Combining regions of interest with segmentation for smart transmission of medical imaging data

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

May 2020

Abstract

The PACS / workstation infrastructure employed at hospitals around the world allows for accessibility of medical imaging data across a hospital infrastructure. However, with increaseing memory size of the average CT and MRI scan [13], medical practitioners spend increasingly more time waiting for data to be downloaded to their workstation. This thesis documents the workings and development of a smart transmission scheme which allows for prioritized data transfer according to the user's region of interest combined with semantic segmentation of medical imaging volumes.

The solution has been shown to be fast enough using quantitative results and has been shown to provide better image quality while not transmitting more data using a number of qualitative metrics like MS_SSIM[32].

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.[13] This is especially the case for data with higher dimensionalities, like CT and MRI imaging.[13] 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.[8] Moreover, practitioners access the data stored on the PACS through client workstations on remote machines, with the network possibly 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

The research serving as an inspiration to the solution described in this thesis document is done in cooperation with Alma IT Systems, located at Passeig de Gracia 11 A, 2-1, Barcelona, 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. [6]

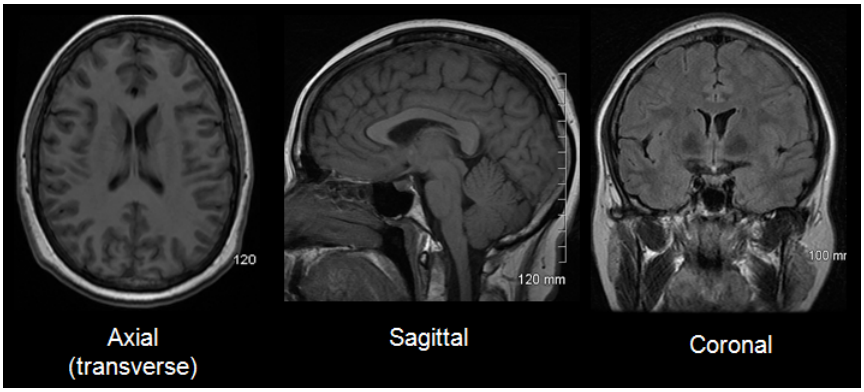

Figure 1: Examples of axial, coronal and sagittal cross-sections of the brain

The specific datapoints in the volume will be referred to as either voxels or volume pixels. The values of these voxels represent the tissue's radiodensity it is stored in Hounsfield Units (HU)

Figure 3 shows a more in depth overview of the architecture which is currently in place.

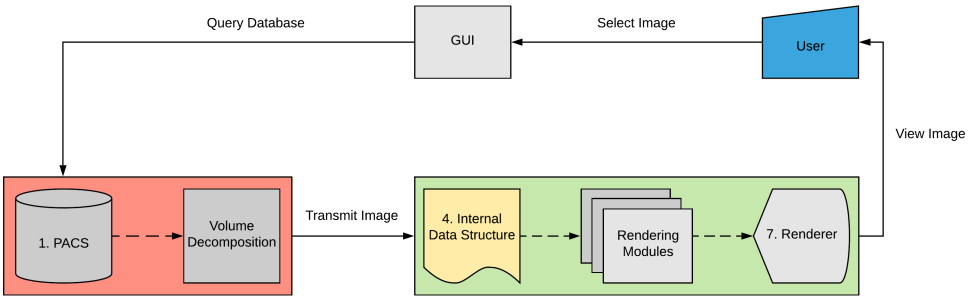

Figure 2: Current architecture employed by Alma's client-server infrastructure for transferring medical imaging data

The user uses the GUI to select the desired image, after which the workstation 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. 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 focus on developing a solution which makes the process of transmission more efficient. This document will be answering the following question:

Can the current architecture be changed 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 first, and use user preferences regarding ROIs to determine which order is best suited for the data transfer. More intuitively, alongside querying the PACS for a given volume, the user will provide which part of the volume is of highest interest to him or her. In this solution, we will focus on a set of organs, of which the location will be defined by masks. If the user is interested in, for instance, the lungs, the lungs will be transmitted before the rest of the volume is, thereby decreasing the amount of time needed to display relevant information on the screen of the medical practitioner.

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.

Since this thesis project had to be developed under a number of constraints, chapter 3 will provide information on the general workflow of the project. Firstly, constraints related to the development of the project will be discussed, followed by the proposition of some solutions to overcome these constraints.

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. Furthermore, some of the theory underlying the workings of the selected solution will be provided.

In chapter 5, quantitative as well as qualitative results will be discussed. Quantitative results will focus on establishing some relations which are necessary to establish some relations which are not covered by the theory discussed in chapter 4, whereas the qualitative results will focus on the quality of the output generated by the solution.

Lastly in chapter 6, the results will be discussed in the context of the research question and the requirements discussed in chapter 2. Furthermore, limitations of the thesis as well as some proposed solutions and avenues of research to address these shortcomings will be outlined.

Contents

2 Background

In this chapter, a technical overview of the various parts of the implementation of the infrastructure discussed in section 1.1 will be provided.

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.

The following figure is a duplicate of figure 3 in section 1.1

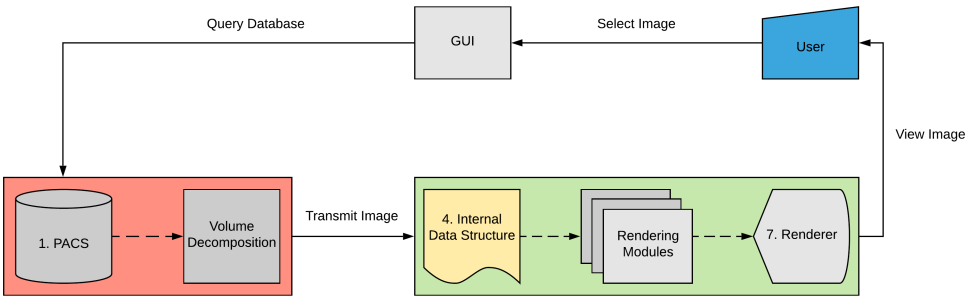

Figure 3: Schematic overview of the PACS/viewer architecture built by Alma

In the coming sections, the following will be explained. Firstly, a list of preliminary terms and nomenclature will be explained. This list serves as a point of reference for a lot of the terminology used in this thesis.

In section 2.2, the DICOM standard will be explained. This is relevant because the implementation used in Alma's PACS/viewer architecture uses the DICOM standard as its transmission protocol. In this section, which parts of the standard are relevant for the solution will be discussed.

Section 2.3 will provide an explanation for the volume decomposition algorithm employed by the company. The algorithm will play a central role in the provided solution, and therefore, it is relevant to understand its workings.

Section 2.4 will briefly discuss the internal representation which the various rendering modules use. As all rendering modules use the same input format, this will be the output of the solution provided in this thesis.

Section 2.5 will discuss the available rendering modules. As the solution is geared towards medical imaging types with large memory sizes, we will evaluate which rendering modules are impacted by the solution, and why.

Lastly, in section 2.6, the requirements for the solution will be provided. These re-

quirements will guide the implementation of the solution, and are therefore relevant to keep in mind when evaluating the results.

2.1 Preliminaries

Throughout the following sections, a some terms will be used to denote certain entities or operations. In order to avoid confusion, a number of terms will be defined in this section.

- **Volume**
  A volume $V \in \mathbb{R}^3$ refers to a 3-dimensional cuboid object storing floating point numerical data. For instance, V stores a floating point x, where $x \in \mathbb{R}$ at every index (i, j, k), where $i, j, k \in \mathbb{N}$ and (i, j, k) is in the domain of V

- **Subvolume**
  A volume V' is a subvolume of volume V if $\forall x(x \in V' \rightarrow x \in V)$

- **Mask**
  A mask $M \in \{0, 1\}^3$ is in all but one aspect the same as a volume, with the difference being that the voxels of a mask only store binary values. The mask denotes for a given voxel (i, j, k) whether that index does or does not belong to the organ which is denoted by the mask. More information on masks can be found in section 4.1.1

- **PACS**
  The network infrastructure which will be replicated in this thesis is based on the client-server model. In this thesis, the server storing the image data will be referred to as PACS (Picture Archiving and Communication System), as this is the type of server which is used in the hospital infrastructures employed by Alma

- **Workstation**
  On the client side of the infrastructure resides the workstation or alternatively, the viewer. The workstation queries the PACS for data and provides tools to view and evaluate the received data. However, much of these tools which are found in Alma's workstation are not available in the implementation provided by this thesis. Nevertheless, for consistency, the PACS/workstation terminology will be used throughout this thesis.

- **Grayscale**
  A grayscale image or grayscale volume V, in the context of this thesis, stores a single value $x \in \mathbb{R}$ at each of its indices. This is represented on the screen by a hue of gray. In contrast with grayscale images, there are RGB-images which store a value $x \in \mathbb{R}^3$ at each of its indices. In this thesis, only grayscale images will be used.

