HoloRepository: A Cloud-Native Architecture for End-to-End Generation and Distribution of Holographic Medical Visualisations

Group Report

N. BOECKER, W. BOON, U. BOONYAPRASERT, F. MENG, and K. WONG

Recent advancements in Augmented Reality have the potential to facilitate and enhance various tasks and procedures in medicine. One key challenge many applications face is creating and distributing three-dimensional models sourced from traditional two-dimensional imaging studies. While there are existing libraries and workflows to solve this task, they are heavily dependent on technical setup and manual steps. Certain neural network architectures can be leveraged to automate and streamline manual steps such as image segmentation; however, they impose even heavier obstacles in terms of setup, usability, and maintenance.

With the HoloRepository, we propose a distributed, cloud-native architecture that offers a seamless experience for its end-users while performing all steps from browsing existing imaging studies, transforming them into three-dimensional holographic representations, to distributing the results. Depending on the use case, different pipelines extract 3D models from imaging studies accordingly, whereby they exploit a pre-trained neural network to carry out key image processing steps. Furthermore, third-party services can build on top of our system through a well-defined REST API. We demonstrate this capability by implementing a client for the Microsoft HoloLens in addition to our core system, as well as integrating with multiple simultaneously developed projects. By offering this interface, distributing our software in reusable containers, and supporting domain-specific standards such as DICOM and FHIR, we hope to provide a robust basis for future projects.

1 INTRODUCTION

Recent technical advancements in the realm of augmented reality (AR) and the availability of consumer head-mounted display (HMD) devices such as the Microsoft HoloLens (HL) have opened a wealth of opportunities for healthcare applications, particularly in medical imaging. Several approaches have been proposed to transform imaging studies, such as CT or MRI scans, into three-dimensional (3D) models, which can be inspected and manipulated in an AR experience [1, 60, 63, 74]. Generally, all studies agree that the technology is very promising and may even revolutionise the practice of medicine [10]. However, all existing workflows seem to rely on significant manual guidance to conduct steps like segmentation or conversion to polygonal 3D meshes.

Some existing neural networks (NN) can help automate segmentation, the process of partitioning images into semantically meaningful parts [79]. Architectures such as U-Net [67] autonomously create segmentation maps even from little annotated training data [46, 67, 89] . However, translating these advancements from theory to clinical practice has unique challenges: Source code may not be available, documentation may be missing or require too much technical knowledge. Furthermore, different operating systems, software packages and dependencies obstruct successful usage [7].

With medical guidance from Great Ormond Street Hospital (GOSH) and technical support from Microsoft, we propose HoloRepository: A cloud-native architecture which provides a seamless and fully automated workflow for practitioners to generate 3D models from imaging studies, persist them, and dynamically access them in an AR setting. We demonstrate how the modular structure of our system allows us to package any pre-trained NN into a Docker container and incorporate it. To increase interoperability and connect the 3D visualisations with existing patient health records, we support the FHIR (Fast Healthcare Interoperability Resources) R4 standard, and provide a well-defined and stable API for third-party systems. Our results are available on GitHub[1].

[1] https://github.com/nbckr/HoloRepository-Core, https://github.com/nbckr/HoloRepository-HoloLens

2 BACKGROUND

2.1 Industry Standards for Medical Data Exchange

The commonly used standard for creating, transmitting and storing digital imaging studies is DICOM (Digital Imaging and Communications in Medicine). Hospitals, clinics, imaging centers and specialists around the world use it, and a multitude of modalities such as CT scanners or ultrasound produce it [11]. These image acquisition devices, in conjunction with workstations and digital image archives, are referred to as PACS (Picture Archiving and Communication Systems) [62].

While the DICOM standard plays an integral role in medical imaging, the open interoperability standard FHIR has recently attracted the attention of the healthcare community, promising to improve upon legacy standards and revolutionise clinical data exchanges [8]. In the field of radiology, FHIR can lead to more clinically integrated and patient-centered systems and better integration [39]. In FHIR, a DICOM entity is represented as resource type "ImagingStudy"[2]. The metadata stored in DICOM headers get translated to the corresponding FHIR fields, whereas the actual image data remains on the PACS, with the FHIR resource just containing a reference to the endpoint [75].

2.2 Virtual and Augmented Reality in Medicine

The technical advances in VR and AR along with low-cost HMD devices have had a strong impact on medicine in recent years, and sparked a very active area of research [40]. The list of possible use-cases for so-called holograms is long; it includes education, analysis, diagnostics, pre-procedural planning, and intra-operative use. In various settings, studies have shown the new technologies can help medical staff understand complex medical conditions and successfully guide procedures [79].

The most popular hardware platform for AR applications in the medical area is the HoloLens [79]. Since its release in 2016, it inspired an ever-growing body of scholarship; it has been used in medical training [24, 36, 86], surgery planning [2, 17, 44], and guidance of medical instruments [31, 45, 48]. Research has found the device to be ideal to use in sterile surgical settings, as it provides "touch free" operation using gestures [44]. Another study praised that it was perceived as comfortable to wear, easy to use, provided sufficient computing power, and supported high-resolution imaging [34]. Furthermore, research has demonstrated that the HoloLens outperforms comparable devices in terms of contrast perception, task load, and frame rate [65]. While some authors observe that the device at its current stage has certain limitations, every single study we reviewed had an overall positive outcome. With the HoloLens 2 being available for pre-order and promising further improvements w.r.t. usability and the quality of visuals [71] as well as eye tracking, a larger field of view and better hand tracking [76], it is the most appropriate target platform for a system like ours.

2.3 Semi-Automated Approaches to Generating Holograph Medical Visualisations

While the technology to create medical holograms exists, it is rarely used in practice due to high cost and inconvenience. Due to the scarcity of available datasets and the complexity of data synthesis, researchers often have to fall back on datasets created by artists, showing the great demand for accessible systems allowing medical professionals to create holograms [60].

The PEACH (Platform for Enhanced Analytics and Computational Healthcare) Reality project [66] explored the design of a platform for sharing holographic patient cases. A semi-automated machine learning (ML) approach for segmentation was explored; however, the model was limited to one use-case and the data generation took over an hour, therefore the network was eventually not included in their system [80]. Also, while the implemented tool was capable of displaying 3D anatomical reconstructions in VR, there was no fully-developed pipeline for producing these models.

[2] https://www.hl7.org/fhir/imagingstudy.html

Smith [74] proposed a blueprint for a medical holographic pipeline: A mostly manual tool-chain that transforms DICOM files into 3D holograms. In collaboration with Baskaran [5], she drafted MHIF (Medical Holographic Interchange Format), an XML-based file schema for transmitting data to the HoloLens, where it could then be visualised. Matveev et al. [52] subsequently built an accompanying web application, providing basic data management functionalities, such as dynamic file upload. While the results were promising, and were used as a basis for this project, this approach is limited, as it requires a multitude of manual steps. The system's architecture can not be scaled.

Pratt et al. [63] proposed a similar tool-chain for translating DICOM images to HoloLens visualisations. Notably, the results of this study were used to guide operations with real patients. The participating surgeons confirmed that the holograms helped them understand a patient's unique anatomy quickly [57]. While this work marks an important milestone, the content production workflow includes many manual steps, and depends on other full-fledged software suites.

Trestioreanu [79] proposed a software suite for automated ML-based segmentation of medical radiology images. His system provides end-to-end generation from DICOM images to holograms which can be inspected on the HoloLens, and incorporates a convolutional neural network (CNN). This approach was shown to be successful and feasible: While the training period was time-intensive, inference on unseen data took less than 60 seconds. While the overall results were promising, the author stopped short of fully automating the system, leaving some steps like transferring, converting, and loading data to be manually performed. Furthermore, the system is bound to one specific medical use-case and the according NN.

We reviewed several further proposed workflows for translating DICOM images into 3D visualisations for AR or VR usage [1, 10, 18, 41, 59, 70, 77], all of whom shared the same limitations: Medium to high human intervention was required, and no seamless end-to-end pipeline was presented.

2.4 Artificial Neural Networks for Medical Image Segmentation

Several NN-based approaches have been applied to medical image segmentation; the state-of-the-art results have been achieved by CNNs [42, 79]. A particularly successful architecture is U-Net: This network addresses the scarcity of training data for bio-medical tasks by extensive use of data augmentation. It combines a down-sampling path for feature extraction with an up-sampling path for precise localisation of these features in higher resolution layers. Achieving precise outputs in very short processing time [67]. Even four years after the paper was published in May 2015, it is still state-of-the-art and the most commonly used architecture for semantic segmentation tasks [89]. Several modifications and related architectures have been proposed. MultiResUNet has achieved a remarkable gain in performance for challenging image [37]. Both 3D U-Net [22] and V-Net [58] replaced all 2D operations of the original with their 3D counterparts. 3D U-Net introduced the additional benefit of supporting sparely annotated training data.

