To appear in 2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW).

Towards a Platform for Smart City-Scale Cognitive Assistance Applications

Thomas Rausch* TU Wien
Waldemar Hummer TU Wien
Christian Stippel TU Wien
Silvio Vasiljevic TU Wien
Carmine Elvezio Columbia University
Schahram Dustdar TU Wien
Katharina Krösl† VRVis Forschungs-GmbH

ABSTRACT

This paper describes CognitiveAR, a system that seamlessly interfaces AR devices with smart city environments. Edge computing nodes distributed throughout the city enable multi-user cognitive assistance applications that require (1) real-time sensor data from the environment, such as approaching cars, and (2) computing resources for low-latency video processing. We discuss three such applications to elicit requirements for a platform to support them, and present our system design.

Index Terms: Computer systems organization—Architectures—Distributed architectures; Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Mixed / augmented reality

1 INTRODUCTION

Cognitive augmentation has been explored by researchers, artists, and science fiction writers alike. Modern wearable augmented reality (AR) devices, the recent advancements in artificial intelligence (AI) techniques, and the unprecedented availability of sensory data from the environment and digitally persisted knowledge, have created exciting new opportunities for enhancing human cognition. Cognitive assistance applications use these technologies to synthesize an overlay that assists people with everyday tasks. These applications can help people with cognitive deficits by giving them hints about social interactions [11, 27], improve the safety of cyclists in urban areas by highlighting dangerous situations, or improve situational awareness of emergency response teams.

Smartglasses are challenged to enable a wide range of real-time applications, unless the devices are highly optimized for a specific use case using ASICs. For AR devices to be practical, it is clear that we will not own a dozen different devices, one for each use case. General-purpose AR devices are the future, but it is challenging to develop cognitive assistance applications, because such applications require real-time access to context-specific data from the environment, or need to run complex computer vision models that are too computationally expensive to run on the device itself. Offloading AI tasks to do real-time analytics on the video and audio feeds from the wearable, as well as processing sensor data in real-time on the wearable, will be key. Using cloud-based solutions, such as Azure Cognitive Services, incurs too much latency to provide a crisp user experience [11, 26]. Edge computing [31] has been recognized as a key enabler for this new family of cognitive assistance applications [21, 26, 27]. Smart city sensor arrays and cameras deployed throughout the city [1], as well as computational resources with AI accelerators at the edge of the network [21], are key technologies to make cognitive assistance work in urban areas.

*e-mail: t.rausch@dsg.tuwien.ac.at
†e-mail: kroesl@vrvis.at

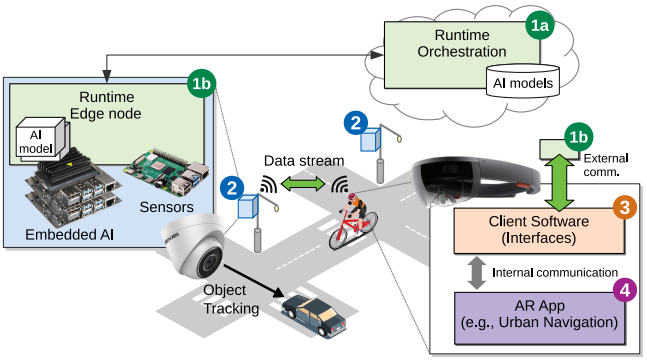

Figure 1: The main architectural components of our system: (1) Runtime platform that provisions AI services from a repository in the cloud (1a) to edge nodes (1b), (2) edge node, such as a smart lamp post, providing compute resources and access to environmental sensor data; (3) device-specific client software on an AR or mobile device, that facilitates transparent access to the system; (4) actual AR application running on the specific device.

Efforts to develop edge computing platforms for multi-user cognitive augmentation that can scale to an entire city while providing real-time interaction is still niche research [11, 16, 34]. There are experimental industry solutions to support low-latency AI services at the edge. For example, Microsoft Azure Live Video Analytics uses Azure IoT Edge to deploy cognitive services on edge devices, allowing real-time AI-based video analytics. The complexity of these heterogeneous and highly distributed computing systems makes it hard for AR engineers to operate the necessary application services (such as AI models for object detection), or access sensor data in real time. Platforms should do more of the heavy lifting, and provide common application functionality such as tracking objects in a global coordinate space, enabling precise device positioning, and abstracting access to distributed infrastructure.