- **Upsampling**

Upsampling refers to the operation by which a volume V with dimensions x, y, z is transformed into a volume V' with dimensions x', y', z', for which holds that $x' > x \vee y' > y \vee z' > z$ and $V \subseteq V'$. In this thesis, nearest neighbour interpolation is used for upsampling, but any nth-order deterministic interpolation method can be applied in the context of this work.

- **Downsampling**
  Downsampling refers to the operation by which a volume V with dimensions x, y, z is transformed into a volume V' with dimensions x', y', z', for which holds that $x' < x \vee y' < y \vee z' < z$ and $V' \subseteq V$.

- **Transmission**
  The process by which one machine can retrieve information from another will be referred to as transmission in this thesis. Consider two machines A and B. If a data object x stored in machine A is transmitted to machine B, both machine A and B have access to this data from their own memory from that point forward.

- **Architecture**
  The architecture refers to the entire configuration of PACS and viewer. This term will be used to denote what capabilities a configuration has. For instance, the naive architecture is a configuration of PACS and viewer which supports volume decomposition, but no data prioritization.

- **Implementation**
  Where the architecture refers to the conceptual configuration of a PACS and viewer, the implementation is the actual instance of this configuration. However, since these will be roughly the same throughout this thesis, these terms will be used interchangeably.

- **Online/Offline**
  In the context of this thesis, the terms online and offline will be used to designate which parts of the solution are performed at runtime (online), and which parts can be performed before the query is performed (offline).

- **Sparsity**
  In chapter 4, an explanation will be provided of the relationship between the number of voxels in a mask m defining an organ o and the number of voxels in the axis aligned bounding box of o. The ratio of number of voxels in o and the number of voxels in its bounding box is what we refer to as sparsity. Intuitively, for a mask m with high sparsity, a low number of voxels in the axis alligned bounding box of m define o.

- **User region of interest** In order to determine which parts of a volume are most relevant to the user, the user needs to provide which organ/organ systems are most relevant to their research. Therefore, user region of interest is a central topic in this thesis. In this thesis, on a high level, user region of

interest is denoted by the set of organs which are most relevant, and on a low level, it refers to the segment or mask which represents that organ.

2.2 DICOM standard

A lot of the standardization surrounding the storage and transfer of medical imaging is defined in what is called the DICOM standard [10]. The standard describes a set of protocols and standards surrounding storage of an image, for instance, the usage of headers. The DICOM header contains information on the patient, the 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.[7][29]

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). These concepts will not be applied in the lab environment, which will be discussed in section 3.3, as it is trivial to the solution. However, this thesis will touch on how the solution can be implemented using the DICOM standard.

2.3 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 $V \in \mathbb{R}^3$ of dimensions ${d_{start}}^3$ and a stopping value d_{end}, where $d_{start}, d_{end} \in \mathbb{I}$. P is a list containing the generated packets of information.

This algorithm divides V into two subvolumes, which we'll call V_1' and V_2', using the following rule: consider a pixel $x \in V$ with coordinates (p, q, r). If p is even, x is part of subvolume V_1', else it is part of subvolume V_2'. Then, prepend V_1' to P and perform the same procedure on V_2' but with regard to q, and a third time with regard to pixel r. This procedure is repeated until a volume with dimensions d^3 is obtained, with $d < d_{end}$. This generates a list of data packets for which the following properties hold:

1. $size(P[0]) \;=\; size(P[1])$
2. $size(P[i]) \;=\; \frac{1}{2} \cdot size(P[i+1]) \quad \text{if} \quad i \neq 0$

Where $size(P)$ refers to the number of indices in P

Each subvolume in P represents a downsampled version of the input volume V.

An example of this procedure is schematically visualized in figure 4. Furthermore, figures 5 and 6 respectively show more intuitive, real-world examples of the volume decomposition procedure, as well as its inverse operation, volume recomposition.

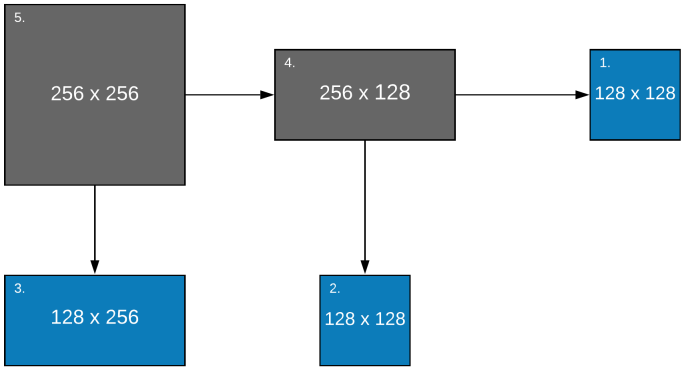

Figure 4: Volume Decomposition: The blue volumes are sent to the workstation from right to left.

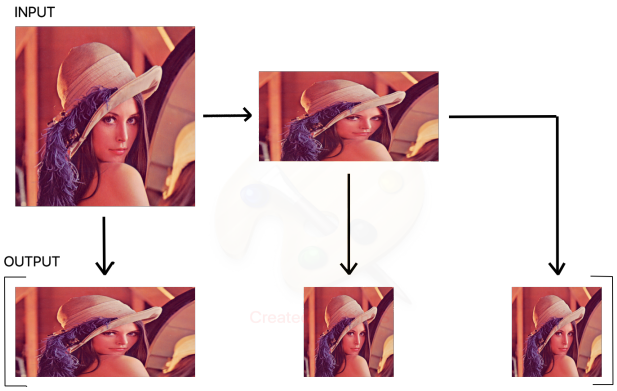

Figure 5: Volume Decomposition: An example with images, the bottom represents the output of the procedure

The blue subvolumes, in right to left order shown in figure 4, correspond with P. When P is transmitted to the workstation, the first subvolume in P (corresponding to subvolume 1 in figure 4) is sent first, which is then upsampled to match the screen resolution. Then, the second subvolume in P (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

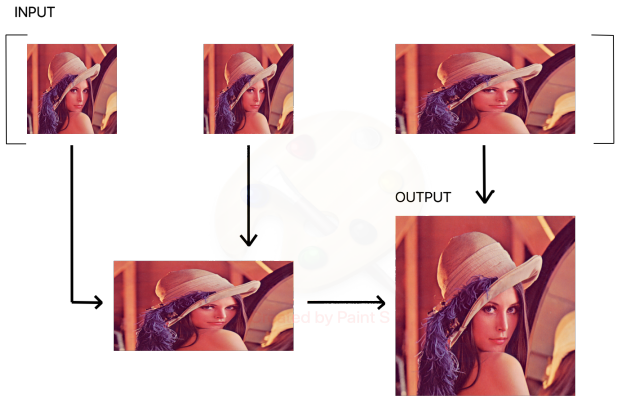

Figure 6: Volume Recomposition: An example of the reverse operation of Volume Decomposition.

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  v* ← [];
2  axis ← 0 ;                          // Divide Volume over x, y or z axis
3  while true do
4      even, odd ← decomposeEvenOdd(V, axis);
5      v*.append(even);
6      if axis = 0  ∧  V.length(axis) <= j then
7          v*.append(odd);
8          return v*;
9      else
10         V ← odd;
11         axis ← (axis + 1) % 3
12     end
13 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 details 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.

2.4 Internal Representation

The workstation uses an internal datastructure to store the accumulated data it's received. This data structure is a 3D volume 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.

2.5 Rendering modules

The workstation employs a number of rendering modules to display the data in various formats. In this section, 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. In general, the rendering modules will not play a significant role in the solution itself. However, the various rendering modules are used for different modalities, and therefore are differently impacted by the solution.

2.5.1 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 7 shows how a user interfaces with this technique.[5] Multi eliminates the need for additional coronal and sagittal

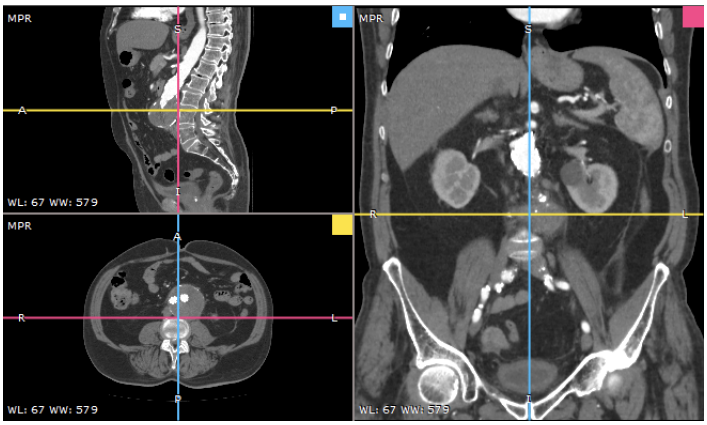

Figure 7: User interface of a multi planar reconstruction module. Top left shows an axial view, bottom left shows a sagittal view and the right half shows a coronal view. All three images show the same volume

scans which lowers the amount of radiation received by the patient.[21] Non-oblique MPR is similar to this, with the addition that the viewable slices do not need to be axis-aligned. Non oblique MPR is the type of MPR which is used by Alma's workstation.

