Combining regions of interest with semantic segmentation to generate smart orders for transmission of medical imaging data

S. Hendrikx
Supervised by Prof. dr. A.C. Telea

May 2020

1 Introduction

In today's diagnostic procedure, imaging has elevated itself to profound relevance. The intuitive nature of spatial information means medical professionals have the means to detect specific structural anomalies at the glance of an eye. The effort to access this data should be kept to a minimum for medical practitioners in a healthcare institution. This, among other reasons like interoperability, scalability and standardization, has led to the development of the Picture Archiving and Communications System (PACS) infrastructure. This infrastructure comprises of a central archive which takes on the responsibilities of storing, compressing and allowing remote access to practitioners, as well as workstations which are configured to communicate with this central entity.

As imaging data of all modalities have significantly increased in resolution, their respective memory sizes have drastically increased alongside. This is especially the case for data with higher dimensionalities, like CT and MRI imaging. Furthermore, the availability of such data per patient has increased as well, which increases the possible number of studies a practitioner might like to review. Moreover, practitioners access the data stored on the PACS through client workstations on remote machines, with the network acting as a bottleneck for file transfer speed. Therefore, an efficient way to transmit these data is needed, as loading times increase with the size of the respective volumes being sent over the internal network to which these machines are connected.

In this thesis we propose a software driven solution to order the flow of data to the workstation according to their relevance to the practitioner, and render this information concurrently with the download from the PACS to the workstation. The solution will be driven by semantic segmentation of imaging data and user preferences with regard to their region of interest (ROI) within a volume.

1.1 Problem Context

This thesis document is written in cooperation with Alma IT Systems, located at Passeig de Gracia 11 A, 2-1, Barcelona[10], which will henceforth be referred to as either Alma or the company. Alma is the developer of Alma Workstation, the application which will be discussed in this document. and the Alma Medical Platform, which provides a standardized communication between PACS and workstations.

As expressed by the company's market research, loading times for large imaging volumes like high resolution Computer Tomography images (CT) and Magnetic Resonance Imaging (MRI) volumes have been increasing to the point where a medical professional needs several minutes to download an entire volume from the PACS to a workstation. For practitioners like radiologists, which spend a lot of their time reviewing these kinds of data, this overhead is rather cumbersome. Given the significant increase in size of 3D images over 2D images, we will be exclusively considering 3D image data.

The 3D data objects will be referred to as a 'volume'. Historically, the core functionality of workstations is the ability to view 'slices' of volumes, which correspond to all data related to an index on a certain axis or more intuitively, the cross section of a volume with respect to a specific axis. These axes are called 'Axial' for cross sections perpendicular to the foot-to-head axis, 'Coronal' for cross sections perpendicular to the chest-to-back axis and 'Sagittal' for planes perpendicular to the shoulder-to-shoulder axis. Figure 1 provides examples for the respective views. [5]

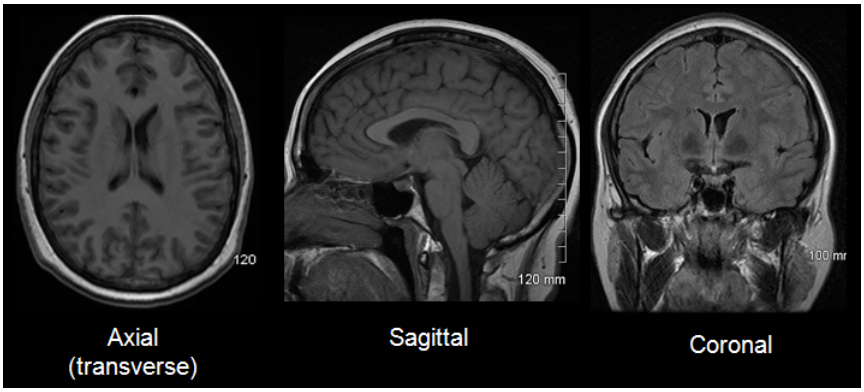

Figure 1: Axial, Coronal and Sagittal view

The specific datapoints in the volume will be referred to as either voxels or volume pixels. The values of these voxels will be referred to as luminance and it is stored in Hounsfield Units (HU)

Figure 2 shows a more in depth overview of the architecture which is currently in place.

The user uses the GUI to select the desired image, after which the worksta-

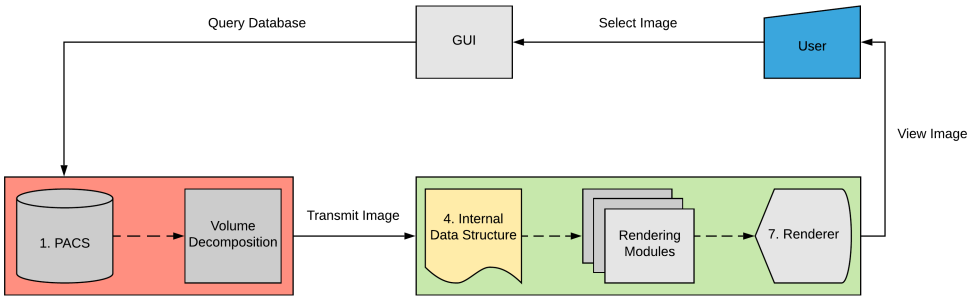

Figure 2: Current architecture

tion sends a query to the PACS, where the image is stored, in order to retrieve this image. Then, the PACS retrieves the desired image and starts preprocessing the data according to the 'Volume Decomposition' algorithm, which will be discussed in section 2.2. This generates a sequence of subvolumes, which we will henceforth refer to as packets or data packets as subvolume will not be descriptive for the solution described in this document, which are to be transmitted to the workstation.

In the current architecture, the workstation implements a concurrent downloading/rendering strategy, which allows it to render low resolution representations of the image while it is being downloaded. The received data is stored in a 3-dimensional datastructure, which the various rendering modules access in order to generate a view on screen, thereby providing the user with the information they queried.

The company has expressed that the transmission of the volume is the main bottleneck regarding speed in the pipeline.[10] Furthermore, the market this viewer is developed for consists largely of hospitals in developing countries, which may not have the best infrastructure in place, thereby compounding this problem. Therefore, this document will be answering the following question: *How can we alter the current architecture in order to provide the user with meaningful information earlier on in the downloading process.* The proposed solution is to segment the transmitted data according to the semantics of anatomy and use user preferences regarding ROIs to determine which order is best suited for the data transfer.

1.2 Thesis Structure

In chapter 2 of this thesis, an explanation of the various parts of the existing implementation will be provided, as well as the discussion of related work. Chapter 2 concludes by stating the requirements which constrain the development of the solution.
Chapter 3 will provide information on the general workflow of the project. Firstly, a number of constraints related to the development of the project will be discussed, followed by the proposition of some solutions to overcome these constraints.
Lastly, chapter 4 will discuss the architecture of the proposed solution. A discussion of various approaches will be provided, concluding with a schematic overview of the chosen architecture and the motivation behind it.

1.3 Table of contents

Contents

As most of these sections are already filled in, I will only provide a summary of the results section (Or are the summaries for the reader as well as for my own overview?)

5.1:

5.1.1

Run length encoding can in theory have a compression ratio of 1, therefore, I dedicate a small part of the results that given the type of data, there is a very small chance of this actually happening. For this I use mean and std to create an impression of the distribution, as well as min median and max values to show any outliers.

5.1.2

I want to show the relation between the actual mask size and its axis alligned bounding box. This is mainly to show that given an organ, the sparsity (i.e. how much of the mask is actually part of the organ, as opposed to zeropadding to fit the mask into a cuboid shape) falls within a predictable range. This is to show that there is a relation between these two metrics which I can use to show other relations, which are discussed more in depth in section 4.3.2.

