Toward a mixed reality domain model for time-Sensitive applications using IoT infrastructure and edge computing (MRloEF)

Mohamed Elawady¹ · Amany Sarhan¹ · Mahmoud A. M. Alshewimy¹

Accepted: 31 December 2021 / Published online: 24 January 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

Mixed reality (MR) is one of the technologies with many challenges in the design and implementation phases, especially the problems associated with time-sensitive applications. The main objective of this paper is to introduce a conceptual model for MR application that gives MR application a new layer of interactivity by using Internet of things/Internet of everything models, which provide an improved quality of experience for end-users. The model supports the cloud and fog compute layers to give more functionalities that need more processing resources and reduce the latency problems for time-sensitive applications. Validation of the proposed model is performed via demonstrating a prototype of the model applied to a real-time case study and discussing how to enable standard technologies of the various components in the model. Moreover, it shows the applicability of the model, the ease of defining the roles, and the coherence of data or processes found in the most common applications.

Keywords Mixed reality · Internet of things · Cloud computing · Fog computing · Domain model

1 Introduction

In light of Facebook's founder's presentation of the Metaverse revolution, anticipated massive growth in users and applications, primarily virtual reality (VR), augmented reality (AR), and mixed reality (MR) applications, is expected in the next era. As a foundation, these applications will employ cloud, fog, and edge computing technologies, as well as the Internet of things. As a result, an organized model upon which these applications are developed and organized should be supplied before the

✉ Mohamed Elawady
elawady5@gmail.com

¹ Computer and Control Department, Faculty of Engineering, Tanta University, Tanta, Egypt

problem of components and applications conflicting with each other arises. This model should help them arrange their work with each other as well as the physical technologies they use. Now is the moment to provide this concept, before it becomes difficult to restructure these applications in order to organize them, which will cost time and effort.

Furthermore, the COVID-19 pandemic presently impacts almost every country on the planet, necessitating the availability of tools to manage and reduce pandemic consequences. New solutions should be available and inexpensive to the majority of users, and they should be distributed to a wide range of people from various backgrounds. According to this epidemic, more research will focus on innovative techniques of controlling and communicating objects remotely. This field will allow for remote control of the underlying environment via a designed application, eliminating the requirement for the controlling persons to be physically present and minimizing the need on high-touch surfaces.

A better understanding of the impacts of the new situation could help to improve the prevention strategies of the spread of the virus. For this purpose, many organizations and scientific communities are providing grants to enable researchers to produce these new solutions that can be used immediately. Various researches have been directed to several areas affected by the pandemic like economics, healthcare, remote learning, and more. Only a few have been initiated toward the everyday usage of devices and machines found in real life.

Among these research areas, MR is a possible solution to prevent the direct manipulation of the devices under control. Instead, it will replace the current situation with applications that remotely access these devices through an MR application. Among these devices are IoT devices that are widely spread across the globe. Moreover, handling IoT devices currently based on both cloud and fog, where each has its roles and scope of managing processes and data. Both VR and AR have the same direction of remote controlling devices from far distances using a virtual user interface (UI).

Moreover, MR and AR are helpful in places that are hard to be there remotely, and it is mandatory to be physically on the site like markets, hospitals, some factories, etc. Also, MR and VR make people feel their surroundings and not live in a separate reality like VR. In addition, MR can provide an application tracking people and offer a colored overlay on them showing the confidence ratio if they were carrying the virus or not using ML-Cloud-based algorithm.

On the other hand, the Internet of everything (IoE) [1] plays a vital role in many applications [2] and offers promising solutions to evolving many industrial systems [3]. IoE connects people, data, processes, and physical things, making it easier to build applications interacting between real objects (sensors and actuators) and virtual objects in MR. IoT, a subset of IoE, focuses on the communications between physical objects, managing the devices, and most notably, data gathering.

The exponential increase in IoT devices makes cloud services scaling much faster to manage the storage and processing power needed [4]. Still, the real challenge here is the network that clearly appeared when the demand increased on the Internet due to COVID-19 lockdown [5]. As a result, fog computing becomes an essential part of decentralizing the cloud to reduce the amount of data sent on the network and the

latency of the IoT devices' responses. Fog computing is one of many aspects of edge computing which is the extension of cloud computing where services are brought closer to the end devices. Edge computing is developed to address location awareness, mobility support, and high latency issues in delay-sensitive applications. Also, edge computing was introduced to improve the performance of latency-sensitive MR applications in some scientific articles. As a result, MR applications supported by edge computing rely on providing more QoE and supporting more functionalities that need more processing power [6].

Model-based systems typically use some models to describe the architecture and the design of a system in an abstracted manner and the behavior of software items. Building a model for a system can guide the system developers and add an improved comprehension during maintenance, better product quality, and improved reliability. Other benefits could also be realized, such as flexibility, productivity, and interoperability. The system thus can be easily viewed as platform-independent, which facilitates its development using various technologies.

The previous research efforts had produced a variety of methods to build AR, VR, and MR applications based on cloud and fog platforms. However, their work lacks an abstracted and generalized model to cope with changes and further development of these applications. In other words, these applications are produced in an inefficient manner in the absence of an overall model that should be presented to combine a view of the whole development process that may necessitate the modification of the system needs and further the interoperability with other applications of the same or another organization. The main research motivation of our work is to provide an overall model that combines MR, IoE, and fog computing in one comprehensive organized methodology rather than the one-by-one approach of handling these technologies. The main challenge is to incorporate these technologies' parameters, specifications, and criteria in this model to enable the developers and researchers to work correctly with this type of application.

Driven by developments made in MR, IoT, Fog computing infrastructures, the main contribution of this paper is to present a generalized domain model of the MR systems which rely on IoT devices and Fog computing capabilities called MRioEF. The model aimed to organize the design, implementation, and interactions presented between the three technologies in an efficient manner that provided many advantages, including the strength of the design, ease of testing, smooth maintenance of the functions, defining the roles of each process, and coherence of data and processes found in the system. This model should reflect on the execution time and consumed power reduction of the processes of the system, which are the main concerns of the real-time applications. It also gives a better understanding of the possible operations encountered by these components constituting the system. As for our best knowledge, this is the first attempt to provide such an abstraction model for MR systems that utilize IoT devices and fog/cloud computing platforms for general purposes.

The rest of this paper is organized as follows: Sect. 2 presents the background of the three pillars: IoE, MR, and fog computing. Section 3 describes the proposed domain model and its main components in detail. A prototype for evaluating the

proposed MRIOEF domain model is introduced in Sect. 4. Finally, the conclusion is presented in Sect. 5.

2 Background and related work

2.1 Internet of everything

IoE merges people, processes, data, and things [1]. To improve the industry and people's lives, IoE turns the environment events into data, turns information into actions, and turns devices communication into end-application. Things are considered one of the four pillars of IoE built upon the IoT, one of the most emerging technologies in society [7]. In addition to the machine-to-machine (M2M) protocols covered by IoT, IoE includes machine-to-people (M2P) and technology-assisted people-to-people (P2P) interactions, as shown in Fig. 1. Also, IoE has better and prosperous semantic models than IoT, which delivers the correct information to the right person or machine at the right time and makes the design of applications more accessible, faster, and covers more scenarios.