3 SYSTEM ARCHITECTURE

3.1 System Overview

We designed the HoloRepository as a collection of loosely coupled, microservice-like components. The core system comprises HoloRepository UI, HoloPipelines, and HoloStorage. They provide an interface for practitioners, perform the generation of new 3D models, and persist the data, respectively. Our set of deliverables also includes a HoloLens application that demonstrates the run-time loading capabilities, and a Unity package that allows third-party applications to access the HoloStorage with ease. Furthermore, the extended system interfaces with multiple externally developed components, such as the DepthVisor and Annotator applications. Figure 1 depicts a system architecture diagram.

3.2 Design and Implementation of Sub-Components

3.2.1 HoloRepository UI (Client). The client-side web application allows practitioners to browse authorised patients and manage the creation of 3D models sourced from imaging studies. New holograms can be created by triggering a job in the HoloPipelines component. The job's status updates and log outputs can then be traced. Furthermore, to provide backwards-compatibility with existing work, it is possible to upload existing 3D models directly to the HoloStorageAccessor. The application is written in the typed, syntactic JavaScript superset TypeScript[3] (TS) and uses the popular front-end library React[4]. Global state is coordinated with the modern Context API[5].

3.2.2 HoloRepository UI (Server). The client-side application is accompanied by a middleware which is responsible for communicating with the other components. It acts as an intermediate layer between the application and the FHIR server, the PACS, HoloPipelines and HoloStorageAccessor, and it performs data aggregation, validation, and mapping. Architecturally, this component hides implementation details, third-party endpoints, and secret variables from the client. The server is using Node.js and Express.js, and implements the same TS interfaces as the web app.

3.2.3 HoloPipelines. The HoloPipelines component is key to the overall system, as it performs the automated generation of 3D models. The image processing steps are discussed in Section 4. Throughout the life-time of a processing job, the system keeps track of its current stage and log outputs, to enable external clients to query and visualise in near real-time. Automated garbage collection regularly deletes obsolete logs and intermediate files.

The component is implemented in Python 3. A lightweight Flask[6] server is built around the pipelines, which are implemented according to the pipes-and-filters pattern [53]. They are assembled of tasks, which in turn use services (for complex computations), clients (for external communications), adapters (for communication with well-defined APIs) and wrappers (for calling external programs). All these building blocks have a unified API as far as possible, so they can be reused and rearranged easily. A process pool with multiple workers allows for multiple jobs to be executed concurrently.

The segmentation process is the section which is most heterogeneous for each pipeline, and dictates the required pre- and post-processing tasks. The first general approach to solve this problem is programmatically implementing certain algorithms, such as intensity thresholding, morphological operations, and largest connected components. We demonstrate this with a pipeline wrapping an existing implementation of a lung and airway segmentation[7] originated from Feng et al. [27] that uses all of these techniques (Figure 2a). Furthermore, we implemented a pipeline that demonstrates how to operate on a lower level; it leverages the scikit-image library[8] to programmatically perform a marching cube algorithm with a Hounsfield threshold of 300, effectively delineating bone structures from CT scans (Figure 2b). The second general approach is using artificial NNs. Pre-trained networks are encapsulated in containers and incorporated in pipelines, implementing a strategy pattern [28]. From the service's point of view, the networks are black boxes. After preparing the input accordingly, an HTTP client posts it to the network and awaits the segmented output before proceeding. This way, any existing or future model can be integrated by wrapping it in a light HTTP interface. We provided detailed instructions on how to extend the catalogue with further models.

[3] https://www.typescriptlang.org
[4] https://www.reactjs.org
[5] https://reactjs.org/docs/context.html
[6] https://www.flask.palletsprojects.com
[7] https://www.github.com/wanwanbeen/ct_lung_segmentation
[8] https://scikit-image.org

3.2.4 Pre-Trained Neural Networks. In the ML community, various platforms for distributing pre-trained models are emerging. We evaluated several such platforms[9]; some of them propose their own standard APIs, often built on top of Docker. Most offer a small selection of pre-trained models that are relevant w.r.t. medical segmentation. To demonstrate the modular nature of the architecture presented above, we intentionally chose a model which does not already ship in a Docker container: We implemented a pipeline that integrates a pre-trained NiftyNet model from Gibson et al. [32], a Dense-V-Net fully convolutional network (FCN) trained to conduct automatic segmentation of eight abdominal organs from CT scans (Figure 2c).

3.2.5 HoloStorage. The HoloStorage combines a blob storage database to host the binary 3D models with a FHIR server to store structured medical data related to the models. For the FHIR server, we evaluated the managed Azure API for FHIR [20], which includes an underlying CosmosDB database, support up to FHIR STU3, and a simplified setup. FHIR Server for Azure [21] is an alternative service, which comes with an SQL database instead, and requires more manual setup and maintenance, while also providing more flexibility. We eventually opted for the latter, due to FHIR R4 support.

3.2.6 HoloStorageAccessor. To protect the databases from unintended access and hide concrete implementation details, the Accessor provides a consistent and unified interface for all data consumers and producers. Beyond this façade property, it performs the necessary data mapping and aggregation, orchestrates FHIR queries, and combines the results. As this component represents the single entry-point for third-party systems, we designed a RESTful API as an interactive OpenAPI[10] specification, which we shared with all relevant stakeholders. Subsequently, we evaluated two code generators[11] and multiple different configurations, along with the programming languages they produce. We eventually used OpenAPI Generator, configured to create Go code with the Gin HTTP web framework embedded[12], and used this blueprint to build the Accessor application upon. This decision was mostly based on Go's inherent support for concurrency and its superior performance, outperforming equivalent Python code by a factor of around 40 [23]. With the Accessor being the focal point of the system, performance and reliability were of paramount importance.

3.2.7 HoloStorageConnector. To facilitate the access to the HoloStorageAccessor for third-party developers, we provide a Unity/C# library distributed as AssetPackage. This is the encouraged approach of sharing open-source Unity packages [49] and facilitates import into other Unity projects. By abstracting the network calls to the API and providing higher-level abstractions of the domain entities, we hope to encourage future work to integrate with the HoloRepository.

3.2.8 HoloLens Demo Application. To demonstrate how AR applications on HMDs can dynamically consume data from the system, we implemented a basic HoloLens application that connects to the Accessor API at run-time and visualises holograms alongside additional patient data. The application provides basic manipulation of the objects, such as scaling and rotating. From a system standpoint, it also serves as an integration test, ensuring the end-to-end functionality of the system. Lastly, the Unity/C# code serves as an example and interactive documentation of the HoloStorageAccessor API and how to access it with the HoloStorageConnector. Both HoloLens components are built with Mixed Reality Toolkit (MRTK) version 2[13]. As a result, even though the project utilised the first generation of HoloLens devices, we anticipate porting it to the HoloLens 2 with minimal effort.

[9] https://github.com/NifTK/NiftyNetModelZoo, https://www.gallery.azure.ai, https://www.modelzoo.co, http://www.modelhub.ai, https://www.github.com/DLTK/models, http://www.deepinfer.org/pages/models

[10] https://swagger.io/docs/specification/about

[11] OpenAPI Generator (https://openapi-generator.tech), Swagger Codegen (https://swagger.io/tools/swagger-codegen)

[12] https://www.openapi-generator.tech/docs/generators/go-gin-server

[13] https://www.microsoft.github.io/MixedRealityToolkit-Unity

3.2.9 Third-Party Applications and Envisioned Future Work. At the time of writing, four concurrently developed applications aiming to plug into the HoloRepository system are in the works [9, 25, 72, 73]. In the HoloRepository system specification, we proposed further potential extensions [12].

3.3 System Integration Points

The public interface to store and retrieve data is the aforementioned HoloStorageAccessor API (Section 3.2.6). It is open to external systems, while also being used by the data-producing internal components (HoloPipelines and HoloRepository UI). When constructing the API, we made sure to follow best practices of RESTful design, with well-defined semantics and consistent naming choices. We followed semantic versioning practices [64]. Furthermore, we equipped the API specification with an exhaustive OpenAPI documentation and a complete change log. In addition to this public API, the system has multiple internal APIs for inter-system communications. The internal data model and the scheme of the JSON transfer objects are specified in code as TS interfaces.

To facilitate API development and testing, we created collections of pre-defined requests with Postman[14]. We added these configurations to version control as a means of reference and documentation. Furthermore, as we anticipate the key future consumers to be HoloLens applications, we implemented the Connector library to facilitate Unity integration (Section 3.2.7).

As discussed in Section 4.1, the HoloRepository system also consumes data from an external FHIR server, which represents the hospital's EHR. Potentially, the second FHIR server, which stores the generated holograms, could be opened up, for other medical systems to directly interface rather than through the Accessor API, and fully take advantage of FHIR's interoperability benefits.

3.4 Infrastructure and Deployment

The increasing amount of imaging data in recent years has driven PACS and EHR vendors towards utilising cloud computing and storage [6], mirroring the general tendency in software engineering [61, 85]. In line with this trend, we designed the HoloRepository as a cloud-native, distributed system. All components have been containerised and represent independently deployable entities. To facilitate deployment to different environments, we separated configuration from code [85].