5.2:

5.2.1:

Building on top of the results of 5.1.2, I show that given an organ, one can expect the download/post-processing time ratio to lie within some range with high confidence. If that is the case, then that means that given a threshold download speed, there is a high probability that the download time for any segment will always take more time than the post-processing time.

5.3:

Pretty much speaks for itself. I will explain ms_ssim and I will look for another metric and maybe include a more general metric like mse.

5.4 5.5:

We've discussed this many times and I'm basically generating and analyzing what the section header says. I think the best way to analyze the output is by using the area under the curve but I haven't really looked into it yet.

2 Solution Requirements

To analyze which parts of the infrastructure should be optimized it is important to have a proper understanding of its structure in order to identify bottlenecks. In this chapter, a technical overview of the various parts of the implementation of the infrastructure discussed in section 1.1 will be provided. The following figure is a duplicate of figure 2 in section 1.1

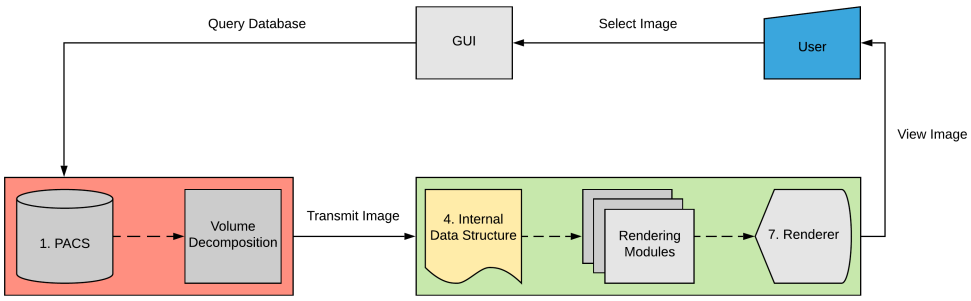

Figure 3: Current architecture

In the first subsection, the DICOM standard will be explained. Subsection 2 will provide an explanation for the volume decomposition algorithm employed by the company, subsection 3 will briefly discuss the internal representation which the various rendering modules use, and subsection 4 will discuss the available rendering modules. In subsection 5, the requirements for the solution will be provided.

2.1 DICOM standard

A lot of the standardization surrounding the storage and transfer of medical imaging is defined in what is called the DICOM standard.[9] The standard describes a set of protocols surrounding storage of an image, most importantly, the usage of headers. The DICOM header contains information on the patient, the ct scanning procedure, the position of the patient, their diagnosis and many other information related to the image. As such, the image cannot be separated from the patient it belongs to. The header in DICOM is similar to the embedded tags in JPEG images, which can hold a myriad of information about the picture. Furthermore, the pixeldata in the DICOM files can be compressed using, among others, JPEG, Lossless JPEG and Run-Length Encoding.[6][**?**][27]

DICOM also provides a set of protocols for the querying and retrieving of images over TCP/IP, as well as a standardized way of interpreting the pixel values according to various printer and monitor types. This is called the DICOM Grayscale Standardized Display Function (GSDF) and both of the concepts in this paragraph will be used to implement the lab environment, which will be discussed in section 3.3.

2.2 Volume Decomposition

The main idea of volume decomposition is to provide a subsampled preview of a volume, which gradually increases in resolution until the full volume has been downloaded to the user's machine and the volume can be displayed at full resolution. Consider a volume V of dimensions k^3 and a stopping value j. L is a list containing the generated packets of information. This algorithm divides V into two subvolumes, which we'll call A and B, using the following rule: consider a pixel x with coordinates (p, q, r). If p is even, x is part of subvolume A, else it is part of subvolume B. Then, prepend A to L and perform the same procedure on B but with regard to q, and a third time with regard to pixel r. This procedure is repeated until a volume with dimensions i^3 is obtained, with $i < j$. This generates a list of data packets for which the following properties hold:

1. $size(L[0]) == size(L[1])$
2. $size(L[i]) == .5 * size(L[i+1]) \quad \text{if} \quad i \neq 0$

An example of this procedure is visualized in figure 4:

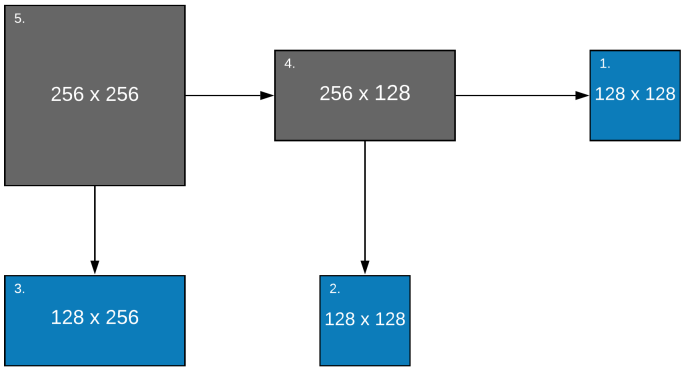

Figure 4: Volume Decomposition: The blue volumes are sent to the workstation from right to left.

The blue subvolumes, in right to left order shown in figure 4, correspond with L. When L is transmitted to the workstation, the first subvolume in L (corresponding to subvolume 1 in figure 4) is sent first, which is then upsampled to match the screen resolution. Then, the second subvolume in L (corresponding to subvolume 2 in figure 4) is transmitted and is merged to form subvolume 4. This is then upsampled to match the screen resolution and when subvolume 3 is received by the workstation, it is merged with subvolume 4 to form subvolume 5, which in this example case, is the original resolution of the volume. The user of the workstation will see subvolumes 1, 4 and 5 being rendered respectively in their screen. Since subvolume 1 contains only a fourth of the number of pixels of the original image (which is subvolume 5), it takes only a fourth of the time to render information on screen when compared to the naive implementation, which is to render the volume when it has been received in its entirety by the workstation. Note that this is only the case if the operations with respect to volume decomposition performed by the workstation are fast enough to be disregarded from a user experience point of view.

Algorithm 1 provides pseudocode for the volume decomposition procedure.

Algorithm 1: volumeDecomposition

```
   Data: A Volume V of dimensions n^3, stopping condition j
   Result: A list of decomposed subvolumes denoted by v*
 1 subvolumes = [];
 2 axis = 0 ;                    // split Volume over x, y or z axis
 3 Volume = V;
 4 while true do
 5 |   even, odd = decomposeEvenOdd(Volume, axis);
 6 |   subvolumes.append(even);
 7 |   if axis == 0 AND Volume.length(axis) ¡= j then
 8 |   |   subvolumes.append(odd);
 9 |   |   return subvolumes;
10 |   else
11 |   |   Volume = odd;
12 |   |   axis = (axis + 1) % 3
13 |   end
14 end
```

Lemma 1. *The volume decomposition algorithm divides a volume with n datapoints into subvolumes in O(n) time.*

Proof. The first line of interest is line 5, where the **decomposeEvenOdd** function is used to split the volume into two parts. As this entails nothing more than copying each voxel to a different location in memory, and knowing copying a value takes $O(1)$ time, we know that **decomposeEvenOdd** takes $m * O(1) = O(m)$ time, with m being the input size of **decomposeEvenOdd**. As all other lines of the code take O(1) time to execute, we know that **decomposeEvenOdd** will determine the upper limit of the time complexity of **volumeDecomposition**.