IoE and IoT have many challenges; one of them is the scalability of the number of devices connected to the Internet and the number of functionalities needed for each device [4], which leads to network congestion and more latency in applications. Another challenge is heterogeneity in communication protocols and data models of devices [8], making M2M protocols hard to handle. Privacy and security of collected data from devices and sending it to the cloud are also considered challenges for IoE/IoT.

One solution to these challenges is edge computing, especially fog computing, an extension of cloud computing to make computation and storage away from servers to the edge of the devices' networks; as a result, it is beneficial for latency reduction to support real-time applications. Also, for bandwidth control to act as a buffer between devices and the cloud or even better act as semi-cloud to compute some heavy computation and deliver it to devices and save more resources in the cloud to other computations.

2.2 Fog computing

Cisco coined the term "Fog Computing" in 2012 and has an analogy from real-life fog closer to Earth, but the cloud is in the sky. At the same, fog computing is used whereas closer to end devices, and it is located between end devices and the cloud. It is used for distributed environments, especially IoT networks, to provide more security, privacy, high mobility support, location awareness, high bandwidth, and ultra-low latency [9].

Fog computing created a new layer that allows developers to control their data without sending all of it to the cloud to preserve users' privacy. Also, it can store

Fig. 1 IoE Pillars

some data to be analyzed for application development and operation, besides the control of end devices to send new firmware as updates or manage them in some events. The good architecture for applications that relies on a cloud-fog-device schema can handle scalability in the number and the functionality of end devices using one or multiple layers of fog computers covering segments of required area or used as load balancers for each other. Also, architecture can handle the heterogeneity of end devices. It supports new devices with new models or protocols or even new micro-architecture by using semantic data for devices.

In [10], a survey is introduced on the application layer protocols for communication and how to integrate them in some IoT applications with cloud-fog-end devices architecture and study the main characteristics like network throughput and latency of responses. In [11, 12], frameworks proposed for computational offloading and resource provisioning in edge computing and cloud computing ecosystems open the road for better implementation of MR applications offloading and the criteria of which processes can be offloaded.

2.3 Mixed reality

On the other hand, MR is a promising technology for human interaction with data and has a rich user experience. MR is the merge between real-world and virtual objects illustrated in the same scene to the user. To be clear, a scene in an entirely artificial environment got another name called VR. On the other hand, AR can merge some virtual objects with real-world scenes [13]. The difference between AR and MR is the interactivity between virtual objects and real objects. MR can immerse virtual objects with real objects to make an illusion to the user, render the virtual objects almost like real objects, and interact like them. Unlike AR, which renders virtual objects without caring about these interactions between virtual objects and real objects. Table 1 shows the differences between VR, AR, and MR.

Nowadays, MR applications can be executed on high-end mobile phones or some specific headsets like Google glasses [14] or Microsoft HoloLens [15] as in Fig. 2a because MR needs powerful resources that can be found on the previous devices. They have some hardware accelerators, like neural processing unit (NPU), for processing heavy machine vision algorithms, especially surface mapping algorithms used in these applications.

Many kinds of research focused on recognizing and tracking objects in real-time. [16] focused on the device's rotation by scanning multiple markers (April Tags) on a device that gives the proper meta-data for rendered objects. This method is used in low resources environments and, consequently, can't use machine learning techniques. Moreover, it gives the target objects a bad appearance with the presence of these tags.

In [17], authors designed an application running on 60 frames per second with low resources on AR devices using some techniques to reduce the latency between tracking the object and rendering the final scene, proving the capability of developing a way to track objects and generate MR objects with low latency. Another direction was recognizing the material of the objects and using them in AR applications as in [18], where a system was proposed to reproduce the material appearance of the physical object with the best rotation of the rendered material, which uses AR applications to render animated materials on the physical objects. Authors in [19] also proposed a real-time approach to estimate high-quality material appearance from a single-color image with the illumination-consistent intersection between virtual and real objects, which can be used in MR applications to render more realistic virtual objects.

The current important direction of MR is to improve the interaction between virtual objects and the physical world. Currently, the data flow between the physical and virtual worlds is unidirectional, meaning changes in the physical environment will affect the virtual objects, not vice versa. Yet, with the help of IoT, the whole concept will change since the data flow will change into a bidirectional flow; in other words, any change on the virtual object will reflect on the physical object (device), which will alter the physical environment as a result.

As a result, the quality of experience of applications significantly increased by the merge between IoT and MR, as shown in Fig. 2b, which demonstrates the light intensity control of smart lamps using a virtual slider as an example. When the user starts using the slider, a virtual event is triggered containing a particular value and received by the physical device (in this case, the lamp), performing a specific physical action that changes the light intensity.

This architecture can be improved further by using IoE models and specific M2P and P2P models. The M2P model interprets any event done by a person, triggering multiple processes either preconfigured or predicted by a certain algorithm using machine learning. On the other hand, the P2P model focuses on the interaction between two or more users with a mutual virtual object sharing the same scene but not necessarily the same angle of view. Virtual TV can be considered an excellent example to explain the P2P relation. Many users can interact with the virtual TV appearing in the same scene, but none can necessarily watch the virtual TV from the same angle.

Table 1 Differences between VR, AR, and MR

	Virtual Reality			Augmented Reality		Mixed Reality	
Environment description	An entirely generated scene (image and sound) by computer using some headset to immerse in this virtual world and make a feeling the user transport in this world			A merge between the real-world objects and computer graphics objects in the same scene using headsets or mobile application And make a feel that the real world has new elements			
Awareness	Rendering the virtual world makes it hard to distinguish between the real world and the virtual world			Rendered objects can be identified; it's like a floating UI in real world		Rendered objects hard to be identified from real objects	
Interactivity	It's users and virtual world interaction only. There is no interactivity between the virtual world and the real world			The primary interaction between users and virtual objects, but it may have some inter-action between the real-world and virtual objects		It is mainly focusing on blending the interactivity between virtual objects, the real world, and user actions	
Remote collaboration	Remote users can interact with applications using avatar representations					It can be achieved using avatar representations. It also adopts remote interaction with the real world like remote surgery

Fig. 2 Example of the merge between IoT and MR

2.4 Related work

The obstacles for real-time MR applications are summarized in prior studies such as [20], which include high latency responses and the need for more network bandwidth to communicate, which is the key difficulty. Other difficulties highlighted include reuse of prior computations, mobility, the discovery of computation resources and service availability, and communication security.

Therefore, fog computing and IoT edge can host time-sensitive MR services to speed up the system response. It can also be used as a storage area for some pre-rendered virtual objects. As shown in Fig. 3, fog computing serves two categories of devices: IoT and MR devices. It should be mentioned that fog computing not only helps MRIOE applications alone but can be used for MR applications that use virtual objects without physical interactions.