For containerisation, we employed the popular Docker platform[15]. Besides being the de-facto industry standard, Docker is known to facilitate reproducibility of research projects [14], help data scientist manage dependencies and requirements [55], and not causing noticeable performance drawbacks while running ML tools [87]. For local development, we added a Docker Compose configuration. For production deployment, we set up Kubernetes[16] so as to benefit from a multitude of orchestration features, such as load-balancing, easy scaling, and environment-agnostic portability.

Adopting the platform-as-a-service (PaaS) model, we hosted the core system on an Azure Kubernetes Service (AKS)[17] cluster interlinked with an Azure Container Repository (ACR)[18], in addition to installing Azure DevOps build and release pipelines linked to our code repositories. Microsoft's cloud platform is particularly attractive to healthcare cloud adopters as it facilitates integrating with health IT infrastructure already running Microsoft products, and helps customers comply with laws regulating protected health information (PHI) [26]. Notably, major EHR vendor Epic Systems leveraged the platform for developing its predictive analytics and artificial intelligence products [16]. Furthermore, Microsoft has repeatedly demonstrated its support for FHIR [51] and stands apart from other cloud platform providers by offering a multitude of FHIR products [20, 21].

14 https://www.getpostman.com
15 https://www.docker.com
16 https://www.kubernetes.io
17 https://azure.microsoft.com/en-gb/services/kubernetes-service
18 https://azure.microsoft.com/en-gb/services/container-registry

3.5 Authorisation and Authentication

The commonly adopted approach to authorisation and authentication for FHIR-related applications is SMART on FHIR. SMART (Substitutable Medical Applications and Reusable Technologies) provides a set of open specifications to integrate apps with health IT systems [19]. In earlier work, we demonstrated how to implement a complete SMART application launch [13]; as it is not the focus of this project, the launch sequence is just simulated in this work. We did, however, conceive and implement a basic concept of SMART authentication with the HoloLens. A regular SMART login, if implemented in a HoloLens app, would require the user to type their password in AR. This can be perceived as cumbersome, and a PIN system is likely to be perceived as more usable [88] while providing sufficient security [30]. After a practitioner is authenticated with our system's UI, a unique, temporary (although currently hard-coded) PIN gets assigned. Using this number, they can authenticate themselves in the HoloLens application, and access authorised holograms.

4 DATA ARCHITECTURE AND IMAGE PROCESSING CHAIN

4.1 Medical Input Data

From GOSH's radiology department, we obtained five de-identified DICOM imaging studies that were suited as inputs for the three pipelines we chose to implement first (Table 1). We made these data publicly available to facilitate reproducing our efforts[19]. In addition, we drew studies from multiple open data-sets, such as the Cancer Imaging Archive[20]. We set up a data container in an Azure Blob Storage to represent the hospital's PACS and host the DICOM data.

In order to embed the imaging data into patient-centric, inter-connected FHIR health records, we generate a population of synthetic patients using Synthea [81]. While the software supports "ImagingStudy" resources, the generated data was too generic for our purposes. Hence, we processed and enriched it with more details. Most importantly, we pointed the endpoints to their counterpart DICOM files on the PACS stand-in. To represent the hospital's EHR, we deployed a recent release of Microsoft Server for Azure, supporting the normative FHIR R4 release [21, 33] and proven to support Synthea-generated patient data [35]. After uploading the modified synthetic data to the server, they would then be queried by the UI server, to translate them into the internal scheme.

4.2 Intermediate Three-Dimensional Image Data

Upon receiving a request, the HoloPipelines fetch the DICOM files from the specified endpoint and translate them into 3D NumPy arrays, the standard representation for numerical data, providing efficient numerical computations [82]. The following manipulation steps vary for each pipeline, they can for instance include normalisation, scaling, resampling, and cropping. Some workflows require temporary transformation to NIfTI (Neuroimaging Informatics Technology Initiative). Originally conceived by neuroimaging scientists, this format is often used as a simpler alternative to DICOM [47]; for example, the pre-trained NN we integrated requires NIfTI input.

After the segmentation is finished, the numerical data are translated to a polygonal mesh utilising a marching cubes algorithm [50]. These 3D data are then transiently stored as OBJ files [84].

4.3 File Size Reduction, Mesh Simplification and Compression

As OBJ files contain uncompressed plain text, they are rather heavyweight, and therefore not well suited to dynamically load onto mobile devices. We evaluated alternative 3D formats for our final 3D models, including glTF (GL Transmission Format), FBX, DAE (Collada), STL (STereoLithography), and 3DS. While FBX seems to be the most prominent, and is deeply integrated into Unity, it

[19] DOI: 10.13140/RG.2.2.28491.95521
[20] https://www.cancerimagingarchive.net/

is proprietary and comes with a restrictive license [3]. Instead, we opted to use glTF, as it has recently received a lot of positive attention in professional circles [4, 15] as well as strong industry support [43]. Among the supporters is Microsoft, promising future support [54] . Often coined the "JPEG of 3D", glTF compresses the file size of 3D objects while also minimising run-time processing. Furthermore, it is royalty-free and has been equipped with a rich set of tooling [43].

To further reduce the file size, we explored various other tools and algorithms. We leveraged Fast Quadric Mesh Simplification (FQMS), an implementation of the quadric-based edge-collapse mesh simplification method [29] which can simplify 2 million triangles to 30,000 in 3.5 seconds[21]. After examining various down-sampling and simplification ratios (Figure 3, Table 2), we found that FQMS provides an acceptable trade-off between processing time and model size. Therefore, we implemented a wrapper to call the distributed binary and integrated it in all pipelines.

Lastly, we evaluated Draco [78], an open-source library recently released for compressing and decompressing 3D geometric meshes. By implementing Edgebreaker 3D compression [68], among other algorithms, this library achieves impressive compression ratios: Using a 52.6 MB OBJ file, we were able to decrease the size to around 1 MB using Draco on top of FQMS, adding less than 3 seconds of additional overhead (Table 2). While our experiments found significantly reduced file sizes in just slightly longer processing time, we also found that Draco compression relies on client-side support. Although there are Unity plugins available[22], they turned out to be unreliable and not fit for production use yet. Hence, while we do believe that Draco has the potential to become the next 3D compression standard, we eventually had to remove it from our project.

4.4 Output Data

We reviewed several existing standards to represent our output data: The IEEE's P3333.2.2 Standard for 3D Medical Visualisation, for instance, covers the reconstruction of 3D models from 2D medical images [38]. While this standard seemed like a great fit at first, it has not gained a lot of momentum, and seems to lack community support. We also considered MHIF [5, 74]; this scheme was conceived for the exact use-cases of our system, but it has some known limitations [74]. Furthermore, it is somewhat redundant with FHIR. Considering the current drive behind FHIR, we agreed that supporting the standard, rather than reinventing the wheel, would maximise the impact of our system. We therefore came up with a data scheme which is entirely compliant with FHIR R4 and centers around "DocumentReference" resources[23], representing a document of any kind and its metadata. Our mapping translates holograms into such documents, and by doing so, embeds them in the broader network of information provided by FHIR. Via their "Context" field, documents can reference any number of other resources, which may be useful for third-party use-cases like storing annotations related to specific holograms. Similar to the "ImagingStudy" resource design, the actual payload is stored externally, and a reference to the endpoint is included in the document's "Attachment" field. The GLB file (the binary variation of glTF) is stored on an Azure Blob Storage.

Our system generates these resources and feeds them into a second FHIR server we set up as part of the HoloStorage component, independently from the one that represents the EHR. This was necessary to decouple the new holographic data from existing health records and overcome the inherent issue of most EHRs in production offering only read-only access to third-party systems. Through references such as the patient ID, the newly generated data can still be affiliated to the original health records. Furthermore, we copy a subset of the FHIR data over to the HoloStorage FHIR server, in order to make it self-contained.

[21] https://github.com/sp4cerat/Fast-Quadric-Mesh-Simplification
[22] https://github.com/google/draco/tree/master/unity, https://github.com/atteneder/DracoUnity
[23] https://www.hl7.org/fhir/documentreference.html

5 EVALUATION

Overview. We successfully implemented and integrated all components of the cloud-hosted HoloRepository core system, as well as a demo application and supporting library for HoloLens usage. Using the most appropriate languages and frameworks for each independent component, we delivered a substantial code-base of more than 10,000 lines of code (Table 3), packaged and deployed in environment-agnostic containers. All key functional requirements that were laid out in Section 1 were met, and the system overcomes the main drawbacks of previous approaches.

Figure 4 illustrates an end-to-end use-case based on screenshots of the developed system.

Validation. Each component is tested by means of the respective programming language's common test frameworks. For crucial components, such as HoloPipelines and HoloStorageAccessor, statement coverage is above 70%. Furthermore, manual integration testing was performed at multiple stages. Due to the distributed nature of the system, the individual test suites require a lot of mock objects to abstract inter-component interactions.