Furthermore, the input size will halve every iteration of the while-loop. Therefore, the total time needed to run **volumeDecomposition** can be described by

$$\sum_{i=0}^{n\_iterations} \frac{1}{2^i} n$$

hence,

$$n + n * \sum_{i=1}^{n\_iterations} \frac{1}{2^i}$$

Since we know that

$$\sum_{i=1}^{\infty} \frac{1}{2^i} = 1$$

We know that

$$n + n * \sum_{i=1}^{n\_iterations} \frac{1}{2^i} = n + n * O(1)$$

And hence,

$$n + n * O(1) = O(n)$$

A similar reasoning can be applied to show that volume recomposition, i.e. combining the subvolumes back into the original volume can be performed in $O(n)$ time as well. □

Volume decomposition works well in the sense that it allows the architecture to display information on screen faster than the naive implementation mentioned in the previous paragraph. However, the subvolumes generated by volume decomposition are subsampled uniformly. Therefore, many smaller artifacts in the volume will still not be visible until after the entire volume has been received by the workstation.

The solution described in this thesis seeks to leverage anatomical information to render specific regions of interest faster than regions of lower interest. However, this does not mean volume decomposition is thereby necessarily obsolete, as it can still be applied to the subvolumes generated by the segmentation procedure described in the architecture of the solution described in this thesis.

2.3 Internal Representation

The workstation uses an internal datastructure to represent the data it's received. This data structure is a 3D Texture and it is central to the execution of the program, as the various rendering modules employ this data structure as input for their rendering procedures. However, the dimensions of the received objects do not necessarily correspond with the dimensions of the input objects for the modules. For instance, when Volume Decomposition is used, the resolution of the received images does not match the resolution of the original image until all the data is received. Therefore, the aggregated raw data has to be upsampled, which is done by interpolation.

2.4 Rendering modules

Boxes 5. and 6. correspond to the rendering modules. In this subsection, a short description of each of the rendering modules will be provided, as well as a discussion on the relevance of the module to this project.

2.4.1 Multiplanar reconstruction

Multiplanar Reconstruction (MPR) purports to a technique which allows the user to view cross sections of a volume which are at different angles than the recorded angle a CT or MRI image, which generally corresponds to the axial plane. There are multiple variants of this technique, such as oblique MPR, which allows the user to view axis aligned slices of isotropic volumes other than the axial view in which most CT and MRI volumes are recorded. Figure 5 shows how a user interfaces with this technique.[4] Multi eliminates the need for additional

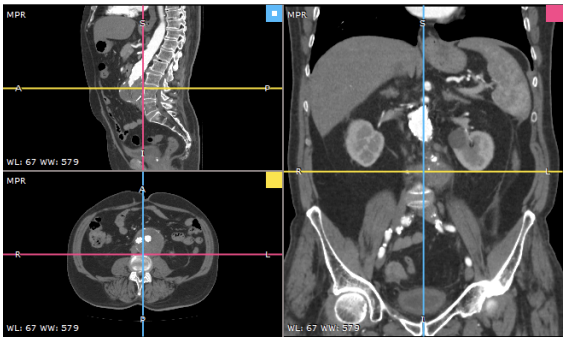

Figure 5: User interface of a multi planar reconstruction module

coronal and sagittal scans which lowers the amount of radiation received by the patient.[22] Non-oblique MPR is similar to this, with the addition that the viewable slices do not need to be axis-aligned. This is the type of MPR which is used by Alma's workstation.

2.4.2 3D Volume

The 3D volume generation procedure generates surface meshes from the received volume. It uses luminance to create non-structured isosurface meshes, which then contain the volume mesh for that specific area. The user has the ability to change the value of the isosurface and, as such can delineate certain areas of the volume according to the recorded amount of Hounsfield units.[11] The meshes are then rendered using the OpenGL library. Figure 6 provides an example of a user interface displaying 3D reconstruction.[1]. The bottom right quadrant show the constructed 3D surface mesh.

This specific procedure could benefit greatly from the implementation of the solution. Not just because of speed considerations, but by using segmentation

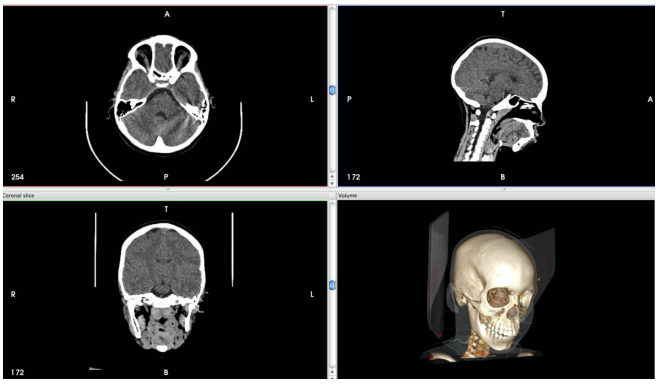

Figure 6: User interface of a 3D volume reconstruction module

masks, which are objects mapping voxels to the organs they are part of, to generate surface meshes, the workstation can generate specific organ meshes without any delineation needed by the user. [26]

2.4.3 Vascular

The vascular module is a module which is optimized for viewing images generated by angiography.[16] These images, often referred to as angiograms, are images obtained through X-ray imaging after the patient has been injected with contrast. This increases the luminance of blood vessels on the image. An often used technique is Digital Subtraction Angiography (DSA) [17], in which an image obtained without contrast is subtracted from an image obtained with contrast using a pointwise operation. An example of an angiogram obtained through DSA is provided in figure 7.[2]

Figure 7: Angiogram obtained through Digital Subtraction Analysis

As angiograms are 2D images, their memory size is an order of magnitude smaller than CT and MRI images. Therefore, querying a PACS for this type

of image does not usually suffer from the same speed constraints as querying the PACS for a CT or MRI image and, for this reason, will not benefit as much from the solution described in this thesis. Therefore, this module will be out of scope for this thesis project.

2.4.4 Nuclear

Nuclear medicine pertains to the branch of medicine which is involved with diagnosis and therapy of patients using radioactive substrates. Most relevant to this thesis is the nuclear imaging technique called scintigraphy. Scintigraphy is a medical imaging techniques in which the patient is injected with a substrate linked to a radioactive substrate. The gamma radiation emitted from the patients body is then recorded by a gamma camera. [25] This imaging technique is mostly used to detect accumulations of certain compounds in the body, most commonly glucose and oxygen in metabolically active tissues like cancers, or iodine in case of suspected hyperactivity of the thyroid gland. An example of a scintigram, an image obtained through scintigraphy, is provided by figure 8.[15]

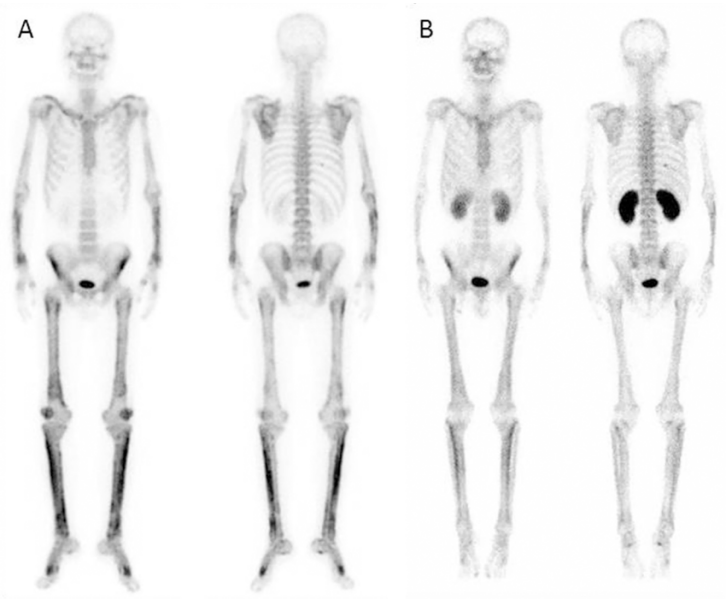

Figure 8: Example of bone scintigraphy using 99m Tc complexes

Regarding the relevance of the solution with respect to this module, a similar reasoning as in section 2.4.3 applies. the recorded images are 2 dimensional, and therefore do not suffer from the same transmission bottleneck as is the case for 3D image volumes like CT and MRI scans. Therefore, this module will be out of scope for this thesis project.