2.5.2 3D Volume

The 3D volume generation procedure generates surface meshes from the received volume. It takes a threshold value as input 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.[9] The meshes are then rendered using the OpenGL library. Figure 8 provides an example of a user interface displaying 3D reconstruction.[2]. The bottom right quadrant show the constructed 3D surface mesh.

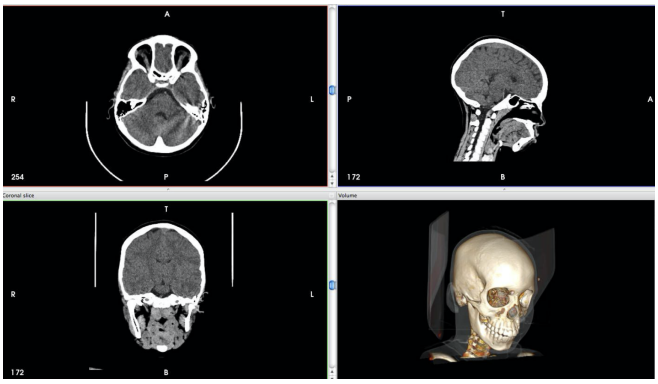

Figure 8: User interface of a 3D volume reconstruction module. The bottom right image shows the generated isosurface, while the other three images show different cross-sections of the volume used to generate the surface mesh in this image.

This specific procedure could benefit greatly from the implementation of the solution. Not just because of speed considerations, but by using segmentation 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, as opposed to isosurface meshes, where the user has to set a threshold value for the voxel values. [28]

2.5.3 Vascular module

The vascular module is a module which is optimized for viewing images generated by angiography.[19] These images, often referred to as angiograms, are images obtained through X-ray imaging after the patient has been injected with contrast solution. This increases the luminance of blood vessels on the image. An often used technique is Digital Subtraction Angiography (DSA) [20], in which an image obtained without contrast is subtracted from an image obtained with contrast using a pointwise subtraction. An example of an angiogram obtained through DSA is provided in figure 9.[3]

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

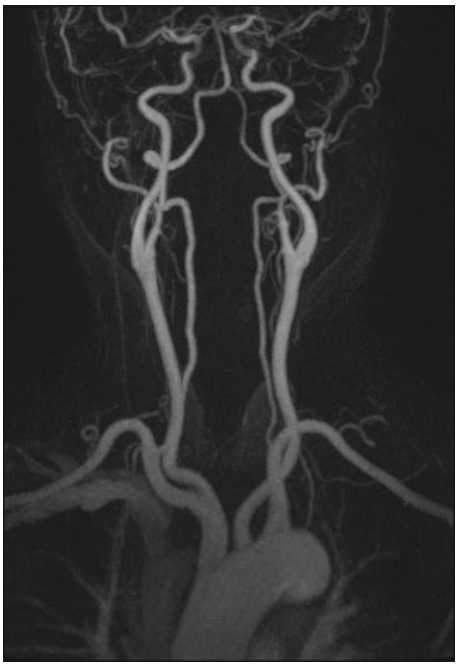

Figure 9: Angiogram obtained through Digital Subtraction Analysis

in this thesis. Therefore, this module will be out of scope for this thesis project.

2.5.4 Nuclear module

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. [27] 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 10.[18]

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.

2.5.5 Dental module

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

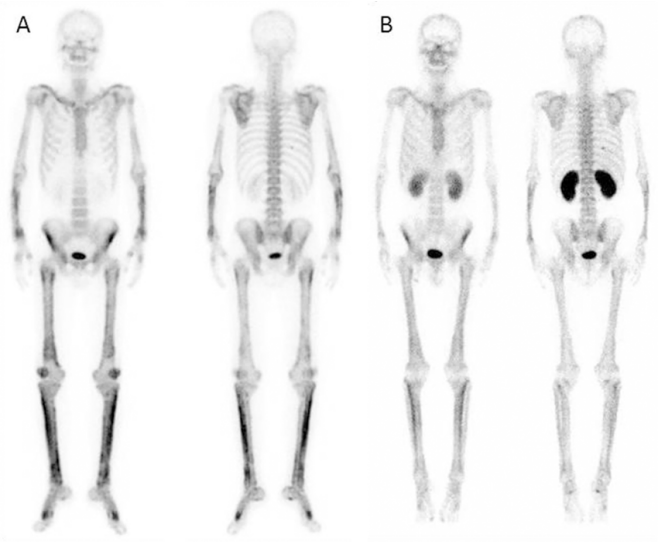

Figure 10: Example of bone scintigraphy using 99m Tc complexes

functions which help dental surgeons plan surgery procedures and, combined with 3D printing utilities, Alma's software 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 11. [25]

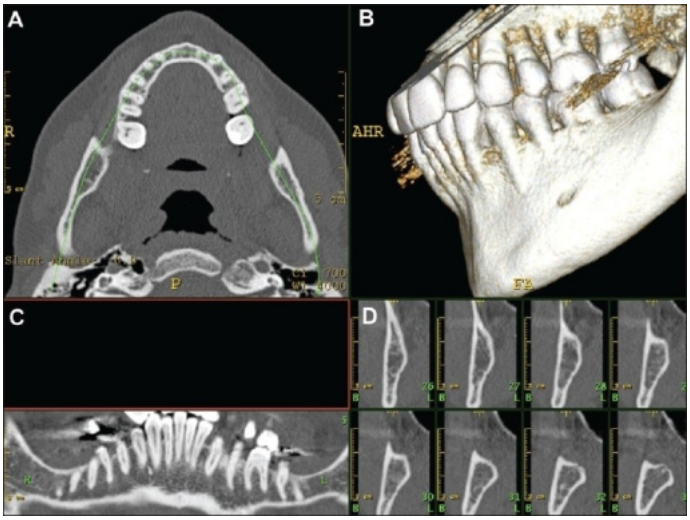

Figure 11: Example of dental imaging software. Image A shows a slice of the input volume. Image B shows an isosurface mesh generated for this volume. Image C shows a related dental X-ray image. Image D, shows an example of some dental analysis software

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. For example, it use can be use to analyse 3D volumes as well as 2D dental X-rays. 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.

2.5.6 Mammography module

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. [14] Images produces by mammography are called mammograms. Figure 12 provides an example of a mammogram. [4]

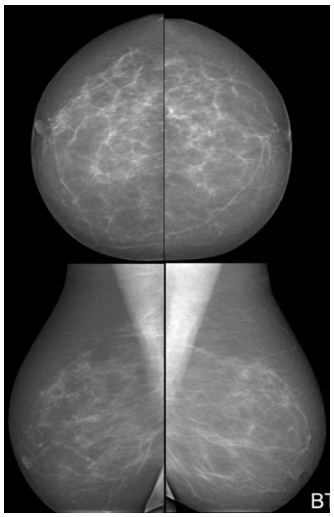

Figure 12: Example of a mammogram

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.[26] 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

2.5.7 Ortho module

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 transmit fast enough in the current implementation and therefore this module will not be part of

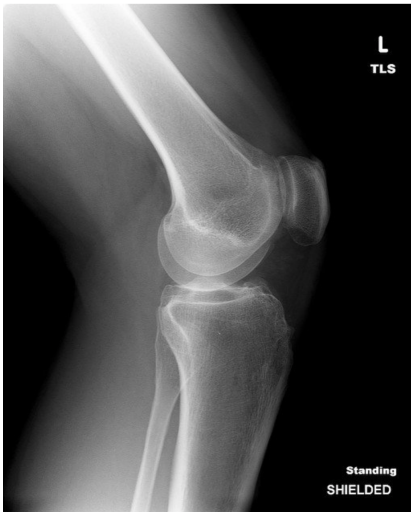

Figure 13: Example of a bone X-ray

the scope of the thesis project.

2.6 Requirements

Now that the various visualization modules have been introduced and their relevance has been defined, the requirements for the solution to be provided can be defined more thoroughly.

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 details 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 require-

ments 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 priority, which are defined as follows: Critical priority means that the requirement has to be implemented in the eventual solution. If this is not the case, the implementation is not a solution to the defined problem. Important means that the requirement will greatly improve the functionality of the solution, and therefore the aim should be to preferably have all 'important' requirements implemented. Lastly, marginal priority is a bit contradictory, as it defines requirements which are nice to have, but not necessarily required. Nevertheless, for conciseness, they will be listed here in the requirements section as marginal, as opposed to having their own section dedicated to them.