Performance. With the HoloPipelines deployed on AKS, we measured how long the separate processing stages take (Table 4). The results show that all pipelines finish after 30 – 40 seconds. Also, the overhead of network transmission is nearly negligible, whereas the process of reading and normalising DICOM files to NumPy arrays makes for a substantial proportion of the run-time.

All user-facing components were put under load testing to identify how many concurrent users they can handle concurrently. We used the Locust framework[24], which provides reproducible user behaviour simulations. Running the components on a single node, fast responses are provided for up to 75 concurrent users (Table 5). With our AKS deployment, it is possible to increase the number of instances to account for increased load. From the HoloLens Demo App, retrieving the textual metadata takes less than 0.5 seconds, and downloading a typical hologram file (between 2 and 5 MB) takes less than 3 seconds.

Availability. The system is fully deployed on an AKS cluster, and thus benefits from the cloud provider's guaranteed high-availability standards and zero-downtime deployments. The SLA (Service Level Agreement) states a VM up-time of 99.99% [56]; applications and data are automatically replicated across multiple data-centers in different physical locations[51]..

Maintainability. Our code-base is extensively documented and contains highly readable code, in order to help future maintainers quickly comprehend the system and make changes when required. However, tracing errors in the deployed system will be challenging, as logs and errors are currently only accessible on the respective components' standard outputs, and not consolidated centrally. Traceability in our AKS deployment is further impaired, as multiple concurrent instances of the same component are running. An improved version of the system would implement a better logging design [69], and utilise a centralised log management platform such as Logstash[25].

Scalability. In most respects, our system is ready to be scaled up: Given the containerised, independent components and the AKS deployment, powerful cloud features such as auto-scaling and load-balancing can be leveraged. However, system-wide scalability is limited by an architectural flaw in the HoloPipelines component: Against common advice [85], the component keeps track of current jobs' state and logs locally. Because of this, when a client triggers a job, the system would always have to route future requests to the same instance—effectively limiting the number of HoloPipelines instances to one. An improved system architecture could include an additional service to keep track of all jobs, independently from the container they are physically executing in.

[24] https://locust.io
[25] https://www.elastic.co/de/products/logstash

Security. As discussed in Section 3.5, security and privacy protection mechanisms are only simulated in this project. While this drawback should be addressed in future work, we did conceive and implement, in outline, a new concept for remote SMART authentication, temporarily coupling a practitioner's authentication with a HoloLens HMD. This is an important first step towards combining SMART on FHIR with AR apps, as, to our knowledge, there is no existing work to do so.

Extensibility. Our system is modular on different levels: The code itself is split into small entities, implementing a separation of responsibilities, and the components themselves are encapsulated in containers. As a result, it is easy to make minor changes, but also feasible to replace whole system components, or extract just selected services. Through the aforementioned concept of encapsulating existing NNs and the Accessor API, the system provides abundant opportunities for extensions.

Interoperability. As discussed in Section 4, the system is compliant with the industry standards of medical data exchange. By supporting the normative FHIR R4 standard, high interoperability with other medical systems will most likely be provided for years to come. Additionally, FHIR support is believed to enable the development of ML-based applications and services [51].

Usability. Both of our client-facing interfaces were peer-reviewed and provide a basic level of usability. Seamless end-to-end workflows are a significant improvement to previous work. However, the three pipelines we implemented may not have the highest real-life value for practitioners; they were chosen from a rather technical perspective. Inherently, results are also dependent on the quality of the incorporated models; research shows that NN-based segmentation may not yet be sufficiently accurate and robust for clinical use [83].While the system is very modular and extensible, the steps required to add new NNs to the catalogue are currently still too complex.

Reproducibility. To prevent the issues we encountered with previous work, we equipped our system with extensive documentation, scripts and instructions for setting up infrastructure, along with sample data to reproduce the system from scratch. Furthermore, for future developers' convenience, we included several other helpful items, including our CI configuration, Postman API endpoint test collections, and scripts for pre-processing Synthea-generated FHIR inputs.

Software Engineering Process. We adopted an agile development workflow, working to resolve user stories from a previously determined back-log in an iterative manner. The distinct components were worked on in parallel by different team members, allowing us to build up the complete architecture gradually in a bottom-up approach. Every code change went through a code review process, ensuring exchange of knowledge within the team and high quality of the software outcomes.

6 CONCLUSION

With the HoloRepository, we developed a cloud-native architecture which allows practitioners to transform traditional imaging studies into medical holographic visualisations, and inspect them on HMD devices such as the HoloLens in a seamless and automated end-to-end experience. Founded on thorough research and building on top of promising earlier work, the resulting system overcomes the main drawbacks of earlier approaches by offering complete automation of the workflow and availability as a cloud-hosted service. We introduced a modular, extensible concept on how to incorporate pre-trained NNs, which will allow data scientists to gradually build up a catalogue of integrated networks. With one such NN, and two traditional segmentation pipelines, we demonstrated the end-to-end capability of generating medical holograms automatically and within feasible processing time. Lastly, the system is topped off with implementing industry standards, such as DICOM and FHIR, and providing a well-defined and stable API for third-party systems. With this compelling result, we hope to provide a robust basis for future projects.

REFERENCES

[1] Raffael Affolter, Sebastian Eggert, Till Sieberth, Michael Thali, and Lars Christian Ebert. 2019. Applying augmented reality during a forensic autopsy – Microsoft HoloLens as a DICOM viewer. *Journal of Forensic Radiology and Imaging* 16 (March 2019), 5–8. https://doi.org/10.1016/j.jofri.2018.11.003

[2] Khaled Amer, Nashaat El-Khamisy, Rafeek Mamdouh, and Alaa Riad. 2018. Survey on Virtual Reality, Augmented Reality and Mixed Reality Techniques for Liver Surgical Operations and Training. (09 2018).

[3] Autodesk. 2014. Autodesk FBX SDK 2014 - Licence and Services Agreement. https://damassets.autodesk.net/content/dam/autodesk/www/Company/docs/pdf/legal-notices-&-trademarks/Autodesk_FBX_SDK_2014_LSA_-_English.pdf. (2014). (Accessed on 08/25/2019).

[4] Gabe Baker. 2018. glTF is a Huge Deal for VR and AR. Here's Why We're Embracing It. https://www.worldviz.com/post/gltf-is-a-huge-deal-for-vr-ar-heres-why-were-embracing-it. (8 2018). (Accessed on 08/25/2019).

[5] Immanuel Baskaran. 2018. Medical Holographic File Format Design. https://goshmhif.azurewebsites.net/Hololens-Webapp/#/documentation.. (08 2018). (Accessed on 08/23/2019).

[6] Luis Bastiao Silva, Carlos Costa, Augusto Silva, and Jose Oliveira. 2011. A PACS gateway to the cloud. *Proceedings of the 6th Iberian Conference on Information Systems and Technologies, CISTI 2011*, 1–6.

[7] Andrew Beers, James Brown, Ken Chang, Katharina Hoebel, Elizabeth Gerstner, Bruce Rosen, and Jayashree Kalpathy-Cramer. 2018. DeepNeuro: An open-source deep learning toolbox for neuroimaging. *arXiv:1808.04589 [cs]* (Aug. 2018). http://arxiv.org/abs/1808.04589 arXiv: 1808.04589.

[8] Duane Bender and Kamran Sartipi. 2013. HL7 FHIR: An agile and RESTful approach to healthcare information exchange. *Proceedings of CBMS 2013 - 26th IEEE International Symposium on Computer-Based Medical Systems*, 326–331. https://doi.org/10.1109/CBMS.2013.6627810

[9] Wilfrid Berry. 2019. *Live Surgery in Microsoft Teams (Work in Progress).* Master's thesis. University College London, London.

[10] Alaa Beydoun, Vikash Gupta, and Eliot Siegel. 2017. DICOM to 3D Holograms: Use Case for Augmented Reality in Diagnostic and Interventional Radiology. *SIIM Scientific Session Posters and Demonstrations* (2017), 4.

[11] Jr Bidgood, W. Dean, Steven C. Horii, Fred W. Prior, and Donald E. Van Syckle. 1997. Understanding and Using DICOM, the Data Interchange Standard for Biomedical Imaging. *Journal of the American Medical Informatics Association* 4, 3 (1997), 199–212. https://doi.org/10.1136/jamia.1997.0040199

[12] Niels Boecker, Wenjie Boon, Udomkarn Boonyaprasert, Fanbo Meng, and Kawai Wong. 2019. HoloRepository: System Design. (6 2019).

[13] Niels Boecker, Joshua Zeltser, Fabiha Ahmed, Fanbo Meng, Chong Yang, Ying Huang, and Kawai Wong. 2019. Speedy Recovery: A Role-Based, Portable Health IT Application Built on the SMART on FHIR Platform. https://github.com/nbckr/speedy-recovery/raw/master/reports/SpeedyRecovery_ProjectReport.pdf. (3 2019). (Accessed on 03/11/2019).