In this paper, we propose CognitiveAR: a platform that makes it easier for AR engineers to develop scalable cognitive assistance applications. Figure 1 shows the overall architecture. The platform federates distributed edge resources to run our platform services, and autonomously manages application-specific services for, e.g., offloading AI-based video processing. Moreover, the platform provides transparent access to sensor data published by edge nodes, via a scalable messaging system. We introduce a global object and user tracking system which we call cyber-physical object positioning (CPOP). CPOP enables the system to accurately display the position of physical objects on the user's device, that are not in their physical field of view. Object coordinates are published as sensor data via the platform's messaging system. We discuss several cognitive assistance use cases that highlight requirements and challenges. We present early results that demonstrate (1) how developers can access the system via high-level APIs, (2) the limitations of cloud-based solutions, and (3) the feasibility of running CPOP on edge nodes.

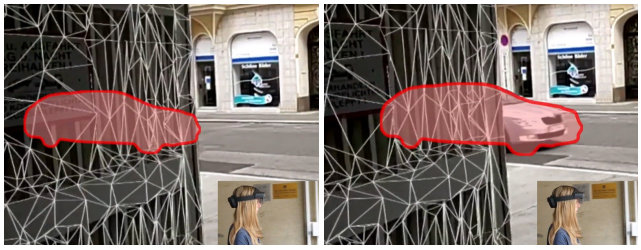

Figure 2: Silhouette of an approaching car behind a wall, as seen by a cyclist using our proposed smart bicycle glasses app.

2 Urban Cognitive Assistance Applications

Cognitive systems collect and process data from their environment with the goal of enhancing human cognitive abilities such as perception, learning, reasoning or planning [17]. As AR devices for assistive technology [12] become more widespread, the need arises to provide infrastructure and platform support on a larger scale, particularly for urban use cases. In this section we present three application scenarios to elicit requirements for such platforms. In subsequent research, we plan to implement these applications to demonstrate the efficacy of our system.

2.1 Application Scenarios

The Smart Bicycle Glasses We have seen a great bicycle boom in 2020 in response to the COVID-19 pandemic [5]. Navigating safely through the city on a bike has become more important than ever. We believe that a navigation system that displays the route directly augmented over the real road is superior to apps on a smartphone that force the cyclist to draw their gaze away from the road to look at a phone screen. We envision an AR app, running on AR glasses such as the Nreal Light, that provides navigation and safety features. To prevent accidents with cars at street crossings, our smart bicycle glasses notify the cyclist of an approaching vehicle behind a building or other occluders, well before it reaches the crossing or can be seen by the cyclist. The occluded car is visualized as a silhouette or bounding box blended over the real-world view of the cyclist, as shown in Figure 2. To provide this kind of safety feature, the smartglasses need to receive data about approaching cars (size, location) in a format that can be translated to the coordinate frame of the AR device. A distributed camera system in the city, with units mounted above street crossings [19], could provide a continuous video feed of the traffic from different angles. Sophisticated object recognition algorithms are necessary to analyze the video feed, detect cars, and estimate their position and speed in real time.

This use case demonstrates the requirement for AR devices to "see through walls", by accessing environmental sensor data such as the position of approaching cars. These data are synthesized by performing real-time video processing on urban cameras.

Cognitive Assistance for People with Disabilities Individuals living with autism spectrum disorder (ASD) often have difficulties with communication and social interaction, especially when it comes to interpreting facial expressions and emotions [3]. The American Psychiatric Association points out that even "*Adults who have developed compensation strategies for some social challenges still struggle in novel or unsupported situations and suffer from the effort and anxiety of consciously calculating what is socially intuitive for most individuals.*" [3]. There have been various studies using AR applications in interventions in people with ASD, showing promising results [4]. While we believe virtual and augmented reality applications are an excellent tool for training and therapy of cognitive abilities, we also want to support the development of assistive technology in this area. Our platform can provide the resources needed to develop cognitive assistance apps that help people with ASD, or similar cognitive disabilities, to understand their environment better and for example interpret emotions of other people better. Imagine a person with ASD, wearing a head-worn AR device when in a face-to-face conversation with another person. The cognitive assistance app recognizes and classifies the facial expression of the other person and provide this information as visual or verbal information to the user in real time. These and other similar applications proposed in [28] can have an immense impact on the lives of people with cognitive impairments.

Because analyzing the video feed from the camera of the AR device in real time is a computationally expensive task that may require specialized AI hardware, it is not feasible to do this computation on the AR device itself. Instead, the AR app needs to offload the video stream of the device's camera onto a more powerful computing platform where custom AI-based video analytics can be used to recognize faces and classify facial expressions in real time.