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 the aim of the project is to speed up meaningful data transfer. Therefore, a smarter strategy has to be developed.

3 Development Process

In this chapter, we will have a look at the constraints surrounding the workflow of the development of this report. A number of decisions had to be made due to constraints in either the development process or the communication with experts at the company. These constraints had to be dealt with and are reflected in development and implementation choices. Therefore, in this chapter, these constraint will be outlined as well as the impact these constraints have had on the development process.

In the section 3.1, an outline of the constraints is provided. In section 3.2, a description of the Scrum methodology will be provided. Lastly, in section 3.3, 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

As mentioned in the previous section, 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. these constraints arer provided as a list below.

- **Communication with stakeholders**
  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.

- **Access to source code**
  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.

- **Working remotely**
  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.

- **Source code documentation**
  Since the source code to Alma's software is not open source, the documentation isn't either. Furthermore, the transfer of knowledge therefore is partly done

by technical briefings, which I was not able to attend because of restrictions related to the Covid-19 pandemic, as described in the previous bulletpoint.

- **Access to knowledge of previous developers**
  Related to the fact that work had to be conducted remotely, accessing information of previous developers has been hard and lackluster. Therefore, a more generalized approach to the implementation had to be employed. This will be described in more detail in section 3.3

- **Developing for a product under development**
  Alma's PACS and workstation software is constantly under development. Again, combined with the fact that work and communication has to be conducted remotely, this makes implementation more complex. A more generalized, proof of concept type of approach is therefore more suited to this development process. This approach will be outlined in detail in section 3.3

3.2 Agile/Scrum Framework

The Agile/Scrum framework has been developed as a response to the older, less flexible waterfall framework.[16][24] 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 2.6. 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 this though, there needs to be a development environment in which the solution can be iteratively built. In the following section, 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 14:

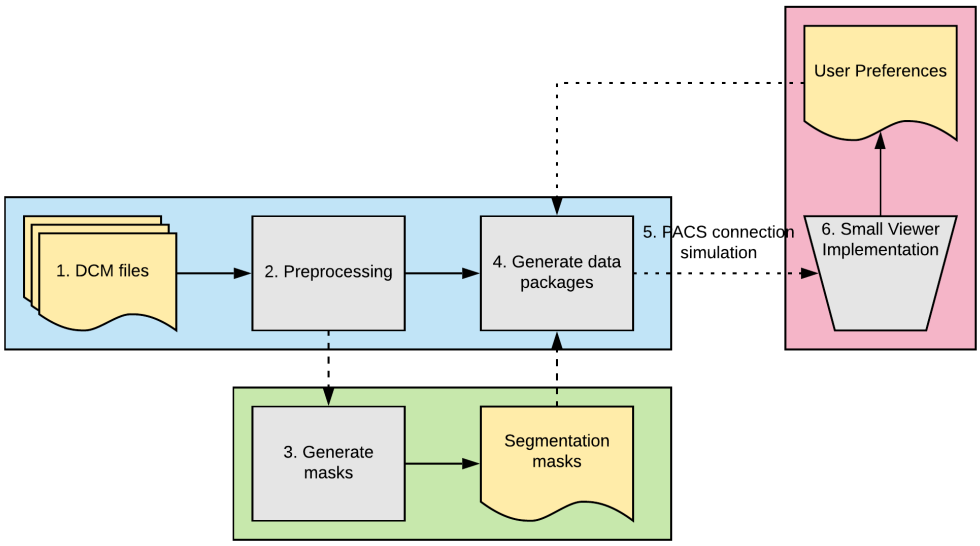

Figure 14: Lab Environment

The lab architecture consists of three main components. In figure 14, 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 14 is the preprocessing engine. As Alma maintains a large body of proprietary code, they prefer to keep things modular. Therefore, the preprocessing 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 4.1.1.

The red box in figure 14 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 2.6 discussed the rough outline of the solution and its requirements. This chapter will focus mainly on the architecture of the solution, whereas its sections 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 15 shows a schematic overview of the proposed infrastructure:

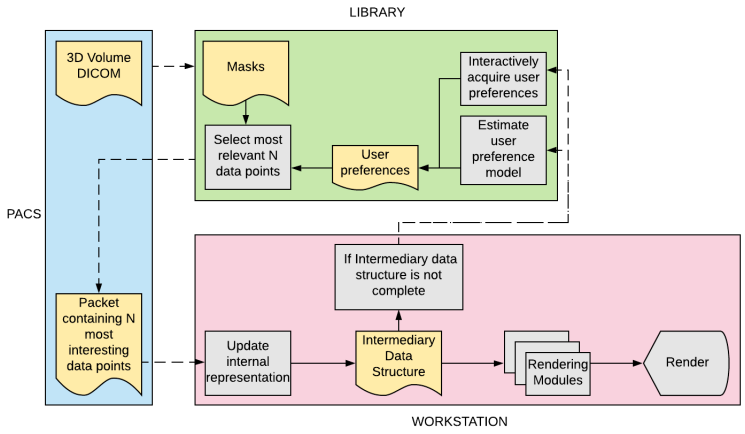

Figure 15: Proposed solution architecture

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 section 4.3.2.

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.

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 [15], feature extraction by hand-crafted shape descriptors [1] as well as classification using convolutional neural networks (CNNs) [17], [23].

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. [22] 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. [17]

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 of the solution 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 $m(x) \rightarrow bool$ 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 $v \in V$, $v \in m \iff v \in O$. Figures 16 and 17 show an example of a mask defining the lungs and a mask defining bone tissue respectively.

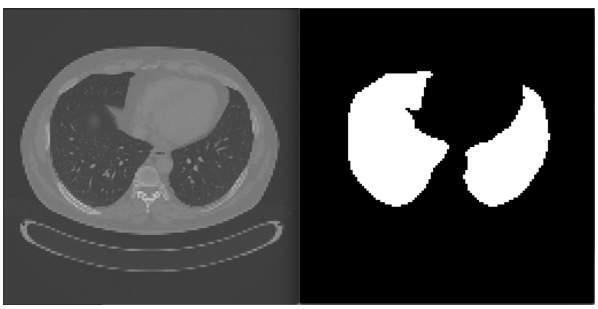

Figure 16: Mask defining the lungs: the white pixels' coordinates in the right image correspond to pixels corresponding to lung tissue in the left image.

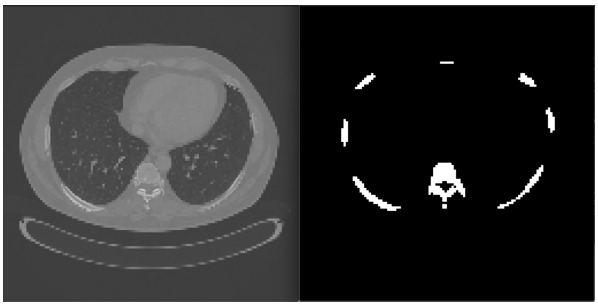

Figure 17: Mask defining the bones, the white pixels' coordinates in the right image correspond to pixels corresponding to bone tissue in the left image.

The masks used to generate these examples are provided by the CT-ORG project[11]. This datasource will be discussed in more depth in section 5.1.

Furthermore, lemma 2 holds for the masks used in the solution.

Lemma 2. *For the set of masks M defining the organs of V holds that for any voxel v, v 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 section, the pre-processing procedures employed by the solution will be described, as well as their potential impact on the performance an complexity of the solution. In section 4.2.1, the mask compression procedure will be discussed. In section 4.2.2, an explanation of the procedure which generates the data packets will be provided, whereas section 4.2.3 will discuss the mask transmission. Section 4.2.4 will touch on how the solution will gather information on the user's region of interest.

4.2.1 Mask encoding

Once the masks have been acquired, the masks are encoded using a variation of run-length encoding (RLE)[29]. the Run-length encoding implementation originally uses a list of key-value combinations $rl = (v, n)$, where rl represents the a n-fold repetition of v. However, the masks used in this implementation are represented by a volume of binary values. Therefore, for the use case described in this document, v can be inferred by keeping track of the current value of v 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 from 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.[29] 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. However, an analysis of the compression ratio will be provided, as the non-functional requirement for the PACS states that the solution should not send significantly more data than the naive implementation.

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 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 lexicographically ordered list of all voxels which are part of the organ it represents. Furthermore, let o' be the axis alligned bounding box of o, containing all indexes which are part of o, as well as all indexes which are not part of the organ, but which are an element of the axis alligned 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, this process is shown in figure 18.

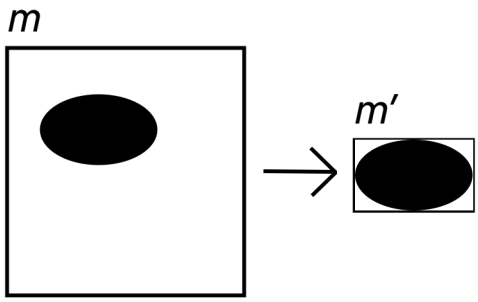

Figure 18: Retrieving m' from m

m' is then used to retrieve o', which is shown in figure 19.