[14] Carl Boettiger. 2015. An Introduction to Docker for Reproducible Research. *SIGOPS Oper. Syst. Rev.* 49, 1 (Jan. 2015), 71âĂŞ79. https://doi.org/10.1145/2723872.2723882

[15] Arthur Brainville. 2018. Why glTF 2.0 is awesome. https://blog.ybalrid.info/2018/07/why-gltf-2-0-is-awesome/. (7 2018). (Accessed on 08/25/2019).

[16] Jennifer Bresnick. 2017. Epic Systems: Machine Learning Is the EHR Usability Solution. https://healthitanalytics.com/features/epic-systems-machine-learning-is-the-ehr-usability-solution. (9 2017). (Accessed on 08/26/2019).

[17] H Brun, R A B Bugge, L K R Suther, S Birkeland, R Kumar, E Pelanis, and O J Elle. 2018. Mixed reality holograms for heart surgery planning: first user experience in congenital heart disease. *European Heart Journal - Cardiovascular Imaging* 20, 8 (2018), 883–888. https://doi.org/10.1093/ehjci/jey184

[18] Arthur A. Bucioli, Edgard A. Lamounier, Gerson F. M. Lima, and Alexandre Cardoso. 2018. Automated Generation of Holographic Heart Visualization from Coronary Tomography for Multi-place Medical Diagnostics using Holographic devices. (2018). http://www.modsimworld.org/papers/2018/MODSIM_2018_Paper_not_required_for_a_48.pdf

[19] Joshua C. Mandel, David A. Kreda, Kenneth Mandl, Isaac Kohane, and Rachel Ramoni. 2016. SMART on FHIR: A standards-based, interoperable apps platform for electronic health records. *Journal of the American Medical Informatics Association* 23 (02 2016), ocv189. https://doi.org/10.1093/jamia/ocv189

[20] Heather Jordan Cartwright. 1019. Lighting up healthcare data with FHIR: Announcing the Azure API for FHIR. https://azure.microsoft.com/en-us/blog/lighting-up-healthcare-data-with-fhir-announcing-the-azure-api-for-fhir/. (2 1019). (Accessed on 08/26/2019).

[21] Heather Jordan Cartwright. 2019. Microsoft FHIR Server for Azure extends to SQL. https://azure.microsoft.com/en-in/blog/microsoft-fhir-server-for-azure-extends-to-sql/. (6 2019). (Accessed on 08/25/2019).

[22] Özgün Cicek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 2016. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. *arXiv:1606.06650 [cs]* (June 2016). http://arxiv.org/abs/1606.06650 arXiv: 1606.06650.

[23] Computer Language Benchmarks Game. [n. d.]. Go vs Python 3 – Which programs are fastest? https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/go-python3.html. ([n. d.]). (Accessed on 08/28/2019).

[24] Sara Condino, Giuseppe Turini, Paolo D. Parchi, Rosanna M. Viglialoro, Nicola Piolanti, Marco Gesi, Mauro Ferrari, and Vincenzo Ferrari. 2018. How to Build a Patient-Specific Hybrid Simulator for Orthopaedic Open Surgery: Benefits and Limits of Mixed-Reality Using the Microsoft HoloLens. (2018). https://doi.org/10.1155/2018/5435097

[25] Jacob Currant. 2019. *HoloRepository Annotator (Work in Progress).* Master's thesis. University College London, London.

[26] EHR Intelligence. 2018. What Makes Azure Attractive to Healthcare Cloud Adopters? https://ehrintelligence.com/news/what-makes-azure-attractive-to-healthcare-cloud-adopters. (5 2018). (Accessed on 08/26/2019).

[27] Xinyang Feng, Jie Yang, Andrew F. Laine, and Elsa D. Angelini. 2017. Discriminative Localization in CNNs for Weakly-Supervised Segmentation of Pulmonary Nodules. *arXiv e-prints*, Article arXiv:1707.01086 (Jul 2017), arXiv:1707.01086 pages. arXiv:cs.CV/1707.01086

[28] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design Patterns: Elements of Reusable Object-oriented Software.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[29] Michael Garland and Paul S. Heckbert. 1997. Surface Simplification Using Quadric Error Metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97).* ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 209–216. https://doi.org/10.1145/258734.258849

[30] Ceenu George, Mohamed Khamis, Emanuel von Zezschwitz, Marinus Burger, Henri Schmidt, Florian Alt, and Heinrich Hussmann. 2017. Seamless and Secure VR: Adapting and Evaluating Established Authentication Systems for Virtual Reality. In *Network and Distributed System Security Symposium (NDSS 2017) (USEC '17).* NDSS. https://doi.org/10.14722/usec.2017.23028

[31] Jacob T. Gibby, Samuel A. Swenson, Steve Cvetko, Raj Rao, and Ramin Javan. 2019. Head-mounted display augmented reality to guide pedicle screw placement utilizing computed tomography. *International Journal of Computer Assisted Radiology and Surgery* 14, 3 (March 2019), 525–535. https://doi.org/10.1007/s11548-018-1814-7

[32] E. Gibson, F. Giganti, Y. Hu, E. Bonmati, S. Bandula, K. Gurusamy, B. Davidson, S. P. Pereira, M. J. Clarkson, and D. C. Barratt. 2018. Automatic Multi-Organ Segmentation on Abdominal CT With Dense V-Networks. *IEEE Transactions on Medical Imaging* 37, 8 (Aug 2018), 1822–1834. https://doi.org/10.1109/TMI.2018.2806309

[33] Grahame Grieve. 2019. FHIR R4 is published. https://onfhir.hl7.org/2019/01/02/fhir-r4-is-published/. (01 2019). (Accessed on 03/06/2019).

[34] Matthew G. Hanna, Ishtiaque Ahmed, Jeffrey Nine, Shyam Prajapati, and Liron Pantanowitz. 2018. Augmented Reality Technology Using Microsoft HoloLens in Anatomic Pathology. *Archives of Pathology & Laboratory Medicine* 142, 5 (May 2018), 638–644. https://doi.org/10.5858/arpa.2017-0189-OA

[35] Michael S. Hansen. 2018. Generating and Loading Synthetic Patient to FHIR Server using Azure Container Instances. https://blogs.msdn.microsoft.com/mihansen/2018/08/06/generating-and-loading-synthetic-patient-to-fhir-server-using-azure-container-instances/. (8 2018). (Accessed on 08/25/2019).

[36] Gregor Kurt Höblinger. 2018. *Learning Anatomy with Mixed Reality: Using the Microsoft HoloLens for MRI image visualization in three-dimensional space.* Master's thesis. St. PÃűlten University of Applied Sciences, St. PÃűlten.

[37] Nabil Ibtehaz and M. Sohel Rahman. 2019. MultiResUNet : Rethinking the U-Net Architecture for Multimodal Biomedical Image Segmentation. *arXiv e-prints*, Article arXiv:1902.04049 (Feb 2019), arXiv:1902.04049 pages. arXiv:cs.CV/1902.04049

[38] IEEE Standards Association. [n. d.]. P3333.2.2 - Standard for Three-Dimensional (3D) Medical Visualization. ([n. d.]). https://standards.ieee.org/project/3333_2_2.html

[39] Peter I. Kamel and Paul G. Nagy. 2018. Patient-Centered Radiology with FHIR: an Introduction to the Use of FHIR to Offer Radiology a Clinically Integrated Platform. *Journal of Digital Imaging* 31, 3 (June 2018), 327–333. https://doi.org/10.1007/s10278-018-0087-6

[40] Leigh Kamping-Carder. 2018. How Holograms Are Helping Medical Training. https://www.wsj.com/articles/how-holograms-are-helping-medical-training-1530795601. (07 2018). (Accessed on 08/23/2019).

[41] Christof Karmonik, Timothy B. Boone, and Rose Khavari. 2018. Workflow for Visualization of Neuroimaging Data with an Augmented Reality Device. *Journal of Digital Imaging* 31, 1 (01 Feb 2018), 26–31. https://doi.org/10.1007/s10278-017-9991-4

[42] Baris Kayalibay, Grady Jensen, and Patrick van der Smagt. 2017. CNN-based Segmentation of Medical Imaging Data. *arXiv e-prints*, Article arXiv:1701.03056 (Jan 2017), arXiv:1701.03056 pages. arXiv:cs.CV/1701.03056

[43] Khronos Group Inc. [n. d.]. glTF Runtime 3D Asset Delivery. https://www.khronos.org/gltf/. ([n. d.]). (Accessed on 08/25/2019).

[44] Pieter L. Kubben and Remir S. N. Sinlae. 2019. Feasibility of using a low-cost head-mounted augmented reality device in the operating room. *Surgical Neurology International* 10 (Feb. 2019). https://doi.org/10.4103/sni.sni_228_18

[45] I. Kuhlemann, M. Kleemann, P. Jauer, A. Schweikard, and F. Ernst. 2017. Towards X-ray free endovascular interventions âĂŞ using HoloLens for on-line holographic visualisation. *Healthcare Technology Letters* 4, 5 (2017), 184–187. https:

//doi.org/10.1049/htl.2017.0061

[46] Curtis Langlotz. 2017. Medical ImageNet: A Resource for Machine Learning from Medical Images. https://gputechconf2017.smarteventscloud.com/connect/sessionDetail.ww?SESSION_ID=110157. (5 2017). (Accessed on 08/28/2019).