Emergency Response Applications Floods affect millions of people each year. In order to be prepared for these natural disasters, flood managers run simulations to develop flood plans and field personnel train in order to be prepared in case of emergency. Virtual Reality applications have already been used successfully to create safe and inexpensive training environments for emergency response training [10, 30] with predefined scenarios or sophisticated flood simulation systems [14]. The major limitation of these VR systems is that the virtual environments are still just approximations for the real world. Furthermore, they can only be used for training and planning and are less useful during a real emergency situation. Hence, we propose an AR app that shows the user live flooding data, captured by a city-wide distributed camera system, as well as simulation data to look at predictions of how the flooding event will unfold. Field personnel equipped with AR devices can see water levels rising a few streets away, or look at virtual markers augmented over the buildings next to them to see predicted water levels at a user-selected point in time. These data are provided by a connected flood simulation system, such as Visdom [35]. Furthermore, live camera data can be used to evaluate and refine the simulation at runtime.

For such an emergency response app to be effective, it needs live access to sensors, such as the mentioned distributed camera system, and fast object recognition to identify breaches in dikes and rising water levels. A virtual city model that would be used by a connected flood simulation system to calculate predictions, needs to be mapped correctly onto the real world. Furthermore, correct localization of water flooding the streets and field personnel in a shared global coordinate system is vital, to show flooding events at the correct location in real time.

2.2 Summary of Requirements

From our three use cases outlined above, we derive several key requirements. For our platform to support a wide range of cognitive assistance AR applications, it needs to:

(1) *Allow apps to access sensor data, providing environment information in real time*; Sensor data needs to be accessible as both raw data and as labeled data. The former allows a client application to visualize and act upon what the system is seeing, while the latter simplifies the user's decision making process by providing output from computationally intensive AI services.

(2) *Let apps offload the device's video feed onto a more powerful computing platform*; In order to maximize processing capability, and utilize the perspective seen by the users themselves, a client should be able to send a video feed (as captured on the AR device) to a computing platform that provides low-latency processing, in order to use that feed in combination with the data provided by other sensors.

(3) *Provide different AI services to enable a wide range of use cases*; A backend platform should provide a range of AI services accessible by any AR device, agnostic to their particular hardware features. The platform should allow the AR application to access

processed environmental sensor data (with additional processing of its own live video feed), in order to offload these computations, both to encourage more robust algorithms, and maximize device performance and battery lifetime.

(4) *Track users and objects*; The platform needs to be able to track objects in the environment and, privacy considerations permitting, users as well, and register both in a shared global coordinate space. This is useful for ensuring that virtual augmentations are displayed at the correct locations. In all cases, users' devices will track themselves, which is needed to place annotations.

3 Related Work

In the following we briefly discuss previous research in the area of cognitive assistance applications, edge-based AR platforms, and smart city edge infrastructure.

A number of academic systems have been developed to help in navigating around urban environments [2, 36]. One limitation shared between these types of systems is in the degree to which virtual augmentations are registered and localized to objects and users in the real world, at city scale, which a unified platform of augmentations would help to solve.

The work by Ha et al. introduces Gabriel [11], a wearable cognitive assistance system that uses edge nodes ("cloudlets") and introduces the concept of "sensor flows" and "cognitive flows". Whereas their approach uses cognitive VMs for different capabilities (e.g., face recognition, OCR, motion classifiers) and focuses primarily on offloading the processing from wearable devices, we utilize lightweight containers to provide composable AI services that are fully managed by the runtime across the lifecycle. Gigasight [28] is an Internet-scale repository of video content to perform edge analytics. While our work uses multi-modal AI and AR techniques to provide cognitive assistance, their approach focuses on video analysis and the enforcement of privacy and access policies. Our approach builds on existing work in the area of AR & edge computing platforms. Ren et al. [25] present a hierarchical computation architecture that connects the physical user to the cloud by means of an edge layer. Whereas their architecture uses static configurations of components deployed in either the cloud or the edge layer, our approach focuses on dynamic reconfiguration and orchestration of migratable AI service pods to achieve efficient and optimal processing. The DARE system [16] uses "edge servers" to offload image data from client devices in order to perform AR tasks like object detection. It focuses on the concept of "Quality of Augmentation (QoA)" and defines a protocol for dynamic adaptation of frame rate and compression factor. While DARE focuses on the particular issue of QoA, our approach more broadly supports the development of cognitive assistance applications.

Current industry solutions for edge-based data analytics include a Microsoft offering that deploys Azure Cognitive Services using its Azure IoT Edge system, or Amazon's AWS Greengrass. Our runtime platform architecture is similar in that it provides a container-based computing runtime on edge devices. However, it adds high-level platform support for object tracking and aggregating sensor data.