Then, as show in figure 20, 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 section, 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 v, v 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

Figure 19: Retrieving o' from V

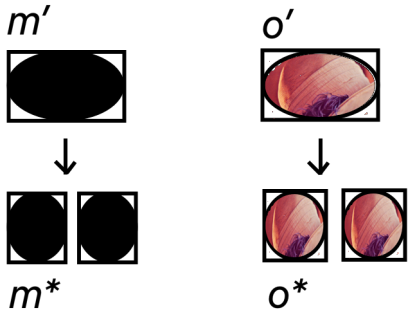

Figure 20: Decomposition of m' and o'

thus part of a mask which can be described as "background", which has a bounding box with the same shape as V. if the datapacket to be transmitted 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 $v \in o^*$, v 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. An example of this is shown in figure 21. This, together with Lemma 2, leads to the following property:

Lemma 3. *For every voxel v in V, v 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

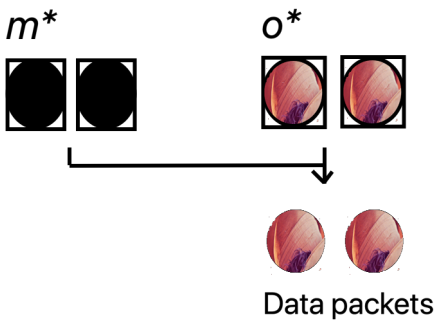

Figure 21: Generating the data packets from m^* and o^*

of the memory size of the compressed masks. The size of the compressed masks, and thereby indirectly the transmission time, will be discussed in the results section.

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 natural language 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. Section 4.3.1 will discuss the mask decoding procedure and its characteristics, whereas section 4.3.2 will discuss the procedure of decoding the image data received by the viewer, as well as incorporating the data into the internal representation of the workstation.

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 [29]. Therefore, this will not 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 intermediary 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.

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 $V' = V[m.\textbf{region}][m.\textbf{mask} == 1] \leftarrow \text{upsampled}[m.\textbf{mask} == 1]$;
5 return V';

Figure 22: Algorithm 2 is used to incorporate the received data into the intermediary data structure in the viewer.

4.3.2 Segment decoding and incorporation

As discussed in section 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 section 4.1.1, the definition of a mask was provided. 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 its location.

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, $k = 2n$, 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 $O(log\ n)$ subvolumes per mask, so therefore, post-processing of an entire segment will take $O(n\ lg\ n)$ 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.

$$Td = N/s_1$$

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.

$$Tp = c \cdot n + b$$

With c being some constant integer value. However, there is no initialization needed for Algorithm 2 or any of its subprocedures, hence

$$Tp = c \cdot n$$

with n being the number of elements in the axis alligned bounding box of the segment. However, it is also possible to think of Tp the following way:

$$Tp = n/s_2$$

With $s_2 = \frac{1}{c}$. Intuitively, s_2 can be thought of as the post processing speed in bytes per second. 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.

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.

$$Tp = N/s_2$$

Therefore, if

$$s_2 > s_1$$

transmission time Td will always be greater than post processing time Tp. If we assume the post-processing speed to be constant on a given machine, there exists a maximum threshold value S_{thr} 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 S_{thr}, 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 chapter, we will evaluate the results of the implemented solution. There are a number of ways to approach this. One could evaluate using many metrics, use user studies, or compare our solution to other solutions, or use a bigger set. However this space of evaluation is too large to process. Therefore instead of doing a shallow and broad analysis, we chose to do a narrower evaluation, and go in depth with the setup we used in this thesis. We have chosen a small set of data containing 21 volumes of 6 organ types, and evaluate our results using a single quality metric and did not consider user evaluation, different repositories or comparison with different software systems. We will present these results in the following sections.

Section 5.1 will discuss the dataset used to generate the results and some of its properties. Section 5.2 will evaluate the results for the compression method used to compress the masks. Section 5.3 will discuss quantitative results obtained from 140 samples. In short, section 5.3 provides a quantitative basis for the theory discussed in section 4.3, namely the relationship between transmission time and post-processing time. Once this relationship has been established, section 5.4 will provide information on the methods used to generate qualitative results from the solution's output. Section 5.5 will then analyze and discuss these results, generated from 21 volumes.

5.1 Data description

In section 4.1.1, a discussion was provided regarding the masks. More specifically, we discussed that generating the masks itself was outside of the scope of this thesis. Not merely because of time constraints, but also because a number of techniques have been described before which have succesfully accomplished this task[23][17]. Therefore, the testing and development of the solution has been done using existing annotated data from the CT-ORG image repository[11].

A reason for choosing this specific dataset is because of the diversity of the masks available. The masks define regions denoted as background, liver, bladder, lungs, kidneys, bone and brain. Figure 23 shows an example of a mask for every organ in the dataset. As described by the documentation of the dataset[11], the first 21 volumes constitute the testing split, and are considered to be the most reliable annotations[11]. Between the testing and training sets, there is only a difference when it comes to the methods used to generate the masks denoting bone tissue. Nevertheless, this thesis uses only the first 21 volumes for qualitative evaluation of the solution. Table 1 provides an overview of the characteristics of the used volumes.

These are the volumes that will be used for the qualitative analysis of the output of the solution. As you can see, every scan provides a sample for each organ, except the brain, which is only represented in scan 20. This means that from a statistical point of view, it is hard to make any claim for this organ. However, in the discussion an explanation will be provided on how properties of the brain segmentations can be inferred from the organs for which samples are abundant.

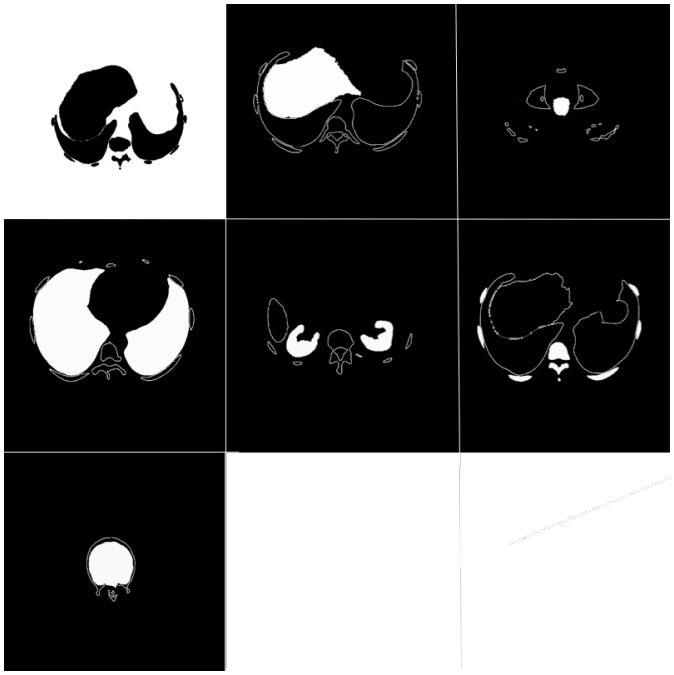

Figure 23: Examples of the masks: the white regions are the masks related to an organ, whereas the regions with white borders are the other organs for reference. Top row from left to right: background, liver, bladder. Middle row from left to right: lungs, kidneys, bone. Bottom row: brain.

	depth	organs
0	75	0, 1, 2, 3, 4, 5
1	123	0, 1, 2, 3, 4, 5
2	517	0, 1, 2, 3, 4, 5
3	534	0, 1, 2, 3, 4, 5
4	841	0, 1, 2, 3, 4, 5
5	537	0, 1, 2, 3, 4, 5
6	518	0, 1, 2, 3, 4, 5
7	541	0, 1, 2, 3, 4, 5
8	541	0, 1, 2, 3, 4, 5
9	549	0, 1, 2, 3, 4, 5
10	501	0, 1, 2, 3, 4, 5
11	466	0, 1, 2, 3, 4, 5
12	455	0, 1, 2, 3, 4, 5
13	605	0, 1, 2, 3, 4, 5
14	588	0, 1, 2, 3, 4, 5
15	565	0, 1, 2, 3, 4, 5
16	689	0, 1, 2, 3, 4, 5
17	574	0, 1, 2, 3, 4, 5
18	437	0, 1, 2, 3, 4, 5
19	247	0, 1, 2, 3, 4, 5
20	536	0, 1, 2, 3, 4, 5, 6

0	Background
1	Liver
2	Bladder
3	Lungs
4	Kidneys
5	Bone
6	Brain

Table 1: Left: number of layers per test CT scan and the organs which are part of the scan, denoted by numbers. Right: legend mapping numbers to organs

	Min	Q1	Median	Q3	Max
R	0.0001	0.0004	0.0005	0.0006	0.0009

Table 2: Minimum, Q1, median, Q3 and maximum values for the compression ratio R of the compressed masks

Furthermore, before the qualitative results will be discussed, an analysis of some necessary quantitative properties of the dataset will be provided in the next section.