This new direction will open new opportunities for High QoE in MR applications and change the style of the user interface from a graphical user interface (GUI) to an entirely natural user interface (NUI) [21]. The market and industry focus on building smart devices containing more advanced features, so it's normal to use a futuristic user interface with rich and interactive information with users; it can be used on any IoT device to add more value. To summarize the current contributions related to our work, Table 2 is given.

Also, some research can complement ours, like [22], which utilizes edge computing to improve manufacturing systems by proposing a systematic solution for Industrial IoT (IIoT) data-driven solutions; it's focused on three domains: manufacturing, data analytics, and edge computing. Moreover, MR is mentioned as the key for the interface design phase to raise awareness and enhance accessibility.

2.5 Problem formulation and plan of solution

The conclusion drawn from the study of the previous work [22, 24–27] is that they have focused only on one or two of the main three pillars of the work considered in this paper (IoE/IoT, AR, fog/cloud computing), ignoring the third one with its benefits to improve the overall performance of the applications. In [23, 28], models for MR applications were built in real-time. Still, they handled the application as a case-by-case

Fig. 3 MRioE and MR supported by fog computing

development without further consideration of in-depth possible future development of similar applications. There was no discussion on how to couple between MR, IoT devices, and fog/cloud computing platforms for any similar applications.

As a result, this motivated us to develop a general domain model that can help integrate all the three pillars (i.e., MR, IoT devices, and fog/cloud), illustrate the benefits of their integration, and clarify how each one has its advantages and how they complement each other. This paper also presents the domain model as a concrete layered model that considers the different issues on each layer and their interactions. In addition, the model ensures that the layers are designed in a decoupled manner to give the applications more interoperability in the future and the extensibility for new futuristic technologies.

3 The proposed MRioEF domain model

This paper defines a domain model of MR coupled with the IoE Infrastructure using fog computing to build MR applications MRioEF. The IoT domain concepts and MR domain concepts are analyzed and identified to correlate them and further engage them into the Fog domain concepts as one model. This model will enable the interaction with IoT devices through fog computing and provide the applicability of real-time MR applications with low latency between MR actions and IoT events.

The central architecture of the proposed MR domain model is based on IoE and fog/cloud computing platforms. As shown in Fig. 4, the model consists of three main layers: IoT, MR, and fog/cloud. The MRioEF domain model is designed to be as generic as possible by developing it as a methodology-independent model to facilitate its reusability in different environments. MRioEF main layers are described in brief before going into the details of each of them:

- 1- **The IoT layer** includes the components representing both the physical hardware and resources of the IoT device.

Table 2 Summary of Recent AR/MR work based on IoT, Fog, and Cloud

Paper	AR/MR	IoT	Fog	Cloud	Contribution
Fernández et al. [23], 2018	AR	✓	✓	✓	Describe Navantia's Industrial AR (IAR) architecture in shipyards which uses cloudlets and fog computing to reduce latency
					It was compared with the traditional cloud-based system, and the cloudlets and fog computing were also compared when sending different sizes of payload and mentioned that there is an Industrial IoT framework (IIoT) that can be integrated with Industrial AR (IAR) at the same level of cloudlets or fog computers
Phupattanasilp et al. [24], 2019	AR	✓	X	X	Design an AR application using AR-UI to visualize the data generated from IoT devices
Zietsch et al. [22], 2019	MR	✓	✓	X	Propose a framework to add MR visualization to IIoT and edge computing infrastructure to improve manufacturing by increasing workers reaction speed
Blanco-Novoa et al. [25], 2020	AR/MR	✓	X	X	Design a framework to reduce difficulties to interconnecting of heterogeneous IoT devices and AR/MR devices
S. Bucsai et al. [26], 2020	MR	✓	X	X	Build an application binding IoT data with MR models using Unity and Android AR-Core/Apple AR-Kit
A. Morris et al. [27], 2020	MR	✓	X	X	Design a framework for MR with IoT and add a bidirectional interaction between IoT virtual avatars and IoT devices
Vidal-Balea et al. [28], 2020	AR/MR	✓	✓	✓	Build an AR/MR application for teaching system based on Microsoft HoloLens [15], edge computing devices, and IoT devices

- 2- **The MR layer** includes the components representing the MR components, its resources, the scene, and the object related to the MR entity.
- 3- **The fog/Cloud Computing layer** includes the fog computing components as the various profiles of the user, services, processes, and data entered the system through IoT devices and the connection to the cloud.

The relationships between the layers of those main layers are shown in Fig. 4, which complete the connection concepts between the components and indicate how they could interact with each other in an organized way. A human interacts with the system through the MR device, not with the IoT device directly, which will increase the system's reliability and provide protection of the IoT device. IoT devices reveal their capabilities to the human user via the interaction designed in the MR application. According to the developed application, connections to the fog component are performed either by the IoT virtual entity or MR entity. Also, the fog component is responsible for connecting to the cloud if needed.

3.1 IoT layer

Figure 5 shows the components of the IoT layer of the proposed MRioEF domain model, which represents all the aspects of the IoT device found in the system. It is organized into four components: IoT device, virtual entity, physical resources, and virtual resources.

Fig. 4 MRioEF Domain Model Layers

3.1.1 IoT device component

Some physical objects may contain an embedded system and are called smart devices in the industry. If this embedded system architecture follows an IoT schema, it is the first component of IoT components and is described as an “IoT Device.” It represents the physical part of the IoT device, which is the central part of the whole layer. The IoT device may vary according to the application at hand, especially when working with MR applications. Unlike previous work, we have separated the IoT device’s physical parts from the processes attached to it, as we have assigned it to the fog components. As shown later in the model layers discussion, this separation yields better control of the processes.

3.1.2 Physical resources component

This component represents the IoT device’s physical resources, including network & location resources, sensor, actuator, storage, and process node. Some IoT devices have a simple data model and have constraints on bandwidth, memory, processing, and power. While other IoT devices, the big things, have complex data models and may include smaller IoT devices inside them. Examples of types of IoT devices are shown in Fig. 6. In this sense, the IoT device component is represented as a complex entity that may contain other IoT devices for more abstraction and simplicity in the design.

Each IoT device has main attributes that describe it semantically, like serial number, manufacture, uniform resource identifier (URI), firmware version, application programming interface (API) hyperlink, etc. The most critical attribute would be the “Physical Resource” described in the domain model as a separate component. The physical resource component uses the inheritance concept in object-oriented programming (OOP). Inheritance in OOP allows children classes to inherit attributes from parent classes; similarly, physical resource components can be considered as parent classes, and all the other components inherit attributes from it. “Sensors” and “Actuators” are considered the essential resource components as they represent the input/output (IO) in embedded systems.

Fig. 5 IoT layer of the MRioEF domain model

Another important resource in IoT devices is network resources that may support TCP/IP stack or not. Regardless of its type, it must contain all the attributes that describe supported protocols in any layer of the OSI network model and the capability of bridging between two different protocols.

