mi A3	Snap-dragon 665	4 GB	236 ms	600 MB	11	52%
OnePlus 7 Pro	Snap-dragon 855	8 GB	25 ms	437 MB	11	25%
Pixel 6	Google tensor	8 GB	24.7 ms	600 MB	12	38%

Tab. 2 Object detection process

Fig. 2: Visualization of the detection process using Scene Form framework.

5 DISCUSSION AND CONCLUSIONS

This paper proposes a Mobile Augmented Reality application for object detection. Object detection can be performed either server-side or client-side, each with its own advantages and disadvantages. Server-side detection offers the advantage of a central processing point without the need to download the trained model to the device, making model up- dates more efficient. However, it has the disadvantage of slower processing time. Even when using web sockets, the time required for detection is much longer than that of local, powerful, and new devices. The use of standard REST API communication is only useful for applications that do not require real-time object detection. Over time, communication using Web Sockets is much faster and more efficient due to its parallel capabilities. The Web Socket allows for simultaneous request reception and response transmission. This means that when the server sends detected objects, it can already be receiving the next image to perform a new detection. In contrast, with the classic REST API, communication is only possible in one direction, which leads to increased detection and response time over time.

Based on the tests, client-side object detection was better in most cases. However, this was not the case with older devices. However, client-side detection can lead to increased memory and CPU usage due to the number of detected objects stored. As mentioned in Ghasemi et al. (2022) there is a trade-off between speed and accuracy of object detection. In our case, the accuracy did not change, since the object were always detected, however the speed of the detection can vary significantly based on the device configuration. Our results prove that with newer devices, this trade-off becomes less significant as the speed of the detection significantly increases. The results show the clear advantage of client-side detection over server-side.

Therefore, we have set minimal requirements for Android mobile devices to perform object recognition. If these requirements are not met, it is recommended to perform ob- ject detection server-side. The minimal requirements are a) at least 6 GB of RAM, b) new octa-core processors, and c) Android version 10 or higher. If the requirements are not met, the system will switch from local detection to remote server-side detection.

Acknowledgements

Supported by grant No. IGA-PEF-TP-22-006 (Opportunities to use metaverse technology to support business processes) of the Internal Grant Agency FBE MENDELU.

This paper was supported by the project CZ.02.1.01/0.0/0.0/16_017/0002334 Research Infrastructure for Young Scientists, this is co-financed from Operational Programme Research, Development and Education.

REFERENCES

APICHARTTRISORN, K., RAN, X., CHEN, J., KRISHNAMURTHY, S. V. and ROY-CHOW-DHURY, A. K. 2019. Frugal following: power thrifty object detection and tracking for mobile augmented reality. In: *Proceedings of the 17th Conference on Embedded Net- worked Sensor Systems (SenSys ,19)*. Association for Computing Machinery, New York, NY, USA, 96–109. https://doi.org/10.1145/3356250.3360044

AZZO, F., TAQI, A. M. and MILANOVA, M. 2018. Human Related-Health Actions Detection using Android Camera based on TensorFlow yObject Detection API. *International Journal of Advanced Computer Science and Applications,* 9(10). https://doi.org/10.14569/IJACSA.2018.091002

CAI, Y., Li, H., YUAN, G., NIU, W., Li, Y., TANG, X., REN, B. and WANG, Y. 2020. YOLObile: Real-Time Object Detection on Mobile Devices via Compression-Compilation CoDesign. *arXiv*. https://doi.org/10.48550/arXiv.2009.05697

GHASEMI, Y., JEONG, H. CHOI, S. H., PARK, K. and LEE, J. Y. 2022. Deep learning-based object

detection in augmented reality: A systematic review. *Computers in Industry*, 139, 103661. ISSN 0166-3615. https://doi.org/10.1016/j.compind.2022.103661

CHILUKURI, D. M., YI, S. and SEONG, Y. A robust object detection system with occlusion han- dling for mobile devices. *Computational Intelligence*, 38(4): 1338-1364. https://doi.org/10.1111/coin.12511

LI, X., QIN, Y., LIU, Z., ZOMAYA, A. and LIAO, X. 2022. Towards efficient and robust intelligent mobile vision system via small object aware parallel offloading. *Journal of Systems Architecture*, 129, 102595. ISSN 1383-7621. https://doi.org/10.1016/j.sysarc.2022.102595

LIU, Q. and HAN, T. 2018. DARE: Dynamic Adaptive Mobile Augmented Reality with Edge Computing. In: *IEEE 26th International Conference on Network Protocols (ICNP)*. Cambridge, UK, 2018, pp. 1-11. https://doi.org/10.1109/ICNP.2018.00011

LIU, L., LI, H. and GRUTESER, M. 2019. Edge Assisted Real-time Object Detection for Mo- bile Augmented Reality. In: *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom ,19)*. Association for Computing Machinery, New York, NY, USA, Article 25, 1–16. https://doi.org/10.1145/3300061.3300116

KNEZ, S. and ŠAJN, L. 2020. Food object recognition using a mobile device: Evaluation of currently implemented systems. *Trends in Food Science & Technology*, 99, 460-471. ISSN 0924-2244. http://doi.org/10.1016/j.tifs.2020.03.017

MARTINEZ-ALPISTE, I. et al. 2022. Smartphone-based real-time object recognition archi- tecture for portable and constrained systems. *J Real-Time Image Proc.,* 19. https://doi.org/10.1007/s11554-021-01164-1

SAVCHENKO, A. V., DEMOCHKIN, K. V. and GRECHIKHIN, I. S. 2022. Preference prediction based on a photo gallery analysis with scene recognition and object detection. *Pattern Recognition*, 121, 108248. ISSN 0031-3203. https://doi.org/10.1016/j.patcog.2021.108248

THIEN, H., PHAM, Q., PHAM, X., NGUYEN, T. T., HAN, Z. and KIM, D. 2023. Artificial intelligence for the metaverse: A survey. *Engineering Applications of Artificial Intelligence*, 117(Part A), 105581. ISSN 0952-1976. https://doi.org/10.1016/j.en- gappai.2022.105581

WANG, X., YANG, Z., WU, J., ZHAO, Y. and ZHOU, Z. 2021. EdgeDuet: Tiling Small Object Detection for Edge-Assisted Autonomous Mobile Vision. In: *IEEE INFOCOM 2021 – IEEE Conference on Computer Communications*. Vancouver, BC, Canada, pp. 1-10. https://doi.org/10.1109/INFOCOM42981.2021.9488843

XIONG, Y. et al. 2021. MobileDets: Searching for Object Detection Architectures for Mo- bile Accelerators. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3824-3833. https://doi.org/10.1109/CVPR46437.2021.00382

ZHOU, X. and ZHAO, J. 2022. Mobile Augmented Reality with Federated Learning in the Metaverse. 2022. *arXiv*. https://doi.org/10.48550/arXiv.2212.08324

Contact information

Jan Strnad: e-mail: strnad.hon@gmail.com
Jaromír Landa: e-mail: jaromir.landa@mendelu.cz