5.2 Pre-processing: run-length encoding on dataset

In this section, the compression ratio for the masks will be discussed. As stated in the non-functional requirement for the PACS, the solution should not send significantly more data than the naive implementation. Lemma 3 establishes that every voxel from the input volume is transmitted exactly once, so any additional data to be transmitted is data related to the masks.

To compress the masks, the solution first downsamples the masks to a fourth of the original size, and then uses run-length encoding (RLE) to compress the masks even further. The type of RLE used is a custom implementation, and is explained in more detail in section 4.2.1.

Table 2 shows that for the CT-ORG dataset, the compression ratio is at most 0.0009. Since the masks are stored in 16-bit floating point format and the volume to be transmitted is stored in 32-bit floating point format, the compression ratio when compared to the input volume instead of the input mask is half of the values denoted in table 2. More intuitively, for every gigabyte of data in an input volume, an additional 450 kilobyte is transmitted at most given the CT-ORG dataset. For the purpose of the solution provided in this thesis, this amount of additional data is small enough to be negligible and therefore, the non-functional requirement of the PACS is met.

5.3 Post-processing

In section 4.3, an explanation of the relationship between transmission time and post-processing time has been discussed. More specifically, given an organ o, if there exists a linear relationship between N and n, where N is the number of voxels v for which holds $v \in o$, and n is the number of voxels v for which holds $v \in B(o)$, where B indicates the axis-alligned bounding box of o, there exists a threshold transmission speed S_{thr}, which if not exceeded, will guarantee that transmission time is always greater than post-processing time. However it is not the case that for every possible configuration, n is bounded by N. Nevertheless, if it is the case that given a set of masks M, the sparsity of the masks in M can be expected to lie within a small range with high confidence, S_{thr} can be expected to exist for that type of data.

Section 5.3.1 will focus on the analysis of sparsity per mask-type, or more intuitively, per organ. Section 5.3.2 will then analyze the results of computing S_{thr}.

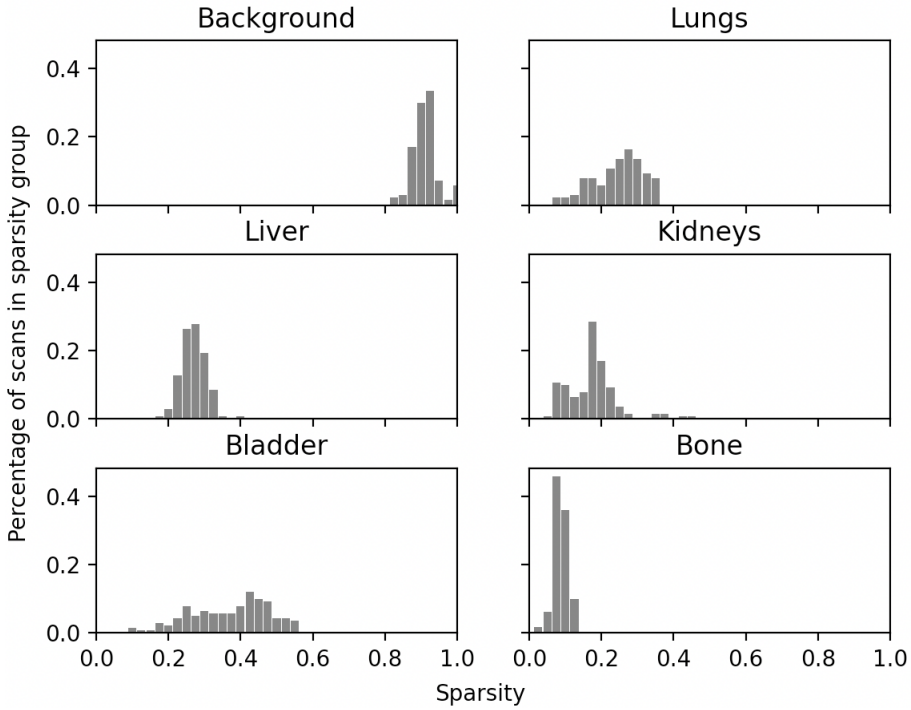

Figure 24: Histogram: Sparsity of the masks per organ type.

5.3.1 Mask sparsity

As discussed, if it can be expected that a mask m of a certain organ o has a sparsity which lies within a small range with high confidence, it can be expected that S_{thr} exists. S_{thr} is the transmission speed threshold value, which if not exceeded, guarantees that post-processing speed will always be shorter than download speed.

Figure 24 shows the density for the sparsity of the masks per organ type. The masks defining the brain have been left out of this analysis, as the CT-ORG dataset contains very little examples of brain masks. While the histograms in figure 24 are not perfect gaussian distribution, the shapes of the plots do suggest that the density lies in some range around a mean or median value. Tables 3 and 4 respectively show the 99% confidence intervals, as well as the minimum, Q1, median, Q3 and maximum values of these distributions.

Table 3 shows the confidence interval for the 99% confidence level. It shows that it is indeed the case that the sparsity per organ type can be expected to lie within a small range with high confidence. Therefore, it can be expected that S_{thr} exists with high confidence for these specific types of data. Of course, these data have to be viewed with some precaution, as the way of computing the confidence interval is designed for normal distributed datasets. However, when interpreted together with the histograms, sparsity per organ type does seem to center around some mean or median value. The following section will compute S_{thr} for every organ type, to further analyze the validity of this assumption, and thereby, the robustness of the solution.

	Mean	Lower bound	upper bound
Background	0.899	0.891	0.906
Liver	0.258	0.25	0.265
Bladder	0.358	0.335	0.382
Lungs	0.235	0.219	0.25
Kidneys	0.16	0.145	0.175
Bone	0.073	0.068	0.078
Brain	0.426	0.343	0.509

Table 3: The mean sparsity, lower bound of the 99% confidence interval and the upper bound of the 99% confidence interval per organ type.

	Min	Q1	Mean	Q3	Max
Background	0.802	0.877	0.899	0.916	0.988
Liver	0.155	0.237	0.257	0.278	0.381
Bladder	0.076	0.276	0.379	0.443	0.533
Lungs	0.054	0.188	0.245	0.288	0.347
Kidneys	0.041	0.118	0.164	0.19	0.442
Bone	0.021	0.06	0.074	0.083	0.109
Brain	0.241	0.415	0.446	0.474	0.48

Table 4: the minimum value, Q1, mean, Q3, and maximum values of the sparsity per organ-type

5.3.2 Transmission speed threshold value

In order to capture the full range of behaviour given a specific organ type, a value for S_{thr} will be computed for the samples at the minimum sparsity value, at the first quartile, the mean, the third quartile and the maximum sparsity value. This will both give an impression of the value of S_{thr} given a specific computer setup, as well as the performance while handling outliers. The sparsity values of these samples are shows in table 4.

S_{thr} is computed as follows: For a given sample and organ type, the post-processing time T_{pp} of the solution is recorded, as well as the amount of bytes n_{bytes} transmitted. The value for S_{thr} will then be determined through the following expression.

$$S_{thr} = \frac{n_{bytes}}{T_{pp}}$$

Table 5 shows the threshold transmission speed S_{thr} for all 5 scans per organ-type. These results have been computed on a Macbook pro, late 2016 model with a 2.9 Ghz Intel Core i5 dual-core processor, with Intel Iris Graphics and 16 Gb of RAM.