The IoT devices may contain other types of resources it wants to expose, like “processing” and “storage” resources to provide extra resources to the system in some situations. For example, the storage of IoT devices can be used as a backup in any failure in the network. The processing power of the IoT device can be used as an extra processing resource for some remote execution for some applications as a service. For example, a heavy processing application needs to be executed on a smartphone; the smartphone will use the extra processing resources exposed by nearby devices in a distributed manner.

3.1.3 “Virtual entity” component and “virtual resources” component

Each IoT device and its resources components have “Virtual Entity” and “Virtual Resources,” respectively. These components, having the information of the physical entities, expose the functionalities of the IoT device and the resources as an API that can easily connect. In addition to that, the API description identifies its input and output semantically and the supported technologies by this API like SOAP, REST, GraphQL, OData, etc.

The separation of the IoT device component and its virtual entity, also, the separation of a resource component and its virtual resource results from using the IoE schema [29]. And this separation facilitates adding more application coverage and gives an abstraction level in the proposed solution model. Some applications can use an entirely virtual device as if it was a real one. For example, a virtual TV, which has a lot of functions in its APIs, can be used as an actual TV with a subscription on cable channels or any other subscription application, as shown in Fig. 7.

Fig. 6 Categories of IoT devices based on data model complexity

3.2 MR layer

This layer represents the MR concepts, including the scene, the object(s), the entity, the marker, the recognizer, and the renderer (Fig. 8). The MR entity represents the MR device it points to. Each MR device has a component called the “MR Entity” component used to register and manage two main components of mixed reality, which are: “Recognizer/Tracker” and “Renderer.” These are the input and output of the MR device.

Like any device that contains an embedded system with network resources and can be classified as an IoT device, the mixed reality device is an IoT device, and its resources could be used in IoT networks. However, for more clarity, simplicity, and focusing on the proposed domain’s target, this connection is not shown explicitly.

The MR entity has an operating system (OS) that manages its resources and connects with the fog or cloud layer to serve the MR applications installed on this OS. It could connect with IoT devices directly when satisfying two conditions: They both have the same communication protocol. The IoT device is categorized as a complex model because the complex IoT model can handle more than one connection asynchronously.

The MR applications that depend on the IoT/IoE devices (called MRIoE applications) are typically handled as processes on the operating system of the MR device. However, some of these applications can follow distributed processing architecture supported by our proposed model. The application processing is handled through this architecture as two main processes; one is executed on the MR entity while the other is executed on the fog. These processes are communicated through remote procedure call (RPC) protocols.

The “MR process” component is responsible for mapping the real-world objects into the proper virtual object, and it is executed into four main steps. Figure 9 shows these steps applied to an example which calls four different MRIoE applications in one scene: a light bulb control [in blue shade], a toaster control [in yellow shade], a smart water faucet control [in green shade], and a coffee machine control [in red

Fig. 7 A virtual TV having a subscription to Netflix

Fig. 8 MR layer of the MRioEF domain model

shade]. In this example, the MR scene adds four different virtual objects, where each object is generated from various applications selected and executed by the MR entity's OS. The MR scene is ready for user gestures to trigger any action on IoT devices like toggling lights or preparing some coffee. The main steps of the MR process are performed as follows:

3.2.1 Step 1: Markers recognition

This step is responsible for recognizing the current scene and determining which application is meant to be opened and which object(s) inside this application to be rendered. This step is handled using the “Recognizer/Tracker” component by recognizing the “Markers” components in this scene.

A “Marker” in MR could be one of the physical scene parts or an external part used to trigger some visual actions. In other words, it could be a frame, image, QR code, location, face, object, or another future marker. To initiate any recognizer installed on the MR entity and generate actions that affect the final MR scene. A different “Marker” component recognizes objects like shape, material, tags, and location. The word “Marker” is known in object and surface recognizers. Still, different markers can be merged with other recognizers like a material marker and gradient color marker merged with a face recognizer in an application of virtual makeup from Dior [30], as shown in Fig. 10.

There are many categories of “Recognizer/Tracker,” such as object recognizer and face recognizer, which can execute sub-algorithms using the “Marker” recognizer components. This separation between recognizers and markers helps to improve the algorithms used separately and adds modularity to recognizers and markers that act as background services in the MR device's OS. Another category of recognizers is surface recognizers, which accurately detect surfaces like floor, wall, and other flat surfaces. It also gives the depth of each physical object and the scene's lighting conditions, which leads to a more accurate intersection between virtual objects and real objects. The last category of recognizers supported by our framework is gesture

Fig. 9 Steps of MR process with an example

recognizer which enables the virtual objects to interact with the user in front of the MR device (unlike tapping on a touch screen in the AR applications).

3.2.2 Step 2: Extract MR meta-objects

We want to discuss the “Recognizer/Tracker” component in more detail as it is critical to MR applications. The “Recognizer/Tracker” component extracts the markers from the scenes as meta-objects that describe the coordinates of the objects in the scene and the output properties of each marker, if found, to detect which MR/MRiOE application is the target, as shown in Algorithm (1). The *MRMetaObject* contains the coordinates of the object and a list of registered attributes extracted by each marker recognizer. It should be noted that a variable d was used to solve the slight difference in output coordinates of each recognizer which may generate duplication of the same object in the following stages.

Algorithm (1): Extract MR meta-objects

```

Get scene object
Parallel execute to marker recognizers
 $L \leftarrow$  empty list of MRMetaObject
Foreach marker as  $m$ :
    Foreach object in  $m$  as  $O$ :
        If  $L$  has item that has coordinates near  $O$  by  $d$  pixels as
             $\sqrt{(O.x - \text{item}.x)^2 + (O.y - \text{item}.y)^2} \leq d$ 
            Then:  $C \leftarrow L[\text{item}]$ 
            Else:  $C \leftarrow$  new item in  $L$  with coordinates of  $O$ 
        Add properties of  $O$  to the list of markers in  $C$ 
Return  $L$ 
  
```

Fig. 10 Virtual Makeup merging material and gradient color markers with a face recognizer

3.2.3 Step 3: Executing MR/MRIoE application(s)

After determining the marker found in the scene, the targeted MR/MRIoE application(s) is executed. These applications are installed on the “MR Entity” component and accomplished by filtering all access points of all installed applications with the list of *MRMetaObject* returned from the “Recognizer/ Tracker” component.

It is possible to run multiple applications at the same time on the same scene. Each application executes its logic and generates a *3D-MRObj* that is ready to be rendered; however, some activities in the MR applications may need the previous state of the *3D-MRObj* generated before and sent inside the *MRMetaObject* by the “Recognizer/ Tracker” component. The *3D-MRObj*s are represented as XML or JSON files with other extra files like texture files sent to the MR process’s next step, the “Renderer” component, to finalize the rendering of generated objects into “MR Scene.”

Like any modern OS running multiple applications, each application has permissions to access the resources and APIs provided by this OS. The “MR Entity” provides the essential resource in the framework, which is the communication with the fog that gives a bunch of new APIs to MR/MRIoE applications which will be described in detail in the next section.