2.4.5 Dental

The Dental module is a module particularly aimed at dentists and dental surgeons. It contains functions to examine CT as well as Dental X-ray images and various functions to combine useful information from these two modalities, such as spacial alignment of the information in the respective modalities. Furthermore, there are functions which help dental surgeons plan surgery procedures and, combined with 3D printing utilities, is able to create ortheses which help identify the sites in which dental prosthetics need to be implanted. An example of a user interface of such a program is provided by figure 9. [23]

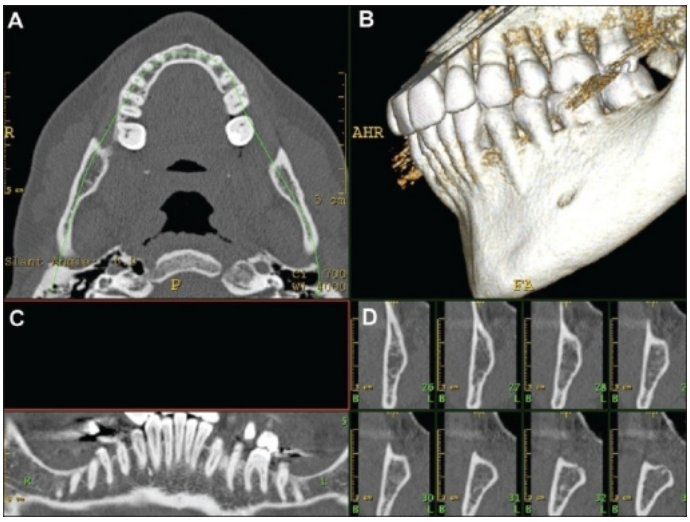

Figure 9: Example of dental imaging software

The use cases for this software, as well as the data which is being visualized by it, differs greatly from general CT and MRI imaging data. For instance, it deals with modalities in multiple dimensionalities. Therefore, given the timeframe of this thesis it seems prudent to focus on the general CT and MRI images which are relevant to the modules described in sections 2.4.1 and 2.4.2, given the fact that the input volumes for these modules are generally larger in memory size and therefore would benefit more from the proposed solution. For this reason, this module will be out of scope of this thesis project.

2.4.6 Mammo

the mammo module is specialized in viewing and analysing images generated by mammography. Mammography is the practice of examining the breast for cancer before any lumps can be felt. In general, low-energy x-ray radiation is used in order to create an image of the breast, similar to chest and bone X-ray imaging. [20] Images produces by mammography are called mammograms. Figure 10 provides an example of a mammogram. [3]

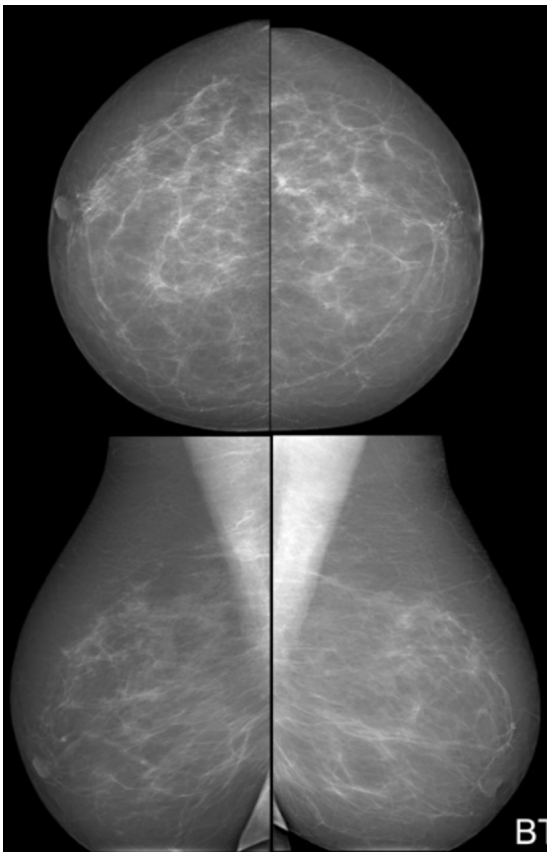

Figure 10: Example of a mammogram

Similar to the explanations provided in 2.4.3 and 2.4.4, these images are two dimensional, and, therefore, transmission of these images is currently fast enough. However, the module is also able to view images generated by tomosynthesis, which allows the generation of 3D imaging volumes by using less projection than CT imaging, thereby limiting radiation exposure.[24] However, these volumes contain only a single organ, rendering the effectiveness of the solution limited, as segmentation will only result in two classes (namely 'breast' and 'other'). Therefore, the implementation for of the solution for this module will be out of scope of this thesis

2.4.7 Ortho

The ortho module offers functionality for orthopaedic professionals and traumatologists. The module offers support such as annotation of images, planning of procedures and post-surgical tests for bone X-rays.

As we've seen before, X-ray images are two dimensional images which trans-

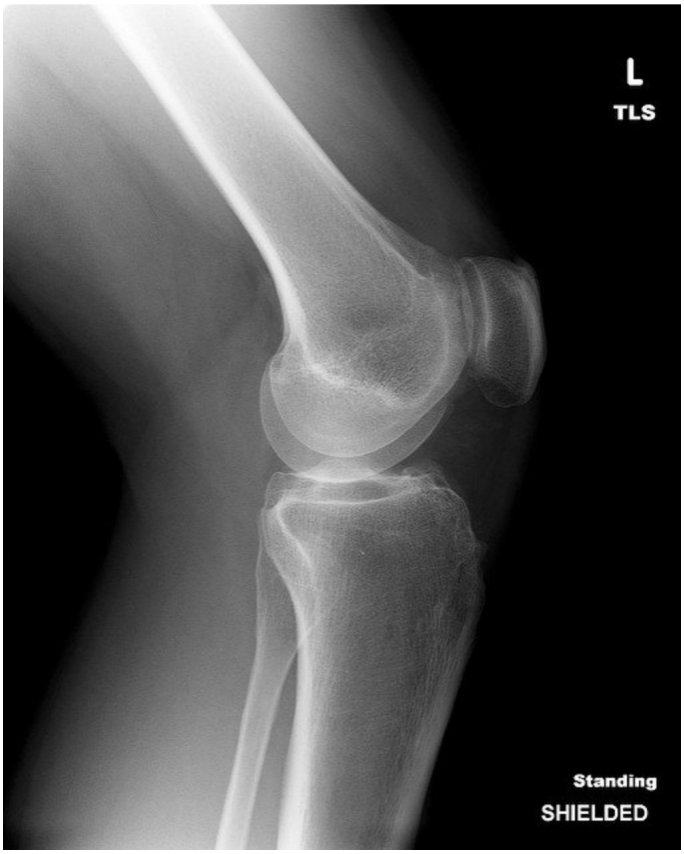

Figure 11: Example of a bone X-ray

mit fast enough in the current implementation and therefore this module will nnot be part of the scope of the thesis project.

2.5 Requirements

The requirements for the solution are defined twofold, namely for the PACS and for the viewer. The current implementation puts the workstation in the role of client and the PACS in the role of server. This is logical from a user experience point of view, given the fact that the behaviour of the workstation is directly controlled by the user. Firstly, the viewer should be able to render a meaningful image while the full volume has not yet been transferred. This is arguably the case in the current implementation. As we saw, due to volume decomposition, it's possible to render subsampled versions of the volume while the rest of the data is being transferred. However, since the resolution increases uniformly, smaller artifacts like local lesions may not be visible in an adequate resolution until the download is complete.