[47] Xiangrui Li, Paul S. Morgan, John Ashburner, Jolinda Smith, and Christopher Rorden. 2016. The first step for neuroimaging data analysis: DICOM to NIfTI conversion. *Journal of Neuroscience Methods* 264 (2016), 47 – 56. https://doi.org/10.1016/j.jneumeth.2016.03.001

[48] Ye Li, Xiaolei Chen, Ning Wang, Wenyao Zhang, Dawei Li, Lei Zhang, Xin Qu, Weitao Cheng, Yueqiao Xu, Wenjin Chen, and Qiumei Yang. 2018. A wearable mixed-reality holographic computer for guiding external ventricular drain insertion at the bedside. *Journal of Neurosurgery JNS* (2018), 1 – 8. https://thejns.org/view/journals/j-neurosurg/aop/article-10.3171-2018.4.JNS18124.xml

[49] Andrew Lord. 2016. Writing an Open Source Unity Package. https://medium.com/@lordcodes/writing-an-open-source-unity-package-877bad3c8913. (11 2016). (Accessed on 08/29/2019).

[50] William E. Lorensen and Harvey E. Cline. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*. ACM, New York, NY, USA, 163–169. https://doi.org/10.1145/37401.37422

[51] Josh Mandel. 2019. Microsoft hosts HL7 FHIR DevDays. https://azure.microsoft.com/en-us/blog/microsoft-hosts-hl7-fhir-devdays/. (6 2019). (Accessed on 08/26/2019).

[52] Road Matveev, Lovepreet Singh, and Jeff Cai. 2019. HoloRespository Prototyoe. http://students.cs.ucl.ac.uk/2018/group19/index.html. (2019). (Accessed on 08/04/2019).

[53] Microsoft Corporation. 2017. Pipes and Filters pattern - Cloud Design Patterns. https://docs.microsoft.com/en-us/azure/architecture/patterns/pipes-and-filters. (06 2017). (Accessed on 08/16/2019).

[54] Microsoft Corporation. 2018. Create 3D models for use in the home. https://docs.microsoft.com/en-us/windows/mixed-reality/creating-3d-models-for-use-in-the-windows-mixed-reality-home. (3 2018). (Accessed on 08/25/2019).

[55] Microsoft Corporation. 2018. Demystifying Docker for Data Scientists. https://blogs.technet.microsoft.com/machinelearning/2018/03/15/demystifying-docker-for-data-scientists-a-docker-tutorial-for-your-deep-learning-projects/. (3 2018). (Accessed on 08/27/2019).

[56] Microsoft Corporation. 2018. SLA for Virtual Machines. https://azure.microsoft.com/en-gb/support/legal/sla/virtual-machines/v1_8/. (3 2018). (Accessed on 08/28/2019).

[57] Microsoft Corporation. 2018. Surgeons are using HoloLens to 'see inside' patients before they operate on them. https://news.microsoft.com/en-gb/2018/02/08/surgeons-use-microsoft-hololens-to-see-inside-patients-before-they-operate-on-them/. (2 2018). (Accessed on 08/27/2019).

[58] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. 2016. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. *arXiv e-prints*, Article arXiv:1606.04797 (Jun 2016), arXiv:1606.04797 pages. arXiv:cs.CV/1606.04797

[59] C. M. Morales Mojica, N. V. Navkar, N. V. Tsekos, D. Tsagkaris, A. Webb, T. Birbilis, and I. Seimenis. 2017. Holographic Interface for three-dimensional Visualization of MRI on HoloLens: A Prototype Platform for MRI Guided Neurosurgeries. In *2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE)*. 21–27. https://doi.org/10.1109/BIBE.2017.00-84

[60] Michael Page. 2017. Visualization of Complex Medical Data Using Next-Generation Holographic Techniques. (2017). https://www.academia.edu/19597566/VISUALIZATION_OF_COMPLEX_MEDICAL_DATA_USING_NEXT-GENERATION_HOLOGRAPHIC_TECHNIQUES

[61] Andy Patrizio. 2018. What is cloud-native? The modern way to develop applications. https://www.infoworld.com/article/3281046/what-is-cloud-native-the-modern-way-to-develop-software.html. (6 2018). (Accessed on 08/27/2019).

[62] O.S. Pianykh. 2008. *Digital imaging and communications in medicine (DICOM): A practical introduction and survival guide*. 1–383 pages. https://doi.org/10.1007/978-3-540-74571-6

[63] Philip Pratt, Matthew Ives, Graham Lawton, Jonathan Simmons, Nasko Radev, Liana Spyropoulou, and Dimitri Amiras. 2018. Through the HoloLens looking glass: augmented reality for extremity reconstruction surgery using 3D vascular models with perforating vessels. *European Radiology Experimental* 2, 1 (2018), 2. https://doi.org/10.1186/s41747-017-0033-2

[64] Tom Preston-Werner. [n. d.]. Semantic Versioning 2.0.0. https://semver.org/. ([n. d.]). (Accessed on 08/26/2019).

[65] Long Qian, Alexander Barthel, Alex Johnson, Greg Osgood, Peter Kazanzides, Nassir Navab, and Bernhard Fuerst. 2017. Comparison of optical see-through head-mounted displays for surgical interventions with object-anchored 2D-display. *International Journal of Computer Assisted Radiology and Surgery* 12, 6 (June 2017), 901–910. https://doi.org/10.1007/s11548-017-1564-y

[66] Navin Ramachandran, Dean Mohamedally, and Paul Taylor. 2017. Project PEACH at UCLH: Student Projects in Healthcare Computing. *Studies in health technology and informatics* 235 (01 2017), 288–292. https://doi.org/10.3233/978-

1-61499-753-5-288

[67] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv e-prints*, Article arXiv:1505.04597 (May 2015), arXiv:1505.04597 pages. arXiv:cs.CV/1505.04597

[68] Jarek Rossignac. 2001. 3D Compression Made Simple: Edgebreaker on a Corner-Table. In *Proceedings of the International Conference on Shape Modeling & Applications (SMI '01)*. IEEE Computer Society, Washington, DC, USA, 278–. http://dl.acm.org/citation.cfm?id=882486.884089

[69] Lucas Saldanha. 2019. Tips on Logging Microservices. *Logz.io* (Aug 2019). https://logz.io/blog/logging-microservices (Accessed on 08/31/2019).

[70] Mohamed Seif, Ryosuke Umeda, Yuji Uehara, and Hiroki Higa. 2016. A Data Conversion for Medical Training System.

[71] Ian Sherr and Scott Stein. 2019. Microsoft's HoloLens 2 announced for $3,500, available to preorder now, ships later this year. https://www.cnet.com/news/microsoft-hololens-2-announced-for-3500-available-to-preorder-now-ships-later-this-year/. (02 2019). (Accessed on 08/23/2019).

[72] Benjamin Smith. 2019. *DepthVisor Surgery Room Capture Tool (Work in Progress)*. Master's thesis. University College London, London.

[73] Charles Smith. 2019. *Synthetic Image Scans Data Generator (Work in Progress)*. Master's thesis. University College London, London.

[74] Caroline M. Smith. 2018. *Medical Imaging in Mixed Reality: A holographics software pipeline*. Master's thesis. University College London, London.

[75] Marten Smits. 2014. FHIR and DICOM. https://de.slideshare.net/DevDays2014/04-b-fhir-and-dicom-marten. (11 2014). (Accessed on 08/24/2019).

[76] Scott Stein. 2019. HoloLens 2 hands-on: This feels like practical magic. https://www.cnet.com/news/hololens-2-hands-on-this-feels-like-practical-magic/. (02 2019). (Accessed on 08/23/2019).

[77] S. K. Taswell, T. Veeramacheneni, and C. Taswell. 2017. BrainWatch software for interactive exploration of brain scans in 3D virtual reality systems. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 3704–3707. https://doi.org/10.1109/EMBC.2017.8037662

[78] The Draco authors. 2019. Draco 3D Graphics Compression. https://google.github.io/draco/. (2019). (Accessed on 08/25/2019).

[79] Lucian Trestioreanu. 2018. Holographic Visualisation of Radiology Data and Automated Machine Learning-based Medical Image Segmentation. *arXiv e-prints*, Article arXiv:1808.04929 (Aug 2018), arXiv:1808.04929 pages. arXiv:cs.CV/1808.04929

[80] UCL Peach Reality. [n. d.]. PEACH Reality Overview. http://students.cs.ucl.ac.uk/2016/group39/index.html. ([n. d.]). (Accessed on 06/14/2019).