More and more smart city projects are deploying edge resources and sensors throughout the city. An example is the Array of Things [8], where several environmental data sensors, a camera, and two single-board computers for computation and communication, are enclosed in a weatherproof casing and mounted on lamp posts. They are interconnected using LTE, 5G, or WiFi. Similar smart city sensing projects focus on very specific urban phenomena, like traffic congestion, or air pollution, as discussed in [1]. There are projects that deal with the use and organization of pervasive camera networks [19, 32] but they do not include concrete technology proposals for edge nodes. We are looking into very specific ways of building edge nodes that can be deployed as smart city infrastructure to enable multi-user AR applications.

4 CognitiveAR: A Platform for Smart City-Scale Cognitive Assistance

We present CognitiveAR, a platform to enable applications described in Section 2. We first describe the architectural components of the system, shown in Figure 1. We then provide a detailed explanation of the core system mechanisms, shown in Figure 3.

4.1 System Overview

4.1.1 Runtime Platform

The runtime platform abstracts the underlying edge computing infrastructure and provides an execution environment for both platform and application services. We use a container-based edge computing system based on Kubernetes that provides additional platform-level facilities for operating AI applications [22]. The edge node runtime federates multiple edge computers of a node and provisions them on-demand. The runtime orchestration runs in the cloud and schedules application services to the edge on demand using a custom container scheduler [24]. Platform services include an object tracking positioning system called CPOP, a message-oriented middleware for device communication, and our own AI analytics offloading service CogStream. We provide several off-the-shelf AI models that are common across use cases and can be used as application services, much like Azure cognitive services. We also allow personalized or domain-specific models that application developers previously uploaded, and that are automatically deployed by the system where they are needed, for example based on physical proximity between users and edge nodes.

4.1.2 Edge Nodes

The edge nodes host the platform runtime for executing application services. They are composed of multiple edge computers, as well as cameras and other sensors, and are interconnected with each other via our messaging middleware. For the edge computers, we are currently experimenting with high-density computing hardware, such as NVIDIA's Jetson platform [18], and specialized AI hardware such as the Google Edge TPU [7]. Edge nodes should be compact and fit on, for example, a street lamp post. However, our platform is agnostic to the hardware, and can be used with any commonly proposed edge computing architecture [20]. We are currently investigating how different wireless technologies, like 5G and 5GHz WiFi, for connecting user devices to the nodes, as well as connecting the nodes to the internet, affect the overall responsiveness of the system.

4.1.3 Platform Client

The platform client provides a device-specific SDK that includes protocol implementations to communicate with our system. The SDK provides high-level APIs to (a) use CPOP to position objects correctly in the field of view, (b) send and receive sensor data from the platform via EMMA, (c) request and consume AI services through CogStream, and (d) access third-party application services. A connection manager proxies network connections in order to perform graceful handoffs between edge nodes, or mask network failures.

4.1.4 AR App

AR apps for devices like the HoloLens or Nreal Light smartglasses, are commonly implemented using Unity and the Mixed Reality Toolkit (MRTK). The purpose of our platform is to facilitate the development and operation of smart city-scale cognitive assistance applications, such as the ones we outlined in Section 2. In these applications, common tasks include accessing the video or sensor feed of the device and invoking remote services, or asynchronously acting on environmental sensor data like drawing the silhouette of a car that is being occluded by a house, as shown in Figure 2. These tasks are streamlined by the platform client SDK, thereby reducing boilerplate code in AR apps.

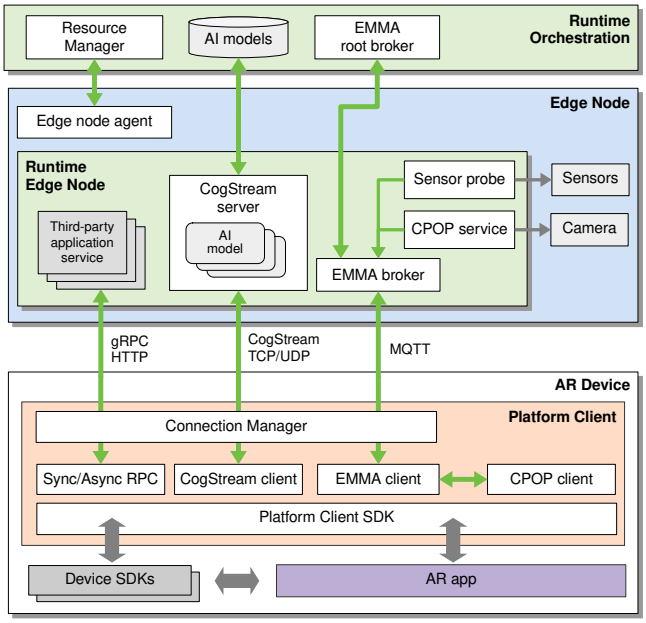

Figure 3: Internal architecture of our platform

4.2 Platform Mechanisms

We describe the core mechanisms of our runtime platform. These are designed to deal with the highly dynamic nature of smart city environments. Capabilities such as object tracking services, message brokers, or AI services are provisioned by the runtime system as containerized applications, using optimized deployment topologies.