The proposed solution to this is to segment the data according to the semantics of anatomy and use user preferences regarding ROIs to determine which order is best suited for the data transfer. For instance, if a medical practitioner is most interested in seeing organ A, and not at all in seeing organ B, which happens to be present in the volume because of the nature of the scanning methods, it makes sense to send voxels pertaining to the regions which are classified as organ A first. Therefore, the PACS should be able to generate this segmentation. Furthermore, the PACS needs information on what the user wants to see, so the workstation should be able to provide this information. Moreover, all of this should not increase the amount of data to be sent significantly, nor the computational load on the workstation, since the current system requirements should stay the same. In summary, a number requirements should be met by the solution. These requirements are defined below and are defined for the workstation and the PACS. functional as well as non-functional requirements are defined, with the former defining what the solution is supposed to implement, and the latter defining under which constraints the requirements should be implemented. The requirements are ordered by their respective priority

Functional requirements workstation

Critical *The workstation should be able to handle the incoming datastream and incorporate it into its internal representation.*

As the PACS is sending information to the workstation, it might not send them in a straightforward order like the order generated by volume decomposition. This gets even more complicated since the user might change their ROI during the transmission. The workstation should be able to infer the location of received voxels given the information it recieves from the PACS

Critical *The workstation should be able to gather accurate user preferences regarding ROI.*

The workstation should provide the PACS with information on which parts of the volume the user would like to see, therefore it should know the user's preferences regarding ROI. For instance, it can do this by estimation from the DICOM metadata, or by allowing user interaction.

Critical *The workstation should be able to query the PACS using these preferences*

The PACS needs to know the user preferences discussed in requirement 2. Since the communication is standardized using the DICOM standard, methods need to be added in order to send this data.

Important *The workstation should be able to access masks generated by the PACS*

The PACS generates masks from the queried volume. These masks are useful to the workstation, since, as discussed in section 2.4.2, they can be used to generate surface meshes for the 3D volume rendering module. Furthermore, they can be used for other applications, for instance, given a

specific organ's luminance, the contrast settings for the renderer for that part of the volume can be optimized for that specific region.

Non-functional requirements workstation

Important *The workstation should do all of the above without significantly increasing space and time complexity*

Since Alma's workstation is predominantly used in developing countries, the solution needs to be efficient, as the systems running the workstations may not be very powerful.

Functional requirements PACS

Critical *The PACS should be able to order data according to the received user preferences*

In order to generate a meaningful data transmission order, the PACS should have a number of rules in order to exhaustively order the voxels in the volume to their respective relevance.

Marginal *The PACS should be able to make an accurate semantic segmentation of any DICOM volume.*

In order for the PACS to generate a meaningful order defined by the anatomy in the volume, the PACS should be able to accurately map voxels to the organs which they represent. Therefore, an accurate segmentation procedure should be in place. However, small errors in delineating specific organs can be tolerated as the user experience would not suffer from it greatly.

Non-functional requirements PACS

Important *The PACS should not send significantly more data than the original implementation*

This is related to requirement 1 for the workstation. The workstation should be able to incorporate the received voxels into its internal representation. However, sending three integers defining the location along with every floating point defining the luminance is extremely inefficient and would render the entire solution irrelevant, as thje aim of the project is to speed up meaningful data transfer. Therefore, a smarter strategy has to be developed.

3 Process Description

In this chapter, we will have a look at the constraints surrounding the workflow of the development of this report. In the first subsection, an outline of these constraints is provided. In the second subsection, a description of the Scrum methodology will be provided. Lastly, in subsection 3, a description of a parallel lab environment will be provided, which closely mimics the behaviour of the infrastructure implemented at Alma, but which can be used when developing off-site.

3.1 Constraints

The development of this thesis project is subject to a number of constraints, which have to be adequately addressed in order to maintain a proper workflow. Firstly, this thesis project is developed for an external company, as opposed to developing a thesis project internally at Utrecht University. This increases the number of stakeholders in the project, and can hamper progress if not accounted for accordingly.

Secondly, the body of source code developed by Alma is not open source. Therefore, easy access to a development environment is not guaranteed. There has been contact with the company to provide this code, but communication has been somewhat lackluster because of company priorities. In order to work around this, we have deployed a parallel 'stubs and drivers' infrastructure, which closely mimics the behaviour of the implementation of Alma's PACS - Workstation architecture. This will be discussed in section 3.2.

Unfortunately, during the development of this thesis, the Covid-19 pandemic ensued, which forced me to return to the Netherlands. This compounds the two problems above, as communication has to be conducted exclusively digitally. Furthermore, the fact that the code is not open source, means the alternative lab infrastructure has to be deployed in order to be able to develop the solution.

3.2 Agile/Scrum Framework

The Agile/Scrum framework has been developed as a response to the older, less flexible waterfall framework.[14][21] The waterfall framework describes a process outline consisting of multiple phases. More specifically, these phases are requirement engineering, design phase, implementation phase, verification phase and maintenance phase. The waterfall model prescribes that a phase has to be completed before continuing to the next phase.

However, given the constraints described inn the previous section, it is easy to identify a number of bottlenecks in the development process. For instance, the fact that this project is developed in cooperation with an external company, which itself defines its course by the broader implications of the market it operates in, the requirements of the project can be subject to changes over time. Furthermore, the code on which Alma's infrastructure is based is constantly under development, and therefore, the implementation specification can be subject

to change as well. Lastly, during the development of the thesis, new constraints might come to light, which could not be known during the requirement engineering phase, but which might be relevant. The Agile/Scrum framework provides a more flexible approach to the development process.

The Agile/Scrum specification consists of a number of roles and concepts. Instead of working on the entirety of the product one phase at a time, the problem owner and the developers work on smaller chunks of the problem, developing a smaller but conceptually similar versions of the final product in short 'sprints', which consist of a prioritized list of requirements, which is called the backlog, to be developed before the next sprint starts, and a deadline. In our case the sprints take between one and two weeks.

There are two types of backlog, namely, the product backlog, which defines a set of requirements for the final product. In the case of this project, that would be the set of requirements defined in section 2.4. Then, there is the sprint backlog, which takes some of the requirements of the product backlog which have the highest priority, and distills them to fit into the next sprint.

There are also a number of roles defined by the Agile/Scrum framework. Firstly, there is the problem owner, who provides an outline of the the features needed in the final implementation of the solution. In this case, that would be Alma. Secondly, there is the Scrum master, who outlines the which elements of the product backlog go into the next sprint. In the case of this thesis, that would be the main writer of this document. Lastly, there is the team. The difference between the latter two is somewhat trivial in our case, given the fact that there is only one developer involved.

This framework is suited much more for this specific instance than the waterfall model, since the project can evolve more freely given the requirements of the product owner at any given point in time. To facilitate though, there needs to be a development environment in which the solution can be iteratively built. In the following subsection, an overview of this environment will be described.