[81] Jason Walonoski, Mark Kramer, Joseph Nichols, Andre Quina, Chris Moesel, Dylan Hall, Carlton Duffett, Kudakwashe Dube, Thomas Gallagher, and Scott McLachlan. 2017. Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *Journal of the American Medical Informatics Association* 25, 3 (2017), 230–238. https://doi.org/10.1093/jamia/ocx079

[82] Stefan van der Walt, S. Chris Colbert, and Gaël Varoquaux. 2011. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering* 13, 2 (2011), 22–30. https://doi.org/10.1109/MCSE.2011.37 arXiv:https://aip.scitation.org/doi/pdf/10.1109/MCSE.2011.37

[83] G. Wang, W. Li, M. A. Zuluaga, R. Pratt, P. A. Patel, M. Aertsen, T. Doel, A. L. David, J. Deprest, S. Ourselin, and T. Vercauteren. 2018. Interactive Medical Image Segmentation Using Deep Learning With Image-Specific Fine Tuning. *IEEE Transactions on Medical Imaging* 37, 7 (July 2018), 1562–1573. https://doi.org/10.1109/TMI.2018.2791721

[84] Wavefront Technologies. [n. d.]. Wavefront Object Files (.obj) Spec. https://www.cs.utah.edu/~boulos/cs3505/obj_spec.pdf. ([n. d.]). (Accessed on 08/25/2019).

[85] Adams Wiggins. 2017. The Twelve-Factor App. https://12factor.net. (2017). (Accessed on 08/26/2019).

[86] Sue Workman. 2018. Mixed Reality: A Revolutionary Breakthrough in Teaching and Learning. https://er.educause.edu/articles/2018/7/mixed-reality-a-revolutionary-breakthrough-in-teaching-and-learning. (07 2018). (Accessed on 08/23/2019).

[87] Pengfei Xu, Shaohuai Shi, and Xiaowen Chu. 2017. Performance Evaluation of Deep Learning Tools in Docker Containers. *arXiv e-prints*, Article arXiv:1711.03386 (Nov 2017), arXiv:1711.03386 pages. arXiv:cs.DC/1711.03386

[88] Z. Yu, H. Liang, C. Fleming, and K. L. Man. 2016. An exploration of usable authentication mechanisms for virtual reality systems. In *2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. 458–460.

[89] Karol Zak. 2019. Paper Review Calls 011 – U-Net: Convolutional Networks for Biomedical Image Segmentation. https://www.youtube.com/watch?v=E1Y8HZcpm-I&t=627s. (4 2019). (Accessed on 08/29/2019).

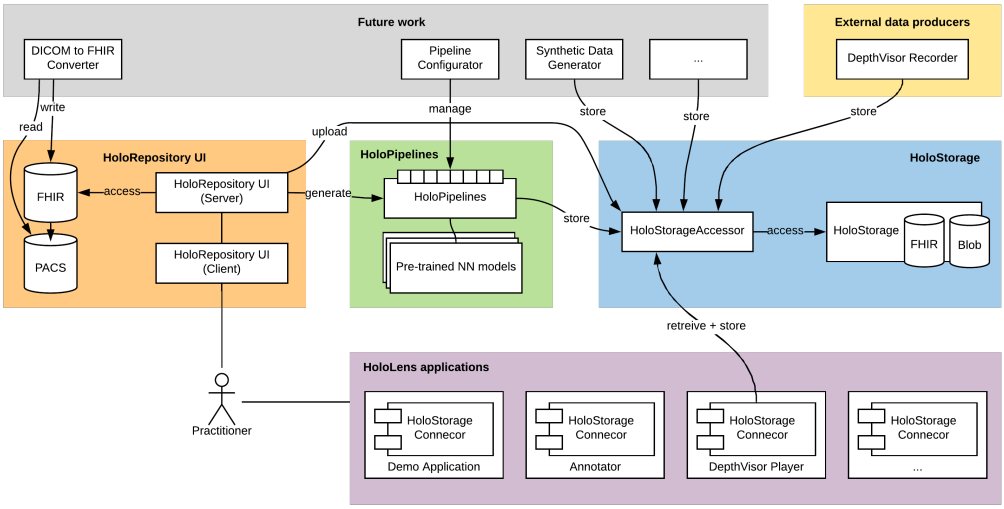
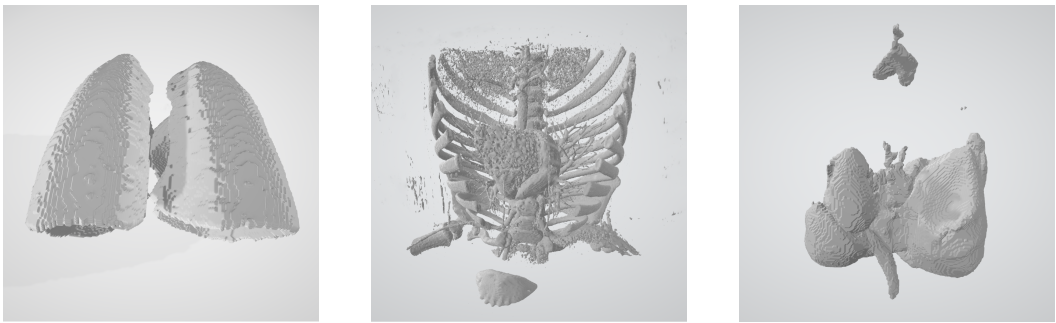
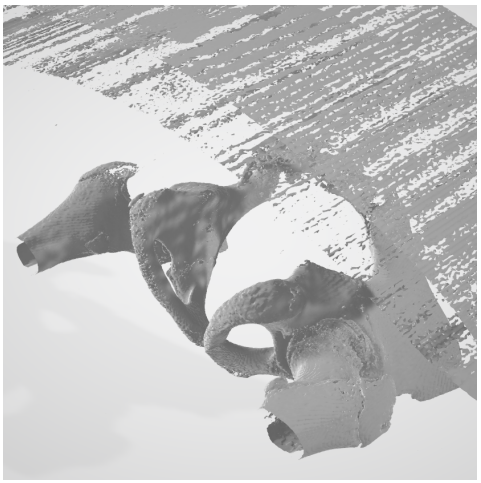


Fig. 1. Overview of the HoloRepository overall system architecture

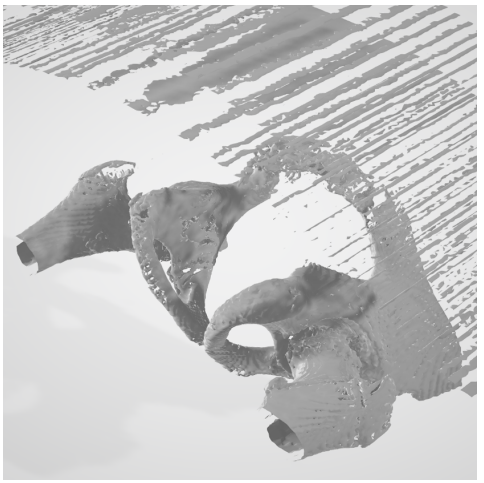


(a) Lung (b) Bone (c) Abdominal organs

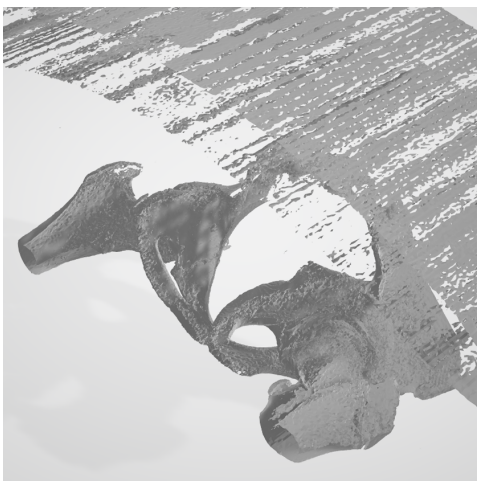
Fig. 2. 3D Output data from each of the three implemented processing pipelines



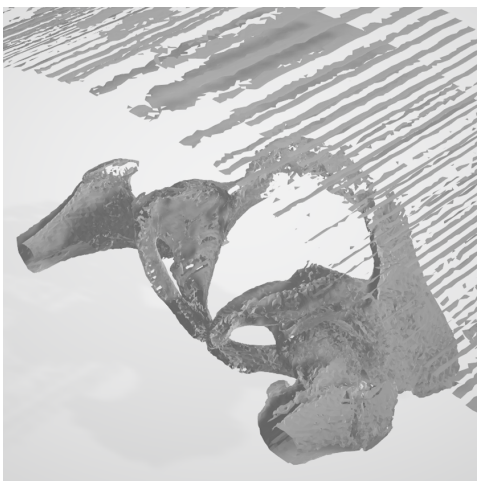
(a) Original ($r = 433x433x145$)