As expected, the samples with high sparsity generally have a lower S_{thr} value. Table 7 shows that there is a strong correlation between sparsity and S_{thr}. Furthermore, table 4 shows that bone tissue generally has high sparsity, which is reflected by poorer performance. Since the median and mean sparsity values of the masks for

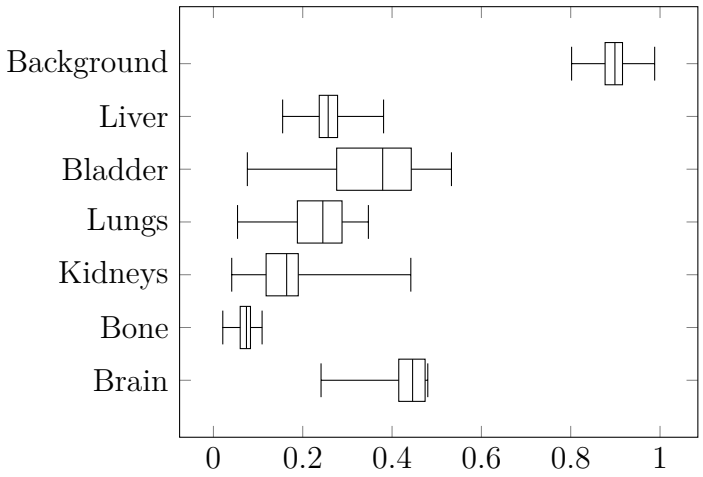

Figure 25: Boxplot showing the sparsity per organ type

	Min	Q1	Median	Q3	Max
Liver	25.02	38.54	38.38	47.12	62.07
Bladder	10.87	43.33	41.02	45.07	81.58
Lungs	10.29	40.66	43.04	45.68	46.35
Kidney	5.93	18.61	32.25	43.45	51.08
Bone	3.0	8.24	13.66	12.77	14.81
Brain	15.57	49.47	36.22	42.92	49.91

Table 5: Computed S_{thr} in Mb/s values for masks with minimum, Q1, median, Q3 and maximum sparsity values

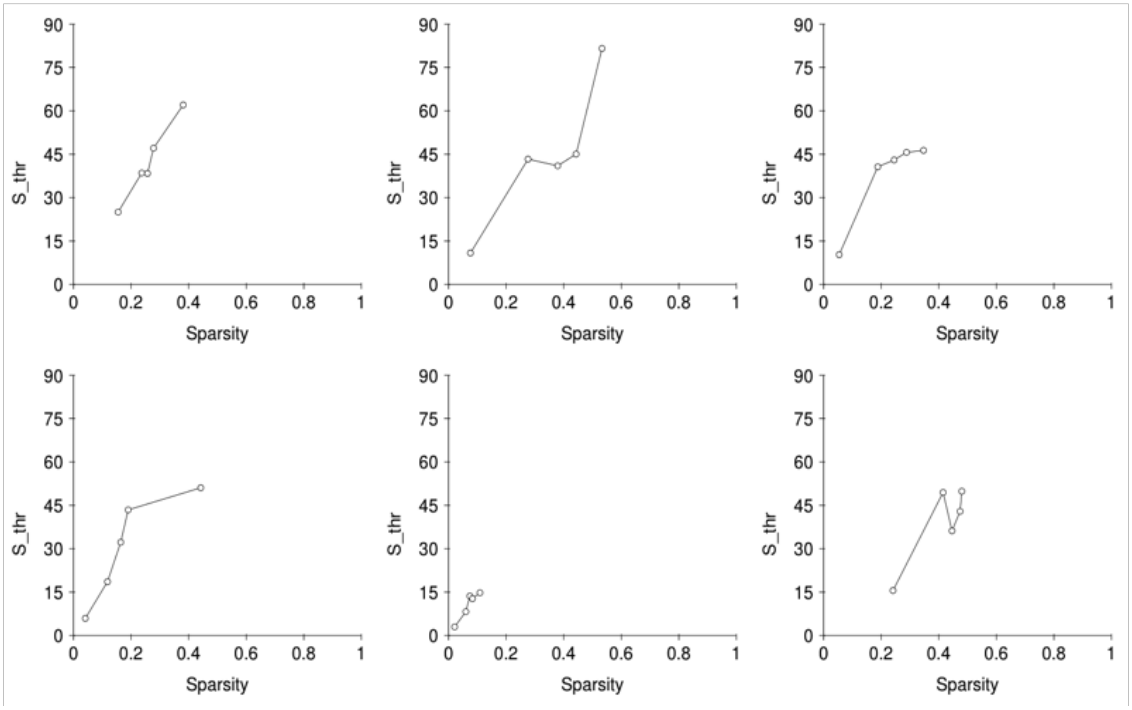

Figure 26: S_{thr} as a function sparsity of the used mask. Top row: mask type liver, bladder and lungs from left to right respectively. Bottom row, mask type kidneys, bone, brain from left to right respectively

	Min	Q1	Median	Q3	Max
Sparsity	97.32	114.14	99.58	93.26	93.47

Table 6: Computed S_{thr} values for volumes with minimum, Q1, median, Q3 and maximum size

bone tissue are both rather low, it would be realistic that the transmission speed would exceed S_{thr}. Therefore, while in most cases S_{thr} seems sufficiently high, it can't be guaranteed that the transmission time will not exceed S_{thr}, which effectively adds additional waiting time on top of the transmission time.

Until now, this thesis has not addressed the performance of the naive implementation described in chapter 2. Recapping, the naive implementation does not use segmentation masks, but simply decomposes the entire volume into packets at the server-side, and recomposing the packets back into a volume at the client-side. This allows for no prioritization of data during transfer. Table 5 shows the results of computing S_{thr} for the naive implementation.

Table 6 shows that the naive implementation generally has a S_{thr} value which is realistically not exceeded, or at the very least much higher than the smart implementation.

Summarizing, it can be concluded that in situations where the transmission speed does not exceed S_{thr} for the smart implementation, the naive and smart implemen-

	R
Liver	0.9902
Bladder	0.9103
Lungs	0.9179
Kidneys	0.8721
Bone	0.9497
Brain	0.8825

Table 7: Computed Pearson coefficient R for S_{thr} and sparsity per organ type

tation are equally fast when it comes to getting the data from the server to the screen. In situation where S_{thr} is exceeded, the naive implementation will display the entire volume faster. However, the added value of the smart solution has not yet been taken into account. In the following sections, the quality of the received images will be defined and the qualitative aspect of both implementations will be analyzed.

5.4 Quality Metrics

In order to analyze the quality of the output of the solution, the volumes generated at several point in the transmission need to be be evaluated. Consider a segment S, which is decomposed into n data packets and transmitted to the viewer. The viewer recomposes these packets into a target segment S'. In total, S' is updated n times, and all n states of S' will be compared to S to determine their quality. The evaluation will only be performed for the individual segment, as that is the most targeted way of testing the output of the solution, since there is no way to test the entire volume and inform the used quality metric which voxels are more relevant than others to the user.

To determine the added value of the smart implementation, a number of measurements have to be chosen to determine whether or not the smart implementation is better than the naive. As the main object of the solution is to display relevant information on screen earlier, one of the measurements will be the moment when a prioritized segment reaches full resolution. This moment will be defined as the number of bytes transmitted before that point divided by the number of bytes transmitted in total. However, while generating these data will shed light on when all information has been incorporated, it does not provide any information on the quality of the image during at every other moment during the transmission. Therefore, for every volume, the area under the curve (AUC) for some quality metric as a function of transmission progress will be computed for both the smart and the naive implementation. The reason why the AUC has been chosen is because it sums the quality metric over the entire transmission interval we're interested in. This data will be analyzed to determine the performance of the smart solution. In order to capture the performance for every organ type, the AUC will be computed for every organ type, or more intuitively, every organ will be prioritized once per volume.

	0	1	2	3	4	5	6
P	90.84	1.94	0.27	3.35	0.41	3.16	0.16

Table 8: Percentage of total data P which, on average, needs to be transmitted before a given segment reaches its maximum resolution.

Next, a quality metric needs to be chosen. Candidates that come to mind are MSE (Mean squared error) and PSNR (Peak signal to noise ratio). However, both MSE and PSNR are not suitable metrics to represent similarity in the perspective of the human eye [31]. The multi-scale structural similarity index (MS_SSIM [32]) better models similarity as perceived by humans when compared to MSE and PSNR[32]. Therefore, in this thesis, MS_SSIM will be used to compute the qualitative results of the solution.

5.5 Quality measures for segments

In this section the results of the selected quality metrics will be provided. Table 8 shows the at which percentage of transmission a segment of a certain organ type reaches its original resolution in the smart implementation. Since the naive implementation selects the data to be transmitted uniformly, we know that the last data packet will always contain some data of every organ type. Of course, when the data type is not restricted to medical imaging data, it is possible to construct a configuration of a mask which finishes loading before all the data has been transmitted. An example of this would be a mask which has only even or odd indices in one of its dimensions. However, since this is a very specific case and not likely to occur in medical data, we will ignore this edge-case and assume that the last data packet to be transmitted contains data from every organ-type. Therefore, the naive implementation always reaches 100% resolution of any segment after all the data has been transmitted.

When considering table 8, it is immediately quite clear that the smart implementation reaches full resolution much earlier in the transmission process. When the background mask is not taken into account (which is organ type 0), the smart implementation on average only needs between $\frac{1}{30}$th and $\frac{1}{625}$th of the amount of data needed by the naive implementation to reach full resolution. To see if there are any outliers in these generated data, table 9 provides an overview of the minimum, Q1, median, Q3 and maximum values of the percentage of data transmitted by the smart solution.

Table 9 shows there are no significant outliers in the dataset. Therefore, it can be concluded that in this dataset, prioritizing an organ o in the smart solution leads to drastically improved efficiency in transmitting o.