3.3 Lab Environment

Since the codebase is not accessible to the developer during the Covid-19 pandemic, a lab environment has been built which mimics the behaviour of the original architecture. Consider Figure 3:

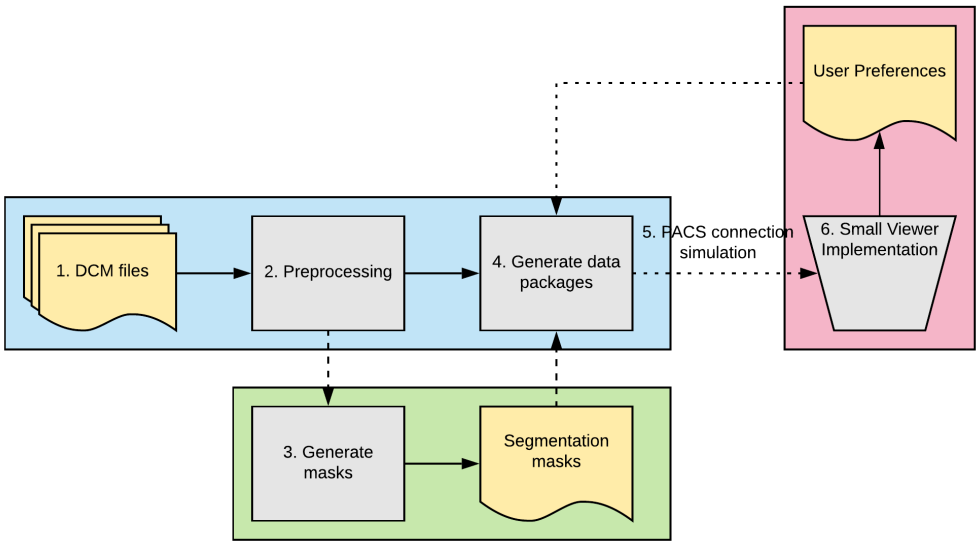

Figure 12: Lab Environment

The lab architecture consists of three main components. In figure 3, these components are color coded in blue, green and pink. The blue part of the architecture is analogous to the PACS in the infrastructure deployed by Alma. It is capable of retrieving a scan from a directory when queried by the workstation, and functions as a server for the workstation to connect to. furthermore, when queried by the workstation, it receives a set of user preferences regarding ROIs, which it can uses to generate an order over the packets of information which are to be transmitted to the workstation.

The green block in figure 3 is the segmentation engine. As Alma maintains a large body of proprietary code, they prefer to keep things modular. Therefore, the segmentation engine, which is currently not part of Alma's PACS implementation, functions as an API for the PACS, which it calls to generate segmentation masks. These masks define which voxel belongs to which organ, and they will be defined more rigorously in section 4.1.

The red box in figure 3 is the workstation. The workstation will be a stripped down version of Alma's workstation. It implements a standard ct viewer, which allows the user to scroll through a volume, as well as oblique MPR. In contrast to Alma's viewer, it does not implement surface mesh generation. However, if needed, this can be added to the infrastructure later on. Lastly, the workstation has the capability to query the PACS for a volume, as well as supply the PACS with the user preferences. This will be done by implementing the DICOM TCP/IP protocol, which is the standardized way of communications between PACS and workstation.

4 Solution architecture

Section 2.5 discussed the rough outline of the solution and its requirements. This section will focus mainly on the architecture of the solution, whereas its subsections will focus mainly on technical implementations of the various parts of the architecture. The main idea is to provide a library for the PACS as well as the workstation which handles the additional work needed for the solution. Figure 13 shows a schematic overview of the proposed infrastructure:

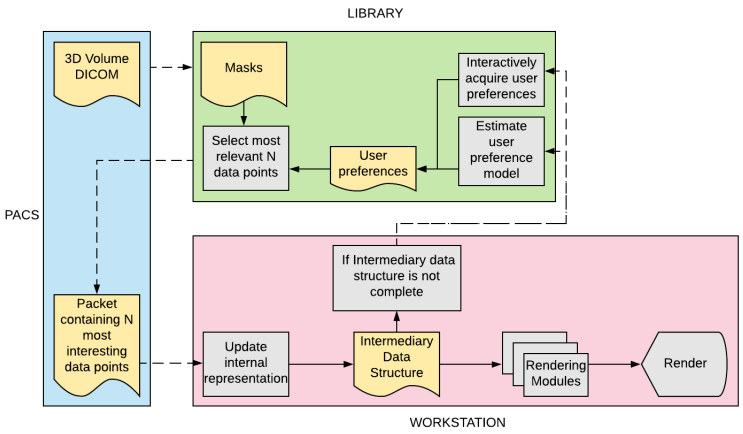

Figure 13: Solution architecture

One of the functions the PACS should have access to is a function which selects the next packet to be sent. This could also be implemented as merely a list of packets which the PACS would sequentially send after the library generated it, but by generating the packets one at a time, the viewer can change its scope during loading time without having to recompute the entire sequence of packets to be sent, thus granting a more dynamic experience for the user.

The masks will be generated by the library in an offline segmentation procedure. The use case for these masks is twofold. Firstly, the masks define the relevance of a certain area with respect to the current user preferences, secondly, they provide information which the workstation can use to infer the intended location of incoming voxels. More information will be supplied on this process in the following sections.

The workstation will query for more data in case the internal data structure still contains missing values. Similar to how the PACS queries the library for packets one at a time, so too is every packet requested separately in order to have the flexibility to grant more control to the user while the loading process takes places. An alternative solution could be to maintain a state within the PACS of the latest user preferences. However, from a consistency point of view,

it is more prudent to make sure all requests have the same format.

The user preferences regarding ROI should be estimated as well, and this can be done in a number of ways. Firstly, the library can infer this information from the DICOM headers, which, as discussed in section 2.1, contains information on the diagnosis and/or reason of the examination. From this information, the most relevant organs might be inferred. This will be discussed in section 4.2.

4.1 Segmentation

An integral part of the solution is the segmentation procedure. Segmentation can be performed by a myriad of existing classification methods, such as decision forests [13], feature extraction by hand-crafted shape descriptors [8] as well as classification using convolutional neural networks (CNNs) [7], [19]. Especially in the field of CNNs, there has been significant activity in recent years and most state of the art segmentation procedures are based on CNNs. [18] A major drawback of these techniques is the need of large amounts of annotated data, the collection of which can be a strenuous task, especially when regarding medical data. Among the options to relieve this burden are data augmentation, annotating data by hand as well as architectural improvements to lower the cost of training. [7] However, as previous research has shown the feasibility of the task of segmenting medical data, this thesis will not provide an implementation of a segmentation procedure. Furthermore, this procedure can be performed offline, and therefore has no impact on the performance at run-time.

4.1.1 Masks

The masks are a key concept of the workings of this solution, so they will be described in detail in this section. Consider a mask $m(x) -> bool$ denoting the voxels of an organ O, represented by a 3D array m of boolean values, and a volume V, represented by a 3D array of single precision floating point values. $S(x)$ denotes the shape of any array x. The masks are generated in such a way $S(m) == S(V)$ and for any voxel $vx \in V$, $m(vx) == true \iff vx \in O$. Figures and show an example of a mask defining the lungs and a mask defining bone tissue respectively.

Furthermore, the following properties hold for the masks.

Lemma 2. *There exists a set of masks M defining the organs of V, for which holds that for any voxel vx, vx is an element of exactly one region denoted by a mask $m \in M$.*

Alternatively, every voxel is part of a region defined by a mask, and no voxel can be part of more than one region defined by a mask. This ensures that voxels are never transmitted more than once or not transmitted at all.

4.2 Pre-Processing

In this subsection, the pre-processing procedures employed by the solution will be described, as well as their potential impact on the performance an complexity

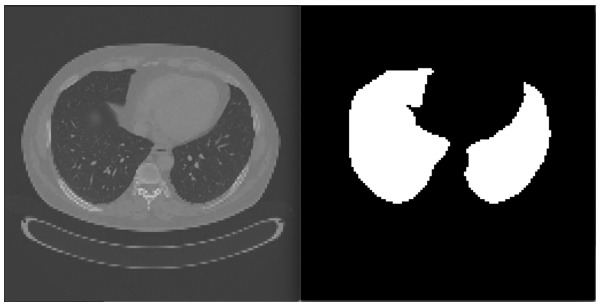

Figure 14: Mask defining the lungs: the white pixels' coordinates in the right image correspond to pixels corresponding to lung tissue in the left image.