(b) $r_{max} = 256$

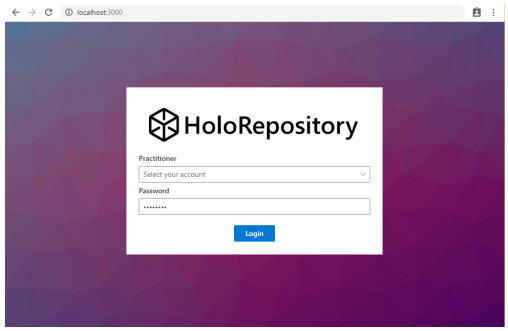


(c) Original resolution, $s = 0.5$

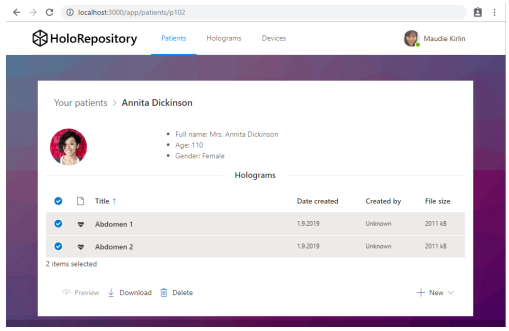


(d) $r_{max} = 256, s = 0.5$

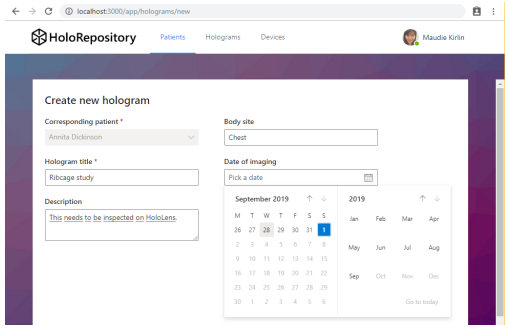
Fig. 3. Different combinations of input down-sampling (capping to r_{max}) and simplification ratios (s)



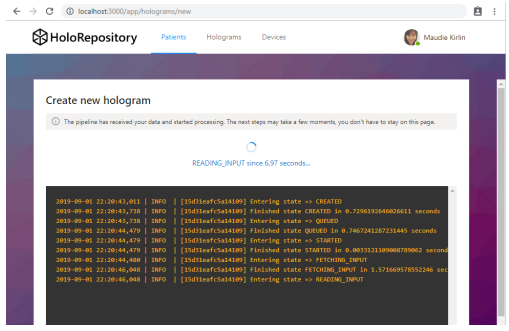
(a) Authenticate with web app as practitioner



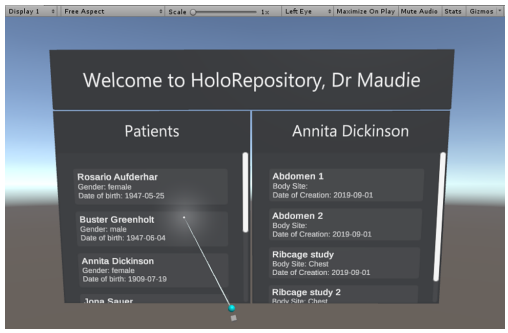
(b) Inspect authorised patients' holograms



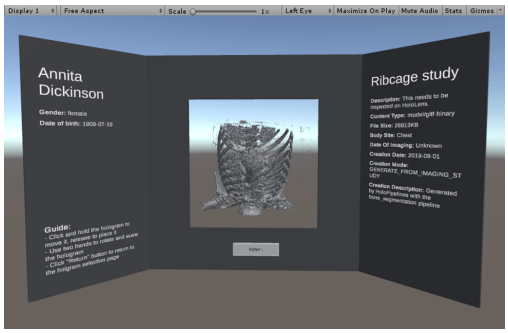
(c) Create a new hologram from an imaging study



(d) Trace progress of the model generation



(e) Inspect authorised patients on HoloLens



(f) Inspect and manipulate the new hologram

Fig. 4. Overview of a typical user journey, where a new hologram gets created in the UI and subsequently inspected on the HoloLens

Subject	Body site and condition	Files	Used with pipelines
1 y/o, M	Abdomen (normal)	normal-abdomen.zip	Bone, Abdominal organs
6.5 y/o, F	Kidney (left renal tumour)	left-renal-mass.zip	Bone
6.5 y/o, M	Pelvis (normal)	normal-pelvis-bone.zip, normal-pelvis-soft.zip	Bone
12 y/o, F	Pelvis (with a left long-standing SCFE)	left-scfe-pelvis-bone.zip, left-scfe-pelvis-soft.zip	Bone
2.5 y/o M	Chest (normal)	normal-chest-lung.zip, normal-chest-mediastinal.zip	Bone, Lung

Table 1. Overview of the DICOM imaging studies that were used as input data. Note that all of the studies obtained are CT studies; SCFE is short for Slipped capital femoral epiphysis.

	OBJ	GLB	GLB, s = 0.5	GLB, s = 0.5, Draco
r = original	size = 52.6MB $n_{\text{faces}} = 1,363,020$ $t_{\text{mesh}} = 11.68s$	size = 32.9MB $n_{\text{faces}} = 1,363,020$ $t_{\text{convert}} = 4,22s$	size = 12.3MB $n_{\text{faces}} = 681,510$ $t_{\text{simplify}} = 7.49s$	size = 1.0MB $n_{\text{faces}} = 681,510$ $t_{\text{compress}} = 2.92s$
$r_{\text{max}} = 300$	size = 22.9MB $n_{\text{faces}} = 513,790$ $t_{\text{mesh}} = 4.53s$	size = 12.4MB $n_{\text{faces}} = 513,790$ $t_{\text{convert}} = 1,93s$	size = 4.6MB $n_{\text{faces}} = 256,895$ $t_{\text{simplify}} = 2.72s$	size = 0.42MB $n_{\text{faces}} = 256,895$ $t_{\text{compress}} = 1.89s$
$r_{\text{max}} = 256$	size = 16MB $n_{\text{faces}} = 340,156$ $t_{\text{mesh}} = 2.95s$	size = 8.2MB $n_{\text{faces}} = 340,156$ $t_{\text{convert}} = 1.39s$	size = 3.1MB $n_{\text{faces}} = 170,078$ $t_{\text{simplify}} = 1.8s$	size = 0.29MB $n_{\text{faces}} = 170,078$ $t_{\text{compress}} = 1.68s$
$r_{\text{max}} = 128$	size = 3.1MB $n_{\text{faces}} = 55,976$ $t_{\text{mesh}} = 0.48s$	size = 1.0MB $n_{\text{faces}} = 55,976$ $t_{\text{convert}} = 0.34s$	size = 0.34MB $n_{\text{faces}} = 27,988$ $t_{\text{simplify}} = 0.32s$	size = 0.06MB $n_{\text{faces}} = 27,988$ $t_{\text{compress}} = 1,06s$

Table 2. Different combinations of methods applied to decrease file size. The OBJ from Figure 3 was used, with original resolution $r = 433x433x145$. Different magnitudes of down-sampling have been applied, such that the longest side is scaled down to r_{max}. n_{faces} represents the number of faces. OBJ was converted to GLB, then simplified (ratio $s = 0.5$), and finally compressed with Draco. t_{mesh}, t_{convert}, t_{simplify}, and t_{compress} represent the time taken for the marching cube algorithm, format conversion, simplification, and compression, respectively. Measured locally on an Intel Core i7 2.7 GHz machine with 16 GB 1600 MHz DDR3.

Language	Files	Lines	Blank	Comment	Code
Go	18	3,728	304	108	3,316
Typescript JSX	55	3,350	435	74	2,841
Python	47	2,624	543	157	1,924
C#	16	1,271	148	100	1,023
TypeScript	37	1,262	182	152	928
Bourne Shell	3	92	20	7	65
Sass	4	57	9	4	44
HTML	2	48	3	20	25
CSS	1	11	1	0	10
JavaScript	1	6	0	0	6
Total	184	12,449	1,645	622	10,182

Table 3. Lines of code contained in the two project repositories (as of 31/08/2019)

	Bone	Lung	Abdominal organs
Input	left-scfe-pelvis-bone.zip	normal-chest-lung.zip	normal-abdomen.zip
Size	51.6 MB	73.3 MB	62.6 MB
CREATED	0.68	0.72	0.72
QUEUED	0.73	0.77	0.72
FETCHING_INPUT	1.21	1.11	1.17
READING_INPUT	16.49	9.75	9.29
PREPROCESSING	4.09	0.10	0.08
SEGMENTATION	2.38	14.38	25.82
POSTPROCESSING	2.97	2.56	2.71
DISPATCHING	0.50	0.82	0.40
Total	28.35	30.22	40.90

Table 4. Performance measurements, broken down by how long each pipeline remains in each processing stage (in seconds). The values represent the average of five independent runs with an otherwise idle system.

Load	UI	Accessor
50 users/sec	150 ms	110 ms
75 users/sec	200 ms	950 ms
100 users/sec	1,000 ms	3,000 ms

Table 5. Average response times for a load test against a single node