4.2.1 Cyber-Physical Object Positioning (CPOP)

To correctly display virtual representations of physical objects in AR apps, our system tracks objects and devices in a global coordinate space. To track physical objects like cars, we use monocular video data from stationary cameras, and state-of-the-art deep learning models for detecting objects and determining their position. To track device positions, we synthesize data from mobile device location services, device tracking sensors, and spatial anchor services. Our system integrates these methods and applies sensor fusion techniques, similar to [13].

Object Tracking Physical object tracking involves several steps. First, objects of interest are detected in monocular frames using object detection algorithms such as *You Only Look Once* (YOLO) [15]. After the initial bounding boxes and classes of objects are determined, objects are tracked in real time by finding rotation and scale-invariant features per detected object and tracking them over multiple successive frames [37]. This approach solves the problem of occlusion, because occluded objects can be found and correctly identified when they reappear in the camera view. Additionally, it allows us to build a motion model for each object. This enables us to use time-intensive algorithms and compensate for latency by interpolating the position of detected moving objects over several frames and predicting their next location [29].

From the 2D camera image of the edge node, we generate a depth map by using existing deep learning models for depth estimation [38]. Combining this depth map with the undistorted 2D camera image and the intrinsic camera parameters allows us to approximate the 3D coordinates of tracked objects in the camera space. Our approach yields the object's position, bounding box, motion model and label in camera space.

User Tracking To be able to place virtual information at the correct position in the field of view of the user, we need to know the user's position relative to points of interest, such as tracked objects. An approximate global position could be estimated using GPS or WiFi data from a user's smartphone, but in most cases (e.g., for our smart bicycle glasses) we need higher accuracy. If a user is visible from a camera, they could be tracked like any other moving object. However, since our platform does not necessarily have video coverage over the whole city (with edge nodes installed only at strategic positions) it is possible that users are connected to the system while not tracked in a video stream. Our approach, which tries to determine the exact position of the user, requires an area-wide distribution of Azure Spatial Anchors (ASAs). ASA is a Software-as-a-Service offering that allows users to store and retrieve spatial-point-cloud information to localize themselves precisely at the same physical space. During the installation and setup of our platform, ASAs have to be manually placed at various positions around the city with their geolocation, and that of other nearby ASAs, uploaded to edge nodes in the proximity. An AR device can retrieve a list of ASAs from the closest edge node and try to find them in its field of view (FOV). Once an ASA is located, an AR device can record its own position relative to the ASA; thus, AR devices are capable to fully self-localize.

Global Coordinate Space Studies have shown that global navigation satellite systems (GNSS) have a mean accuracy of about 4.9 meters with smartphones under open sky and that the accuracy in urban environments correlates to building height [33]. Since a GNSS does not provide sufficient accuracy for our platform, we decided to construct a global coordinate system based on a city map and corresponding height field for altitude information. This is crucial to be able to register all platform components (edge nodes, tracked objects, ASAs, users) in the same shared global coordinate space and accurately position virtual objects even in hilly areas. Global coordinates of edge nodes and ASAs are recorded when they are installed or created (taking the height-field information into account). Relative coordinates from tracked objects can then be transformed from the camera space of edge node cameras to world space. Since users track themselves, the client software on the AR device can transform all global coordinates from world space to the view space of the user or transfer its own position to the platform. Virtual information, such as bounding boxes of tracked objects, can be positioned accurately in the field of view of the user. While sending user positions to the platform can be useful, our platform also supports operation without that, out of concern for privacy.

4.2.2 Low-latency Device Communication

We use the EMMA [23] messaging middleware to facilitate loosely coupled communication between devices. It is based on the Message Queue Telemetry Transport (MQTT) protocol, which is a lightweight binary publish–subscribe protocol, commonly found in Internet of Things scenarios. Devices that communicate in our system include AR devices, mobile devices, edge nodes, or cloud nodes. AR devices publish data, such as their location, to other participants in the system. Conversely, edge nodes use EMMA to publish sensor data, or coordinates of objects tracked by CPOP, to participants in proximity. Edge nodes can serve as brokers to facilitate efficient broadcast or multicast of messages for devices in proximity. When devices are mobile, the system takes care of connection handoff, such that the device is always connected to the closest broker. EMMA automatically deploys brokers to resources based on current demand, and re-configures client–broker connections to optimize latency. We add content-based message filtering that allows AR engineer to express queries to, for example, receive the coordinates of specific object types in a particular radius from the wearable.