3.2.4 Step 4: Rendering the objects

The “Renderer” component includes two stages, as shown in Fig. 11, which are used to render the 3D-MR objects and merge them with the physical object to polish the MR scene and make the rendered objects look like real ones. The first stage is getting 3D-MR objects generated by the MR/MRIoE applications and adding attributes to the objects based on the model profile to change the colors, contrast, color saturation, texture filters, general graphics settings, and animation settings. According to the current user identified, all model profiles are stored on the fog, and the renderer uses only the selected profile based on this user. The “Renderer” starts rendering the objects from the different applications in one virtual scene after preparing all the data needed.

The second stage is merging the virtual objects with the real world using the surface recognizer’s meta-data. It also hides from the virtual objects the parts intersected with the physical objects, giving the user the best illusion and making it hard to discriminate between virtual and physical objects. Finally, it adds the virtual objects’ proper rotation, lighting, and shadows according to the attributes extracted by the surface recognizer and lighting estimation algorithms executed.

3.3 Fog/cloud computing layer

The Fog/Cloud computing components are responsible for executing the MR and IoT processes related to the application and storing the various user profiles. The Fog components are responsible for handling all the communications with the cloud

Fig. 11 Stages of the renderer component

and sending the proper data reports to be held on the cloud. Figure 12 shows the fog/cloud layer components in the domain model. It comprises the following components: the profile, service, process, data report, raw data entity, and cloud connector.

Fig. 12 Fog/Cloud layer of the MRioEF domain model

3.3.1 Privacy & security profile component

This component represents the various user profiles. Each user using the MR devices has a profile stored on the fog storage area. The user profile is organized into three different profiles according to the information and functions intended by these profiles: Models Profile, Performance Profile, and Privacy & Security Profile. A separate component is developed for each profile as follows:

- The “Models Profile” component helps the “Renderer” components to get color schema, text language, graphic settings, and other settings that can change the models during rendering.
- The “Performance Profile” component helps the “Renderer” cache the first stage of rendering to reduce the overall time of the rendering process. It also helps the “Recognizer/ Tracker” component by using the powerful processing resources in the fog instead of the MR device and allows the recognizers to perform faster by reducing the number of targets that can be recognized using a real-time location updater.
- The “Privacy & Security Profile” component filters all the data in/out from the fog with user restrictions using the meta-data of each record. It also has the encryption keys, the authentication database, and the authorization schema.

Other profiles can be added to this component. Each user profile data affects the attributes of the Fog components and the MR components depending on the current user preferences. This profile, however, can be synchronized with the cloud to be imported on other authorized fog computers.

3.3.2 Raw data entity component

Each “Virtual Entity” component in the IoT layer has a driver on the fog that communicates with the virtual entity using the proper protocols that fit the network resources of the IoT device and with the current communication channel properties between the IoT device and the fog. The driver can be downloaded automatically from the cloud to the fog after reading the properties of the “Virtual Entity” component. This driver adds a container, called the “Raw Data Entity” component, to collect all data from/goes to the IoT virtual entity. It also acts as a translator between the “Raw Data Entity” and the “Virtual Entity” components using a powerful extension for programming languages called the reactive extension (Rx) [31–33].

The “Raw Data Entity” component has a template in the fog, based on the type of the IoT “Virtual Entity” component, to bind it with the suitable “IoT Process” component, which adds a variety of functionalities based on the features supported by the “Virtual Entity” component. The driver of the virtual entity can bring new features not supported by the “IoT Process” component, and therefore, the IoT process exposes it as RPC.

3.3.3 Data report component

The “Data Report” component generates reports using filters of the “Privacy & Security Profile” component. These reports can be sent to the cloud for synchronization between the fog and the cloud. It also generates logs for system tracking, and it can be used to re-train some algorithms for system improvement. Finally, the “Cloud Connector” component is used to connect different cloud vendors for different usages. It saves the credentials of each cloud and the relation between each application and each registered cloud.

3.3.4 MR processes distributed computation

As mentioned before, the MR application can follow a distributed architecture that distributes the execution into two parts; one on the “MR Entity” and the other on the fog as an “MR Process” component connected using RPC. The MR process is downloaded from the cloud to the fog storage the first time the MR entity requests it to the fog; otherwise, the MR process is loaded from the local storage of the fog if the exact timestamp and version of the requested process are stored on the fog.

The applications following this architecture are designed for scenarios when the resources of the MR entity are insufficient or may consume the power of the MR device faster than usual. In that manner, the light part of the process will be executed on the MR entity while the heavy part executed on the fog. For example, a rich MR simulation application performed on an MR entity that follows this architecture could be divided. The simulation algorithms are executed on the fog while the other parts are executed on the MR entity because simulation consumes a lot of processing power and memory on the MR entity. These parts running on the MR entity could focus on rendering the simulated objects produced from the fog execution of the simulation algorithms when requested by RPC with targeted parameters.

The “MR Process” components pose the greatest danger on the fog if it has malicious code, so each process should run in a sandbox to avoid memory corruption or other execution attacks. It also should be given limited resources via permissions requested by the application and filtered by the cloud and the “Privacy & Security profile” component. The process manager in the fog component has an API that checks the request’s authorization which can spawn or terminate any process, whether it is an IoT process or an MR process. Each process can communicate with other processes using Inter-process communication (IPC) like named pipes or message queuing [34].

3.3.5 MR communicating IoT through fog components

Another possible architecture of the fog components is when the MR application executes totally on the “MR Entity” component; however, it requests the data from the fog without executing the “MR Process” on the fog. This architecture will present some IoT processes as a service. The fog computer has a service broker to ease the real-time data flow between the IoT device and the MR device to cover such architecture.

Each IoT service publishes its data on a topic that allows subscribers to follow multiple services simultaneously. For example, as shown in Fig. 13, if an MR application called “app1” controls one model of smart bulbs, it could be on the topic “*light.model1.**”. Another application called “app2” that controls all smart bulbs could be on the topic “*light.*.**”; while to control a bulb without knowing its model could be on the topic “*light.*.ID*”. Also, the fog computer has a service orchestrator to compose multiple services as one service to add more functions supported by fog and save the processing power of MR devices.

3.4 Model layers integration

Figure 14 shows the overall model layers integrated as a single model. The interaction between the layers is shown as directed edges between them. It should be mentioned that red components describe the physical elements, “human” who uses “MR device” and directing the device to see a “physical object” in reality which is called in the domain model “Physical scene.” Each physical scene the human saw through the MR device will be processed to extract the physical objects that have an application on the MR device’s OS, whether it is an MR application or an MRIoE application. For MR applications, the physical object will be recognized using marker-less recognizers or marker recognizers. Then, the proper MR object is rendered in “MR scene,” which is associated with the physical object. While in MRIoEF applications, the connection with the IoT device is added inside this object which means adding the targeted functionalities of the system. The following sections will explain the three main layers of the domain model in more detail.