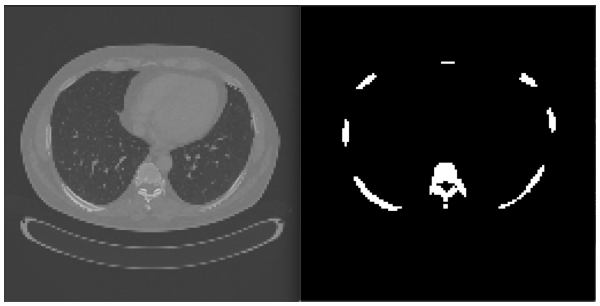

Figure 15: Mask defining the bones, the white pixels' coordinates in the right image correspond to pixels corresponding to bone tissue in the left image.

of the solution. In the first subsubsection, the mask compression procedure will be discussed. In the second subsubsection, a small discussion of the mask transmission will be provided, whereas the third subsubsection will discuss the mask extraction procedure.

4.2.1 Mask encoding

Once the masks have been acquired, the masks are encoded using a variation of run-length encoding (RLE)[27]. the Run-length encoding implementation originally uses a list of key-value combinations $rl = (k, v)$, where rl represents the a v-fold repetition of v. however, the masks used in this implementation are represented by an array of boolean values. Therefore, for the use case described in this document, k can be inferred by keeping track of the current value of k in the machine state.

The voxel values of the input volume are represented by single precision (32-bit) floating point values, and 32-bit unsigned integers have sufficient range to denote all possible run-lengths in our test set. Therefore, our implementation of RLE reduces the amount of space needed to describe a run-length form 64 bits to 32 bits, reducing the amount of data needed to describe the masks by

50%. Nevertheless, the upper bound of Run-Length encoding is O(n) space.[12] However, in the results it will be shown that for this application, RLE suffices as a compression method for segments determined by human organs. We will not discuss the time complexity of RLE in this document, as this procedure can be performed offline and therefore has no impact on the preprocessing time at run-time.

4.2.2 Segment Generation and volume decomposition

After the masks have been retrieved by the server, the server uses the masks to generate segments and data packets. Consider a mask m, denoted by a 3D array of boolean values, denoting organ o, denoted by a 3D array of floating point values, in volume V, denoted by a 3D volume of floating point values as well. A note to keep in mind when thinking about o is that a human organ does not necessarily have a rectangular shape. Therefore, o can be regarded as the axis alligned bounding box of the organ it represents, containing all indexes which are part of the organ, as well as all indexes which are not part of the organ, but which are an element of the bounding box of the organ. Furthermore, the properties described in section 4.1.1 apply here as well.

First, from m, m' is computed, such that m' has the shape of the axis alligned bounding box of all non-zero values in m. m' is then used to retrieve o. Then, volume decomposition, as described in section 2.2, is applied to both m' and o. Now, for each subvolume $o* \in o$, there exists a corresponding subvolume $m'* \in m'$. An important point to note, as discussed in this subsection, is that $o*$ does not only contain values that define the organ the user might be interested in, but also every other value in its axis alligned bounding box.

It is easy to see how this can lead to problems: as stated in section 4.1.1: for any voxel vx, vx is part of *exactly* one mask m. For example, this also counts for voxels outside of the body, which are usually stored in V to some degree as well. These voxels are thus part of a mask which can be described as "background", which has a bounding box with the same shape as V. if o contains not only the voxels corresponding to the organ it describes, but also every other voxel contained in the axis alligned bounding box, in the case of the "background" mask, the bounding box of o will simply contain all voxels in V, rendering the whole solution useless.

This is where the decomposition of m' becomes relevant: If, for every subvolume $o*$, there exists a corresponding mask $m'*$, $m'*$ can be used to function as an index map for $o*$, such that for every voxel $vx \in o*$, vx is to be selected for transmission if and only if it is defined by $m'*$ to be part of the organ it represents. the selected voxels are then stored in a one-dimensional array in lexicographical order. This, together with Lemma 2, leads to the following property:

Lemma 3. *For every voxel vx in V, vx is selected to be transmitted exactly once.*

4.2.3 Mask transmission

The mask transmission will be regarded as part of the pre-processing procedure, as there is no need to transmit masks in the naive implementation. However, throughout a single transmission, the data transfer speed from PACS to workstation will be assumed to be constant. Therefore, the transmission time will be a linear function of the memory size of the compressed masks. The actual transmission time will be discussed in the results section along with the results of the compression procedure.

4.2.4 User Preference Estimation

In order to generate a meaningful order for the data to be transmitted, the user's region of interest should be known by the PACS. This can be done either interactively, i.e. supplying GUI elements which the user can use to specify his preferences, or by estimation. One option is to infer relevance of organ classes from the diagnostic information stored in the metadata of the DICOM files. For this, a NLP classifier is the most obvious candidate, given the fact that diagnostic information is not necessarily stored in a standardized way. However, since the data to train such procedures is not available at the time of this writing, the user preference information will solely be acquired through user input.

4.3 Post-processing

The post-processing procedure takes place on the workstation side entirely, and it is responsible for the incorporation of the received data into the internal data structure of the workstation, which is used by the rendering modules.

4.3.1 Mask decoding

Unlike mask compression, mask extraction cannot be performed offline, as the workstation receives the compressed mask data at run-time. Decoding the masks takes O(n) time. [12] this will be evaluated further in the results section.

Furthermore, as we'll see in the following section, an index map for every subvolume is needed to incorporate the received data in the intermediate data structure. This follows similar reasoning to section 4.2.2, where the subvolumes acquired through volume decomposition of the relevant mask function as an index map. Therefore, volume decomposition needs to be performed on the masks extracted masks as well.

4.3.2 Segment decoding and incorporation

As discussed in segment 4.2.2, the received data is in the form of a one-dimensional array and is stored in lexicographical order. We again need an index map for each subvolume to determine which value belongs at which index. In segment 4.1.1 we have discussed the definition of a mask. However, throughout the entire transmission process, masks appear in a number of related formats such as the discussed decomposed subvolumes.

In the implementation, these representations are part of the mask object as an attribute called **subvolumes**, represented by an ordered stack of the generated subvolumes. The format defined in section 4.1.1 will be stored in the mask object as an attribute called **mask**. Furthermore, the mask object contains the current state of the received segment in an attribute defined as **current_segment**, since the state of the subvolumes attribute and the current state of the segment are closely related. Lastly, the mask object contains an attribute called **region**, which denotes the indices defining the region which is defined by the axis alligned bounding box of the mask along with it's location.

Algorithm 2: incorporate_data

Data: serialized data D, corresponding mask object m, intermediary volume V
Result: Updated intermediary volume V'

```
1 extracted_subvolume = extract_subvolume(D, m.subvolumes.pop());
2 m.current_segment = volume_recomposition(m.current_segment,
    extracted_subvolume);
3 upsampled_segment = upsample_to(m.current_segment,
    m.mask.shape);
4 V' = V[m.region][m.mask == 1] = upsampled[m.mask == 1];
5 return V';
```

Consider algorithm 2. let n be the number of elements contained by the bounding box of m and i be the number of elements in the received data.

In line 1, the received data is extracted using the corresponding mask subvolume as an index map. Since the order in which the segment subvolumes are transmitted is known by the viewer, the stack of mask subvolumes can be ordered accordingly. The relevant mask subvolume can then be popped of the stack when it's needed for data extraction. Since *extract_subvolumes* entails nothing more than replacing the every true value in the subvolume by it's corresponding value in the compressed representation, this call takes $O(n)$ time. It is not possible to bound the complexity of this line in the algorithm by i, since bounding boxes of any size can be constructed using a mask with only three elements.

Volume recomposition is performed in the second line, which combines the received subvolume with the current state of the segment into the new state of the segment. Volume recomposition returns a volume which has $2n$ values, and since section 2.2 shows that volume recomposition has a time complexity of $O(n)$, it can be concluded that line 2 of the algorithm takes $O(n)$ time as well.