4.2.3 CogStream AI Analytics System

CogStream is the subsystem that enables offloading of AI workloads from AR devices to edge nodes. Examples of such tasks are object detection, depth estimation, or interpreting facial expressions. Similar to systems like Gabriel [11], or DARE [16], CogStream provides a backend for serving AI models, and abstractions to access them. The selection of an appropriate AI model, the negotiation of the model parameters, and the data stream, are abstracted into the CogStream protocol. The CogStream client and server initiate a connection through a handshake, where the client communicates the type of data it will send (e.g., a video stream and its resolution), as well as the AI model it wants to use (e.g., detecting faces and their emotions). Based on this information, the server dynamically chooses the parameters to optimize the stream for the input of the selected AI model, such as the frame size, color space, and whether the client should do the frame transformation. This is orthogonal to the protocol presented in [16], that dynamically adapts the framerate to meet quality of experience (QoE) requirements. CogStream operates in highly mobile environments, where wearable devices connect to edge nodes for short periods of time, and the handoff between nodes is managed by the system. This requires active connection management, which is handled by the connection manager as shown in Figure 3. Deploying models to the appropriate edge nodes is handled by the orchestration mechanism of the runtime.

4.2.4 Resource Management and Dynamic Provisioning

Many applications require personalized or domain-specific AI models that should be deployed to edge nodes based on the physical context they operate in [22]. For example, an application for a retail scenario may use an object detector that is trained on different data than say, an application that targets museums. Moreover, due to the limited resource of edge nodes, the platform needs to employ efficient resource management techniques to deploy and evict application services based on current demand. To that end, our platform uses advanced container scheduling techniques that make precise placement decisions in edge infrastructure scenarios [24].

5 Prototypes and Experimental Results

This section presents selected aspects of the platform that we have implemented so far, and preliminary feasibility experiments.

5.1 Platform Client SDK

Much of the system's interaction is event-based and asynchronous by nature. Our SDK appropriately enables a reactive programming style. The following example shows how a C# script attached to a Unity object could subscribe to CPOP updates to receive tracking data on cars and cyclists within a 100m radius, and get their positions, bounding boxes, and the object class label. The event handler lambda is called asynchronously when new CPOP data is available.

```
// define CPOP query parameters and request features
var query = new Cpop.Query(["cars", "cyclists"], "100m");
var features = ["bbox", "label"];
Cpop.Subscribe(query, features, (cpopObj) => {
    cpopObj.ObjectId; // tracks the object's identity
    cpopObj.Position; // global position
    cpopObj.Features["label"]; // e.g., "car"
    cpopObj.Features["bbox"];  // the current bounding
    // ... code to react to data update
});
```

Listing 1: Asynchronously reacting to objects tracked by CPOP

5.2 Cloud-Based vs. Edge-Based Offloading

We demonstrate the problems of using cloud-based AI services for interactive systems. Our results corroborate existing findings on cloud vs. edge-based offloading [9, 11, 26]. As a representative

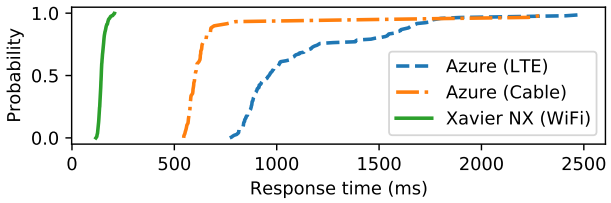

Figure 4: CDF of end-to-end latency when performing object detection over an HTTP service

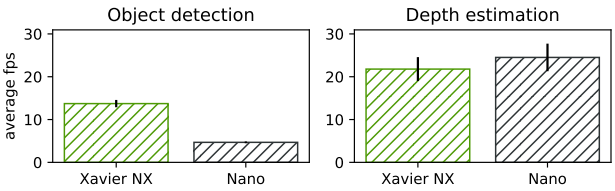

Figure 5: Framerate of our object tracking prototype on Jetson devices

cloud service, we use an object detector of Azure Cognitive Services, deployed in the closest Azure region. The service exposes an HTTP endpoint that receives an image, and returns the detected objects and their bounding boxes. We run the experiment once from a mobile phone using an LTE internet provider, and once from a desktop PC that is connected via fiber to the Internet. We invoke the API 400 times with a 416x416 pixel image, which is the common input size for YOLO. For the edge-based scenario, we use a mobile device that is connected via WiFi to a Jetson Xavier NX that hosts the service. For a fair comparison, we wrap our object detector as an HTTP service, and invoke it in the same way. Figure 4 shows the request response time for the three scenarios. The mean response times are 1116 ms (Azure via LTE), 678 ms (Azure via cable), and 146 ms (Edge device via WiFi). We found that the Xavier can serve up to three instances of the model concurrently without any degradation, demonstrating how a single computer can serve multiple tenants.