3.5 Functional model

Figure 15 summarizes the functional model of the proposed domain model and how the data flows from a layer to another. The functional model adds a layer not mentioned before in the domain model, “Network hub.” The fog computer can be used

Fig. 13 Example of publisher–subscriber architecture in fog computer

Fig. 14 The Proposed MRioEF Domain model

Fig. 15 Functional Model of the MRioEF Domain Model

as a bridge between the different communication protocols. For example, a car, classified as a complex IoT device, can be a host for itself without the need for fog. Still, it has a LoRa connection only; to connect it with a Wi-Fi MR device, we must use the fog computer as a Wi-Fi–LoRa bridge.

The “Network Hub” also has a location updater to update the location of each device connected with the fog using indoor positioning system helpers or using a reference location. The work in [35] uses a relational localization between MR and IoT devices using 3D coordinates and rotation of devices.

3.6 Extensibility

Many different applications can benefit from the proposed MRIoEF domain model or at least some of its various components that cover the application requirements. But the target is not to use some components and leave some; on the contrary, the target is to use all the components to give the applications more interoperability in the future and the extensibility for new futuristic technologies. Some components are inspired by [36], and IoE models are the main secret behind implementing this framework due to the powerful abstraction of everything and their interaction.

4 Model implementation and evaluation

The MRIoEF domain model is designed to be generic; by developing it as a methodology-independent model to facilitate its reusability in different environments. To illustrate the value of the proposed domain model, we have built a real-time prototype using this model called “My Hives.” The objective is to implement the model and its components in a real environment and describe their interaction. The application is a part of the R&D department in GaMP [37] to test the MRIoEF domain model and ensure the capability of using the model in future projects. Another objective of this project is to develop the internal GaMP services to follow this domain model for fast delivery and ease of development of IoT projects that require fog computing. This project is also used to ensure the applicability to extend the applications with MR technologies quickly and smoothly when this technology becomes a part of our life in the near future.

4.1 Prototype description

The system used as a prototype of the proposed model in this paper is a monitoring system for a beehive as part of the company’s current project called “My Hives” [38]. This project aims to prevent any problems causing bees’ death and reduce the probabilities of bees’ extinction. This project is supported by the Financial Instrument for the Environment (LIFE) program by the European Union (EU) [39]. Figure 16 shows the system overview of the investigated application, which collects the necessary data from hives using custom-designed IoT devices. This data is sent using LoRa protocol to a fog computer. To prove that the fog computer can serve more than one application, we have added another application that helps an old glasshouse system. This second application uses a programmable logic controller (PLC) connected to the fog computer using a 2.4 GHz 802.11n Wi-Fi access point. The MR device is connected to the fog computer using 5 GHz 802.11ac Wi-Fi. The fog computer is connected to the Internet with Ethernet as a primary connection and LTE as a backup connection.

Fig. 16 “My Hives” system overview

4.2 Hardware components

The hardware components used in the case study are composed of various IoT devices, the fog computer, and the MR device with their interconnections and connections to the external world. The IoT layer uses the LoPy4 board [40] by Pycom [41], connected to many hardware devices like temperature and humidity sensors, weight sensors, small microphones, GPS, Gyroscope, and accelerometer. One advantage of Pycom boards is that the primary development programming language is micro-python, which supports different programming paradigms. Moreover, it has rich built-in libraries like experimental thread library, various data structures, and networking libraries.

The fog layer consists of two parts, as shown in Fig. 17. The first part uses the FiPy board [42] by Pycom as the network hub of the fog computer because it supports five types of networks: Wi-Fi, Bluetooth, LoRa, Sigfox, and LTE-M. The FiPy board is connected using SPI protocol with Jetson Nano Developer Kit [43] by Nvidia [44], which has Quad-core ARM A57, 4 GB DDR4 RAM, and 128 CUDA cores used for heavy processing ML algorithms and other layers discussed in the fog components. A gigabit Ethernet interface is the primary interface for Internet connection.

Today’s MR experience requires an expensive headset with the needed sensors and cameras like Microsoft HoloLens, as shown in (Fig. 2a) and Magic Leap [45]. We have used a custom MR device built as a prototype to test the domain model to avoid such a cost. This device has three main parts, as shown in Fig. 18. The first part is a laptop 2-in-1 Dell XPS 13 inch with CPU Intel Core i7-1065G7 Quad-core 1.3 GHz, 16 GB DDR4, and 512 GB PCIe-NVMe×4 SSD. The second part is a Razer Core X external GPU platform with GPU Nvidia RTX 2060 Super, connected

Fig. 17 Hardware used in the case study—Fog Layer

Fig. 18 Hardware used in the case study—MR layer

with the laptop using Thunderbolt 3. The third part is a Microsoft Kinect for Windows connected to the laptop using USB 3.

4.3 Domain model implementation

To demonstrate the domain model components implemented in this project, it should be mentioned that Figs. (19, 20, and 21) are color-coded with the same color used in the domain model to show how to map a generic domain model to a real implementation with different technologies used for each layer.

It is better to start with the IoT layer to illustrate how IoT devices interact with the framework. In this project, we have two different IoT devices; the main device is a new device (supporting the IoE scheme in the framework) which is the beehive node, while the other one is an old IoT device that follows its own scheme but has an application on the fog that handles the integration with the framework. In this manner, we will focus on implementing the first device only.

The “Virtual Entity” component, representing one of the beehive nodes, is described as a JSON response where all functionalities and virtual resources are exposed as a REST API. To test the request on the device without a gateway, we can use the Wi-Fi and HTTP server on the device, and the request will be like

Fig. 19 IoT node abstraction

Fig. 20 Fog computer abstraction for My Hives application

“http://<IP-Address>/Hive/” or “http://<IP-Address>/IoT/”. When the device is requested on the endpoint “Hive,” the response has readings of each “Virtual Resource” on the device. Each connected sensor and network resource is exposed as a “Virtual Resource” and has a universally unique identifier (UUID) to help the fog and the cloud register new resources dynamically in data logs.

Figure 19 shows how the IoT devices follow the framework by building the resource manager to translate physical resources to virtual resources. Also, it shows the network manager for additional functions needed to communicate with other devices. On top of them, embedded OS, which reads, writes, controls them by the endpoints controlling the virtual entity like REST API, LoRa broadcast API, and Bluetooth serial commands for initial configuration by the user.

Fig. 21 MR device abstraction

The fog computer collects data from IoT devices and stores the data for an entire month in a time-series database (the “Raw Data Entity” in the framework), giving less latency to the most frequent queries requested from end-users. After that, data is distributed to three processes: one predicts the behavior of bees by their sound using an ML algorithm, one prepares a data report that will be sent to the cloud using the cloud connector, and one handles other applications’ logic. All data sent to the cloud are afterward filtered by the user privacy profile, which blocks the location of IoT devices by default. The MR devices connect with the fog computer using a gRPC service, which connects to other services internally. Figure 20 shows the fog computer abstraction to show how it follows the framework.

This application currently has a single user for which it collects data from many IoT devices it owns. In other words, it’s a single application serving a single user on multiple devices, which is one type of many applications that can be performed by the fog computer, not the only one. However, as proof that the fog can serve multiple users, we have added another beehive that belongs to another user and traced how the fog computer behaves on multiple-user applications. We have also implemented another application that controls and monitors a glasshouse to check for multiple applications on one fog. We have also added a simple application that follows the distributed architecture of dividing the application to MR entity—fog layers as mentioned in Sect. 3.3.4, playing some virtual butterflies and saving the last position for each butterfly.