The current state of the segment, which is stored in the mask object, is upsampled using zero order interpolation to fit the shape of the dimensions of the segment in V. Zero-order interpolation can be performed in $O(k)$ time, with k being the output size of the upsampling procedure. In this case, $k == n$, so therefore, line 3 takes $O(n)$ time.

Since line 4 of algorithm 2 consists of nothing more than copying n datapoints to a different place in memory, we can conclude that line 4 takes $O(n)$ time as well.

therefore, we can conclude that incorporation of a received packet of data takes $O(n)$ time, with n being the number of elements in the axis alligned bounding box of the segment in the original resolution.

Further reasoning can show that there are $O(lgn)$ subvolumes per mask, so therefore, post-processing of an entire segment will take $O(nlgn)$ time. However, depending on local factors like network speed and processing power, this might not be relevant for the following reason.

The download time of a single packet is determined by the following linear equation.

$$Td = N/s$$

With N being the size of the downloaded packet and s being the download speed. we also know that the post-processing time is $O(n)$, so we can describe the postprocessing time as follows.

$$Tp = n * c + b$$

However, there is no initialization needed for Algorithm 2 or any of its subprocedures, hence

$$Tp = n * c$$

with n being the number of elements in the axis alligned bounding box of the segment. Theoretically, n is not bounded by N, since it is possible to create

arbitrarily large axis alligned bounding boxes with $N = 3$ points belonging to a segment. A full proof for this is provided in appendix A.

However, if it is assumed that in real world data N is in fact bounded by n, this means that we can describe the post-processing time as follows.

$$Tp = N * c$$

Therefore, there exists a maximum threshold value for the download speed, which if not exceeded, ensures that the download time for a packet will always be larger than the post-processing time. In the implementation provided in this thesis, separate download and post-processing threads are implemented. Therefore, if the download speed does not exceed its threshold value, the post-processing thread will always wait for the download thread to finish downloading the next packet. In the results an evaluation between n and N will be provided to see if the assumption follows from real world data.

5 Results

In this section, we will evaluate the results of the implemented solution. To do this, the following structure will be adhered to.

In subsection 5.1 the properties of the pre-processing procedure will be discussed, whereas in subsection 5.2, the post-processing procedure will be discussed. In order to move on to the discussion of the quality metrics in subsection 5.4, an explanation for the used metrics will be provided in subsection 5.3. Then in subsections 5.4 and 5.5, the quality metrics for the individual segments as well as the entire volume will respectively be discussed.

5.1 Pre-processing

5.1.1 run-length encoding on dataset

mean and std for compressionratio: show that RLE is a suitable choice for this dataset. mean and std for time: Probably not necessary as it can be stored in its compressed format.

5.1.2 mask sparsity

mean and std: Show that masks are in general not that sparse (they probably are for bone tissue though), so the threaded implementation can be justified

5.2 Post-processing

Show that post processing is O(n)

5.2.1 Relation between download speed and post processing time

Show that for a given maximum download speed, the naive and smart implementation are equally fast

5.3 Explain quality metrics

explain $\text{ms}_s simwithpro's and cons$
$Maybe find some other related metrics?$

5.4 Quality measures for segments

Use $\text{MS}_S SIM(and maybe some other metrics) to show the value of the transmitted segments. Use the percentage of t$

5.5 Quality measures for entire image

Same control variables

References

[1] Example of 3d volume reconstruction. https://medevel.com/invesalius-3d-dicom/.

[2] Example of angiography. http://www.catawbaradiology.com/interventional-team/.

[3] Example of mammogram. http://www.rad.msu.edu/Course/RAD553/$\text{image}_l ib/body/BT14.htm$.

[4] Example of multi planar reconstruction. https://www.radiantviewer.com/dicom-viewer-manual/3d-multiplanar-reconstructions.html.

[5] Neuroradiology learning module. https://sites.google.com/site/postgraduatetraining/image-acquisition/the-basics.

[6] Ccitt recommendation t.80 (1992), common components for image compression and communication. https://www.w3.org/Graphics/JPEG/itu-t81.pdf, 9 1992.

[7] J. Hosang M. Hein B. Schiele A. Khoreva, R. Benenson. Simple does it: Weakly supervised instance and semantic segmentation. Technical report, Max Planck Institute for Informatics, Saarbrücken, Germany, 11 2016.

[8] D. Adams. *Pattern Recognition: Techniques, Technology and Applications*. San Val, 1995.

[9] National Electrical Manufacturers Association. Nema ps3 / iso 12052, digital imaging and communications in medicine (dicom) standard. https://www.dicomstandard.org/current/, 2020.

[10] H. Herrero Anton de Vez. Alma it systems streaming solution description, 4 2020.

[11] G. Frieder E. Artzy and G. Herman. The theory, design, implementation and evaluation of a three-dimensional surface detection algorithm. *Computer Graphics and Image Processing*, 15(1):1–25, 1 1981.

[12] K. G. Gray and R. S. Simpson. Upper bound on compression ratio for run-length encoding. *Proceedings of the IEEE*, 60(1):148–148, 1972.

[13] C. Rother J. Shotton, J. Winn and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23, 2009.

[14] J. Sutherland K. Schwaber. *The Scrum Guide. Definitive guide to Scrum, the rules of the game.* 11 2017.

[15] N. Kobayashi K. Tateishi M. Komatsu, M. Yasuo. Hypertrophic pulmonary osteoarthropathy in anaplastic lymphoma kinase (alk)-positive lung cancer. *Internal Medicine*, 16(54):2045–2049, 8 2015.

[16] E. Martin. *Concise Medical Dictionary*, chapter Angiography. Oxford University Press, Oxford, 9 edition, 2015.

[17] E. Martin. *Concise Medical Dictionary*, chapter Digital Subtraction Angiography. Oxford University Press, Oxford, 9 edition, 2015.

[18] D. Mwiti. A 2019 guide to semantic segmentation. https://heartbeat.fritz.ai/a-2019-guide-to-semantic-segmentation-ca8242f5a7fc, 7 2019.

[19] P. Fischer O. Ronneberger and T. Brox. U-net: Convolutional networks for biomedical image segmentation. Technical report, Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Freiburg, Germany, 5 2015.

[20] K. J. Jørgensen P. C. Gøtzsche. Screening for breast cancer with mammography. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6464778/.

[21] W. W. Royce. Managing the development of large software systems: Concepts and techniques. In *Proceedings of the 9th International Conference on Software Engineering*, ICSE '87, page 328–338, Washington, DC, USA, 1987. IEEE Computer Society Press.

[22] S. Vinci I. Salamone S. Racchiusa I. Pandolfo S. Mazziotti, F. Arceri. Role of coronal oblique reconstruction as a complement to ct study of the temporal bone: normal anatomy. *La Radiologia Medica*, 111(4):607–617, 2006.

[23] A. Zaccarella T. Re R.P. Mucelli S. Perandini, N. Faccioli. The diagnostic contribution of ct volumetric rendering techniques in routine practice. *The Indian journal of radiology imaging*, 2(20):92–97, 5 2010.

[24] I. Sechopoulos. A review of breast tomosynthesis. part ii. image reconstruction, processing and analysis, and advanced applications. *Medical Physics*, 40(1), 1 2013.

[25] J.C. Segen. *Concise Dictionary of Modern Medicine*, chapter Scintigraphy. McGraw-Hill Medical, 1 edition, 11 2005.

[26] John M. Sullivan and Ziji Wu. 3d volume mesh generation of human organs using surface geometries created from the visible human data set. In *In Proceedings of the 3rd Visible Human Project Conference, NIH*, pages 5–6, 2000.

[27] K. Kakuse S. Saba S. Ozaki K. Itoh T. Tsukiyama, Y. Kondo. Method and system for data compression and restoration. https://patentimages.storage.googleapis.com/53/aa/40/a8c81c61d4eaac/US4586027.pdf.