5.3 CPOP Prototype and Performance

We have implemented a prototype for the object tracking component of CPOP in Python. The prototype uses a PyTorch implementation of YOLOv5s for object detection, and the depth estimation model of Bian et al. [6] trained on a surveillance camera dataset. Figure 5 shows the performance results of running the models on both an NVIDIA Jetson Xavier NX, and an NVIDIA Jetson Nano. We found that the Xavier device can run both models in parallel without any performance degradation. The results demonstrate the feasibility of running our object tracking method on edge nodes for interactive applications. Currently, the bottleneck is the object detection with 13 FPS. However, a TensorRT C++ implementation and a smaller input size can improve the framerate by up to a factor of 3.

6 Summary and Conclusion

General-purpose wearable AR devices are currently a key technology for cognitive augmentation. To enable interesting applications, devices need access to environmental sensor data, and the ability to perform compute-intensive AI tasks. While AI accelerators for some workloads will likely be provided on the device once a common set of use cases appears, there is still a need to offload tasks to proximate computing resources. Edge computing and smart city infrastructure is a logical step in this technological development. Our platform CognitiveAR enables cognitive assistance applications by abstracting this complex new computing environment, and providing functionality common across applications. Our preliminary experiments show the necessity for edge computing solutions for

interactive applications. Moreover, we demonstrated the feasibility of tracking the position of physical objects using monocular cameras and compact edge computers that can be mounted on, for example, lamp posts in the city. We are working on a full end-to-end implementation of our platform, and using it to implement the example applications that we discussed.

Acknowledgments

This research is funded in part by the Internet Foundation Austria through the Netidee grant program, and by the Austrian Federal Ministry of Science through the Austrian infrastructure program (HRSM 2016) as part of the CPS/IoT Ecosystem project. VRVis is funded by BMK, BMDW, Styria, SFG, Tyrol and Vienna Business Agency in the scope of COMET – Competence Centers for Excellent Technologies (879730) which is managed by FFG.

References

[1] A. Anjomshoaa, F. Duarte, D. Rennings, T. J. Matarazzo, P. deSouza, and C. Ratti. City scanner: Building and scheduling a mobile sensing platform for smart city services. *IEEE Internet of things Journal*, 5(6):4567–4579, 2018.

[2] C. Arth, C. Pirchheim, J. Ventura, D. Schmalstieg, and V. Lepetit. Instant outdoor localization and slam initialization from 2.5 d maps. *IEEE Computer Architecture Letters*, 21(11):1309–1318, 2015.

[3] A. P. Association et al. *Diagnostic and statistical manual of mental disorders (DSM-5®)*. American Psychiatric Pub, 2013.

[4] C. Berenguer, I. Baixauli, S. Gómez, M. d. E. P. Andrés, and S. De Stasio. Exploring the impact of augmented reality in children and adolescents with autism spectrum disorder: A systematic review. *International Journal of Environmental Research and Public Health*, 17(17):6143, 2020.

[5] A. Bernhard. The great bicycle boom of 2020. `https://www.bbc.com/future/bespoke/made-on-earth/the-great-bicycle-boom-of-2020.html`. Accessed: 2021-01.

[6] J.-W. Bian, H. Zhan, N. Wang, T.-J. Chin, C. Shen, and I. Reid. Unsupervised depth learning in challenging indoor video: Weak rectification to rescue. *arXiv preprint arXiv:2006.02708*, 2020.

[7] S. Cass. Taking AI to the edge: Google's TPU now comes in a maker-friendly package. *IEEE Spectrum*, 56(5):16–17, 2019.

[8] C. E. Catlett, P. H. Beckman, R. Sankaran, and K. K. Galvin. Array of things: a scientific research instrument in the public way: platform design and early lessons learned. In *Proceedings of the 2nd international workshop on science of smart city operations and platforms engineering*, pp. 26–33, 2017.

[9] Z. Chen, W. Hu, J. Wang, S. Zhao, B. Amos, G. Wu, K. Ha, K. Elgazzar, P. Pillai, R. Klatzky, et al. An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, pp. 1–14, 2017.

[10] Environmental Tectonics Corporation. Advanced Disaster Management Simulator, 2019. `http://trainingfordisastermanagement.com`, (Accessed: 2019-08-03).

[11] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan. Towards wearable cognitive assistance. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pp. 68–81, 2014.

[12] M. Hu, Y. Chen, G. Zhai, Z. Gao, and L. Fan. An overview of assistive devices for blind and visually impaired people. *International Journal of Robotics and Automation*, 34(5):580–598, 2019.