The OS used on our custom MR device is Microsoft Windows 10×64. On top, we use Microsoft Kinect SDK 2.0 and Interactive Virtual Object Engine (IVO) by Zugara [46], which is a gesture recognizer used in AR/VR environments to interact with 3D virtual objects using Microsoft Kinect, also we used a facial recognizer built with OpenBR [47] and object recognizer built with Detectron2 [48] by Facebook. Each recognizer is called and handled by a separate thread using the .Net core framework by C# programming language and Task Parallel Library (TPL) to extract MR Meta-objects in a concurrent list. The app selector is used to call the proper application with suitable parameters. The MR application uses other components

such as the network manager to communicate with the fog computer, the profile manager to get authorized resources from the fog, and the application settings. It may be using the MR model cache for more performance and less latency. Figure 21 shows the MR device construction.

4.4 MyHives System Results

In this section, we will describe an important feature of MyHives system, which is identifying a beehive using a QR code, as shown in Fig. 22. Using a QR code to identify an object is the fastest way to identify similar objects as each object has a different QR code. Each beehive has its own IoT device that will be used to monitor it. The beehive will be our test subject in this scene, and the IoT device will provide us with some data about the status of the beehive as the data shows a different MR response each time. Figure 23 shows the test beehive in different states with simulated data from its IoT device.

The experiment scenario consists in changing the data in our test subject using some python scripts on its IoT device to simulate different states in a specific time—range between 5 and 20 s—then measure the end-to-end (E2E) latency when the data in the IoT device is sent, and the MR device renders the scene; this is called the time to virtual element (T2VE), as is described in [49, 50]. The performance evaluation compares three different configurations that can use this scenario. The first configuration uses Microsoft Azure cloud instance, and the second configuration uses a fog computer to transmit IoT devices data to MR devices only, while the

Fig. 22 MyHives MR application test scene

Fig. 23 Different states of the test subject in MyHives MR application

third configuration uses a fog computer like the second one but in addition to MR scene rendering profiles shown in Sect. 3.2.

Every experiment was tested for two hours on the same configuration in order to avoid any network congestion. Figure 24 shows the T2VE in the test scene with three different configurations: using cloud only, using cloud and fog, and using fog for rendering the MR object. From the figure, we can see that using the third configuration, which performs the MR object rendering on the fog instead of the cloud, gives the minimum latency value which is less than the other two configurations. It also solves the jitter problem produced when using the cloud only, which is shown as the sparks in the latency values, explained as the fetching time for the correct

Fig. 24 T2VE for the test scene

object in the background every time we change the state of the object. These sparks totally disappear when using the fog to help MR for rendering and caching.

Another experiment scenario targets measuring the performance metrics when many MR objects are available in the scene. We use the script on another 25 beehives simultaneously and a 4G router used to monitor the total bandwidth required by devices. Figure 25 shows that each virtual object will take its own share from the bandwidth, but when using the rendering profiles, the bandwidth is reduced and, at some point, changed to constant because the heavy part of the scene was rendered in the fog. This part directly affects the processing usage in the MR device, and the frame rate dropping in the scene does not exist due to the CPU bottleneck solved.

It should be mentioned that the framework provides a better abstraction for each layer and decoupling the code for more business logic was easy, also the ease of changing any component with different technologies. We have also performed a quick survey on number of volunteer users that are required to use the model to build their application and report their experience with the model. One test user said: “It’s nice to provide tools control which data I want to send to the cloud,” this was a benefit of using fog computer and not losing any QoE in the application. The problems faced in development are the debugging, testing, benchmarking tools that are still not mature enough with the framework and need some improvements, especially for the applications using test-driven development (TDD) approaches.

Fig. 25 Performance metrics in the second test scenario

5 Conclusion

This paper has introduced a general domain model capable of dealing with the design of various applications based on IoT, MR, and fog computing technologies and gaining the benefits of each one. The domain model can cover many different IoE applications, even with physical devices (IoT devices) or with virtual ones. It also supports changing the recognition algorithms used at the MR level and using them as a component for more generic respect. Finally, the fog layer is used as a host for heavy resource-intensive applications and preserves user data privacy. Based on the real-time experiment that applied this domain model, the fast delivery of system responses to user actions. It also showed the ease of development for the IoT-based applications (as stated by the application developers) requiring fog computing. The project also clearly illustrates the capability to extend the applications with MR technologies in a short time and smooth way.

Many different adaptations, tests, and experiments have been left for the future due to the diversity of applications covered by this framework. Also, the framework still needs better tools for component tests, integration tests, and benchmarking.

Declarations

Conflict of interest The authors declare the following financial interests/personal relationships which may be considered as potential competing interests.

References

1. Miraz MH, Ali M, Excell PS, Picking R (2015) A review on internet of things (IoT), internet of everything (IoE) and internet of nano things (IoNT). *Int Technol Appl (ITA)* 2015:219–224. <https://doi.org/10.1109/ITechA.2015.7317398>
2. Langley DJ, van Doorn J, Ng ICL, Stieglitz S, Lazovik A, Boonstra A (2021) The internet of everything: smart things and their impact on business models. *J Bus Res* 122:853–863. <https://doi.org/10.1016/j.jbusres.2019.12.035>
3. Di Martino B, Li KC, Yang LT, Esposito A (2018) Trends and Strategic Researches in Internet of Everything. In: Di Martino B, Li KC, Yang L, Esposito A (eds) *Internet of Everything. Internet of Things (Technology, Communications, and Computing)*, Springer, Singapore. https://doi.org/10.1007/978-981-10-5861-5_1
4. Čolaković A, Hadžialić M (2018) Internet of things (IoT): a review of enabling technologies, challenges, and open research issues. *Comp Net.* 144:17–39. <https://doi.org/10.1016/j.comnet.2018.07.017>
5. A. Feldmann et al. (2021) Implications of the COVID-19 Pandemic on the Internet Traffic, Broadband Coverage in Germany; 15th ITG-Symposium, p. 1–5.
6. Khan WZ, Ahmed E, Hakak S, Yaqoob I, Ahmed A (2019) Edge computing: a survey. *Fut Gen Comp Syst.* 97:219–235. <https://doi.org/10.1016/j.future.2019.02.050>
7. Nicolescu R, Huth M, Radanliev P et al (2018) Mapping the values of IoT. *J Inf Technol* 33:345–360. <https://doi.org/10.1057/s41265-018-0054-1>
8. Kazmi A, Jan Z, Zappa, A, Serrano M (2016). Overcoming the heterogeneity in the Internet of things for smart cities. In *International workshop on interoperability and open-source solutions*. Springer, Cham, p. 20–35

9. Bilal K, Khalid O, Erbad A, Khan SU (2018) Potentials, trends, and prospects in edge technologies: fog, cloudlet, mobile edge, and micro data centers. *Comp Netw.* 130:94–120. <https://doi.org/10.1016/j.comnet.2017.10.002>
10. Dizdarevic J, Carpio F, Jukan A, Masip X (2018) A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. *ACM Comp Surv.* <https://doi.org/10.1145/3292674>
11. Khorsand R, Ghobaei-Arani M, Ramezanpour M (2018) FAHP approach for autonomic resource provisioning of multitier applications in cloud computing environments. *Softw Pract Exper* 48:2147–2173. <https://doi.org/10.1002/spe.2627>
12. Shahidinejad A, Ghobaei Arani M (2020) Joint computation offloading and resource provisioning for edge cloud computing environment: a machine learning based approach. *Softw Pract Exper.* 50:2212–2230
13. Milgram P, Kishino F (1994) A taxonomy of mixed reality visual displays. *IEICE Trans. Inform Syst.* E77-D(12):1321–1329
14. <https://www.google.com/glass/start/>. Accessed 1 Jan 2022
15. <https://www.microsoft.com/en-us/hololens>. Accessed 1 Jan 2022
16. Cao A, Dhanaliwala A, Shi J, Gade T, Park B (2019) Image based marker tracking and registration for intraoperative image guided interventions using augmented reality. *Int Soc Opt Phot* 11318:1131802
17. Luyang Liu, Hongyu L, Marco G. (2019). Edge assisted real-time object detection for mobile 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>
18. Tsunazaki S, Nomura R, Komuro T, Yamamoto S, Tsumura N (2018) Reproducing material appearance of real objects using mobile augmented reality. In: IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct) Munich, Germany 196 197 <https://doi.org/10.1109/ISMAR-Adjunct.2018.00065>
19. A. Meka et al., "LIME: Live Intrinsic Material Estimation (2018) IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, p. 6315–6324. <https://doi.org/10.1109/CVPR.2018.00661>.
20. Shannigrahi S, Mastorakis S, Ortega FR (2020) Next-generation networking and edge computing for mixed reality real-time interactive systems. In: IEEE International Conference on Communications Workshops (ICC Workshops) 1 6 <https://doi.org/10.1109/ICCWorkshops49005.2020.9145075>
21. Fu L, Landay J, Nebeling M, Xu Y-Q, Zhao C. (2018). Redefining Natural User Interface. 1–3. <https://doi.org/10.1145/3170427.3190649>
22. Zietsch J, Büth L, Juraschek M, Weinert N, Thiede S, Herrmann C (2019) Identifying the potential of edge computing in factories through mixed reality. *Procedia CIRP.* 81:1095–1100. <https://doi.org/10.1016/j.procir.2019.03.259>
23. Fernández Caramés TM, Fraga Lamas P, Suárez Albela M, Vilar Montesinos M. A fog computing and cloudlet based augmented reality system for the industry shipyard. *Sensors.* 18, 1798. <https://doi.org/10.3390/s18061798>
24. Phupattanasilp P, Tong S-R (2019) Augmented reality in the integrative internet of things (AR-IoT): application for precision farming. *Sustainability* 11:2658. <https://doi.org/10.3390/su11092658>
25. Blanco-Novoa Ó, Fraga-Lamas PA, Vilar-Montesinos M, Fernández-Caramés TM (2020) Creating the internet of augmented things: an open-source framework to make IoT devices and augmented and mixed reality systems talk to each other. *Sensors* 20:3328. <https://doi.org/10.3390/s20113328>
26. Bucsaí S, Kučera E, Haffner O, Drahoš P (2020) Control and monitoring of IoT devices using mixed reality developed by unity engine. *Cybern Inform (K&I) Velke Karlovice, Czech Rep.* <https://doi.org/10.1109/KI48306.2020.9039901>
27. Morris A, Guan J, Lessio N, Shao Y (2020) Toward mixed reality hybrid objects with IoT avatar agents. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada p. 766–773. <https://doi.org/10.1109/SMC42975.2020.9282914>
28. Vidal-Balea A, Blanco-Novoa O, Picallo-Guembe I, Celaya-Echarri M, Fraga-Lamas P, Lopez-Iturri P, Azpilicueta L, Falcone F, Fernández-Caramés TM (2020) Analysis, design and practical validation of an augmented reality teaching system based on microsoft hololens 2 and edge computing. *Eng Proc* 2:52. <https://doi.org/10.3390/ecsa-7-08210>

29. Miraz MH, Ali M, Excell PS, Picking R (2015) A review on Internet of things (IoT), internet of everything (IoE) and internet of nano things (IoNT.) *Int Technol Appl (ITA)* Wrexham. <https://doi.org/10.1109/ITechA.2015.7317398>
30. <https://tej.ie/diors-instagram-ar-virtual-makeup-experience/>. Accessed 1 Jan 2022
31. Bainomugisha Engineer L, Carreton C, Van Andoni S, Tom W, de olfgang M (2012) A survey on reactive programming. *ACM Comput Surv*. <https://doi.org/10.1145/25016542501666>
32. <https://github.com/dotnet/reactive>. Accessed 1 Jan 2022
33. <https://rxjs-dev.firebaseio.com/api>. Accessed 1 Jan 2022
34. Weisshaar BP, Kolnick FC, Kun AI, Mansfield BM. (1987) Computer X Inc., Method of inter-process communication in a distributed data processing system. U.S. Patent. 4 694–396
35. Baskaran S, Nagabushanam HK (2018) Relational localization based Augmented reality Interface for IoT applications. In: 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju p. 103–106. <https://doi.org/10.1109/ICTC.2018.8539469>.
36. Smith NM, Heldt-Sheller N (2020) Intel Corp. Internet of things (iot) network domain resource model. U.S. Patent Application 16/609,711
37. <https://gampltd.com/>. Accessed 1 Jan 2022
38. <https://myhives.eu/>. Accessed 1 Jan 2022
39. <https://ec.europa.eu/easme/en/life>. Accessed 1 Jan 2022
40. <https://pycom.io/product/lopy4/>. Accessed 1 Jan 2022
41. <https://pycom.io/>. Accessed 1 Jan 2022
42. <https://pycom.io/product/fipy/>. Accessed 1 Jan 2022
43. <https://developer.nvidia.com/embedded/jetson-nano-developer-kit>. Accessed 1 Jan 2022
44. <https://www.nvidia.com/>. Accessed 1 Jan 2022
45. <https://www.magicleap.com/en-us>. Accessed 1 Jan 2022
46. <http://zugara.com/>. Accessed 1 Jan 2022
47. <http://openbiometrics.org/>. Accessed 1 Jan 2022
48. <https://github.com/facebookresearch/detectron2>. Accessed 1 Jan 2022
49. Toczé K, Lindqvist J, Nadjm-Tehrani S (2020) Characterization and modeling of an edge computing mixed reality workload. *J Cloud Comp* 9(1):1–24
50. Toczé K, Lindqvist J, Nadjm Tehrani S. (2019). Performance study of mixed reality for edge computing. In: *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing* p. 285–294

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.