[13] M. Huber, D. Pustka, P. Keitler, F. Echtler, and G. Klinker. A system architecture for ubiquitous tracking environments. In *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007.

[14] K. Krösl, H. Steinlechner, J. Donabauer, D. Cornel, and J. Waser. Master of disaster. In *The 17th International Conference on Virtual-Reality Continuum and its Applications in Industry*, pp. 1–2, 2019.

[15] C. Kumar B., R. Punitha, and Mohana. Yolov3 and yolov4: Multiple object detection for surveillance applications. In *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 1316–1321, 2020. doi: 10.1109/ICSSIT48917.2020.9214094

[16] Q. Liu and T. Han. Dare: Dynamic adaptive mobile augmented reality with edge computing. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, pp. 1–11. IEEE, 2018.

[17] T. Metzler and K. Shea. Cognitive products: Definition and framework. In *11th International Design Conference DESIGN 2010*, 2010.

[18] NVIDIA. NVIDIA Jetson - the AI platform for autonomous machines. Online. `https://developer.nvidia.com/embedded/develop/hardware`.

[19] H. B. Pasandi and T. Nadeem. CONVINCE: collaborative cross-camera video analytics at the edge. In *2020 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2020, Austin, TX, USA, March 23-27, 2020*, pp. 1–5. IEEE, 2020. doi: 10.1109/PerComWorkshops48775.2020.9156251

[20] G. Premsankar, M. Di Francesco, and T. Taleb. Edge computing for the internet of things: A case study. *IEEE Internet of Things Journal*, 5(2):1275–1284, 2018. doi: 10.1109/JIOT.2018.2805263

[21] T. Rausch and S. Dustdar. Edge intelligence: The convergence of humans, things, and ai. In *2019 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 86–96. IEEE, 2019.

[22] T. Rausch, W. Hummer, V. Muthusamy, A. Rashed, and S. Dustdar. Towards a serverless platform for edge AI. In *2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19)*, 2019.

[23] T. Rausch, S. Nastic, and S. Dustdar. Emma: Distributed qos-aware mqtt middleware for edge computing applications. In *2018 IEEE International Conference on Cloud Engineering (IC2E)*, 2018.

[24] T. Rausch, A. Rashed, and S. Dustdar. Optimized container scheduling for data-intensive serverless edge computing. *Future Generation Computer Systems*, 114:259–271, 2021.

[25] J. Ren, Y. He, G. Huang, G. Yu, Y. Cai, and Z. Zhang. An edge-computing based architecture for mobile augmented reality. *IEEE Network*, 33(4):162–169, 2019.

[26] M. Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[27] M. Satyanarayanan and N. Davies. Augmenting cognition through edge computing. *Computer*, 52(7):37–46, 2019.

[28] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, and B. Amos. Edge analytics in the internet of things. *IEEE Pervasive Computing*, 14(2):24–31, 2015.

[29] R. Schubert, E. Richter, and G. Wanielik. Comparison and evaluation of advanced motion models for vehicle tracking. In *2008 11th international conference on information fusion*, pp. 1–6. IEEE, 2008.

[30] Y. Sermet and I. Demir. Flood action vr: A virtual reality framework for disaster awareness and emergency response training. In *Proceedings of the International Conference on Modeling, Simulation and Visualization Methods*, pp. 65–68. CSREA Press, 2018.

[31] W. Shi and S. Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016.

[32] J. Simonjan. Towards large-scale pervasive smart camera networks. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, 2015.

[33] F. Van Diggelen and P. Enge. The world's first gps mooc and worldwide laboratory using smartphones. In *Proceedings of the 28th international technical meeting of the satellite division of the institute of navigation (ION GNSS+ 2015)*, pp. 361–369, 2015.

[34] J. Wang, Z. Feng, S. George, R. Iyengar, P. Pillai, and M. Satyanarayanan. Towards scalable edge-native applications. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, 2019.

[35] J. Waser, A. Konev, and D. Cornel. On-the-fly decision support in flood management. *GIM International*, issue Nov/Dec:22–25, 2018.

[36] S. White and S. Feiner. SiteLens: Situated visualization techniques for urban site visits. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 1117–1120, 2009.

[37] S. Wu, Y. Fan, S. Zheng, and H. Yang. Object tracking based on orb and temporal-spacial constraint. In *2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI)*, pp. 597–600, 2012. doi: 10.1109/ICACI.2012.6463235

[38] C. Zhao, Q. Sun, C. Zhang, Y. Tang, and F. Qian. Monocular depth estimation based on deep learning: An overview. *Science China Technological Sciences*, 63(9):1612–1627, Jun 2020.