However, from these results, little can be said about the quality of the image before the image reaches full resolution at the viewer's side. To analyze this, we calculate the AUC for the value of MS_SSIM before the smart solution reaches full resolution. The reason why the AUC is only calculated for values prior to reaching full resolu-

	Min	Q1	Median	Q3	Max
Background	85.3	88.69	90.92	92.59	98.16
Liver	0.25	1.78	2.1	2.29	3.44
Bladder	0.01	0.13	0.25	0.37	0.64
Lungs	0.49	2.05	3.1	3.87	7.31
Kidneys	0.03	0.33	0.45	0.51	0.67
Bone	0.91	2.5	3.21	4.04	4.71
Brain	0.16	0.16	0.16	0.16	0.16

Table 9: Smart implementation: minimum, Q1, median, Q3 and maximum values for percentage of data needed to be transmitted to reach full resolution for that organ.

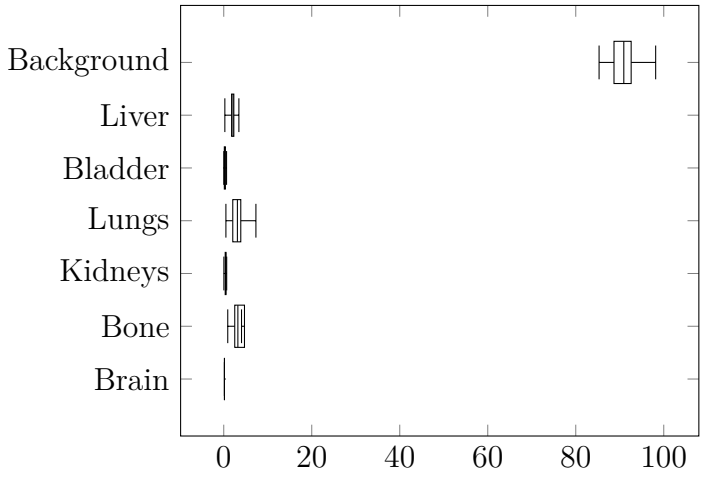

Figure 27: Boxplot showing the percentage of data needed to be transmitted before the target segment reaches full resolution

	Smart	Naive
Background	0.982	0.980
Liver	0.988	0.550
Bladder	0.946	0.076
Lungs	0.990	0.692
Kidneys	0.974	0.125
Bone	0.989	0.707
Brain	0.925	0.035

Table 10: Mean AUC calculated for the MS_SSIM values for a segment s up to the point where s reaches full resolution in the smart implementation and the naive implementation

	Min	Q1	Median	Q3	Max
Background	0.0	0.001	0.002	0.002	0.004
Liver	0.242	0.345	0.378	0.439	0.913
Bladder	0.508	0.848	0.88	0.933	0.984
Lungs	0.103	0.199	0.247	0.379	0.833
Kidneys	0.744	0.812	0.841	0.881	0.961
Bone	0.171	0.19	0.243	0.31	0.703
Brain	0.89	0.89	0.89	0.89	0.89

Table 11: The difference between the AUC for the smart implementation and the AUC for the naive implementation per sample, per organ-type

tion is based on the following assumptions. Firstly, consider T_{full}, which denotes the point in transmission where a prioritized segment representing an organ o reaches full resolution. After T_{full}, the segment representing o will not be altered. Furthermore, since the segment representing o reached full resolution, its MS_SSIM value will be 1. Secondly, since we know that the last data packet transmitted by the naive implementation contains data from every segment, it can be deduced that for every segment $S \in V$, $MS\_SSIM(S) < 1$ if the entire volume has not yet been transmitted. Therefore, it can be deduced that after the smart implementation reaches its maximum resolution, the MS_SSIM value of the prioritized segment will always be higher or equal for the smart implementation than for the naive implementation.

Table 10 shows the mean values of the computed AUC per organ type. At first glance, the AUC shows better performance of the smart implementation when compared to the naive implementation. However, there might still be outliers. Therefore, for every organ type, a list has been computed which at each index shows the difference in AUC between the smart and naive implementation for a given sample. A positive value indicates an improvement in performance of the smart implementaton over the naive implementation. Table 11 and figure 28 show these data. For every organ type, the minimum values are positive, which means that given the CT-ORG dataset, for every available sample and for every organ-type, the smart implementation outperforms the naive implementation when regarding quality of the image per transmitted byte before the smart solution reaches full resolution, which is denoted as T_{full}. Since it has already been established that after T_{full}, the

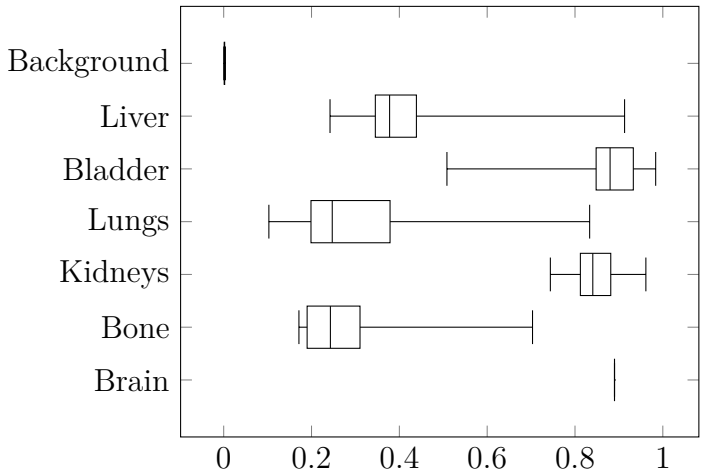

Figure 28: Boxplot showing the difference in AUC between the output of the smart implementation and the naive implementation

smart implementation outperforms, or at least has equal performance to the naive implementation, it can be concluded that over the entire transmission process, the smart implementation accomplishes displaying relevant information on screen faster than the naive implementation.

6 Discussion

In this section, the implications of the results and their meaning in the context of the requirements and the research question will be discussed. Furthermore, in section 6.3, an overview of the shortcomings of this thesis will be provided, as well as some solutions to these shortcomings. Moreover, possible future avenues of research will be discussed and some alternative implementations will be suggested.

6.1 Requirements

In section 2.6, a number of requirements together with their importance have been provided. In this section, these requirements will be discussed

Functional requirements workstation

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

The solution provides this functionality, therefore, this requirement has been met

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

The workstation queries the user which organ is relevant to their research.

Therefore, if the segmentation is accurate, this requirement is met.

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

The workstation is able to inform the PACS, given a segmentation, which organs are relevant to the user. This requirement has been met.

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

The masks are the first thing to be transmitted from the PACS to the workstation. Therefore, from the start of the transfer of the actual medical volume, the workstation has access to the masks stored in the PACS.

Non-functional requirements workstation

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

If the sparsity of the masks can be shown to be bounded by some constant, The time and space complexity of the solution are the same as those of the naive implementation. Since this is the case for the CT-ORG dataset, in our experiments, this requirement has been met.

Functional requirements PACS

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

The solution provided by this thesis provides the functionality to order the data in such a way that data which is regarded by the user as most relevant is transmitted before data about which the user has expressed no interest. Therefore, this requirement has been met.

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

Because of time constraints, as well as resource constraints, the solution does not provide functionality to automatically segment volumes of medical data. However, section 4.1 refers to some pieces of literature which do provide theory and implementation related to automatic segmentation.

Non-functional requirements PACS

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

Section 5.2 showed that the additional data to be transmitted is at most 450 kilobyte per gigabyte in the input volume for the CT-ORG dataset. This is

small enough to be negligible and therefore, this requirement has been met.

6.2 Research question

In section 1.1, the following research question has been posed: *Is it possible to change the current architecture in order to provide the user with meaningful information earlier on in the downloading process.* Given the provided results, two answers can be formulated.

Firstly, if the transmission speed from PACS to viewer does not exceed the threshold transmission speed S_{thr}, then the solution proposed in this thesis always provides the user with more meaningful information earlier on in the transmission process.

Secondly, if the transmission speed does exceed S_{thr}, then the solution does provide meaningful information earlier on in the transmission process, but the post-processing will add some waiting time for the user. Nevertheless, the results show that the smart solution does need orders of magnitude less data than the naive implementation to show a relevant organ on screen. Therefore, even if the transmission speed does exceed S_{thr}, the smart transmission will probably shows meaningful information on screen faster than the naive implementation will, but the naive implementation will finish processing the entire volume before the smart implementation does.

6.3 Limitations and suggestions for future research

As explained in section 4, the smart solution was implemented using a lab environment. As the provided solution was intended as a proof of concept, no consideration was put into the efficiency of the used programming language, which is Python 3. A way to optimize Python's handling of numerical data is by using C-types [12]. This can be done by implementing the whole codebase in C or by using Cython, which provides an API for running native C code in python. According to the developers of Cython, it greatly improves the speed in which python handles numerical data [12], which would improve S_{thr} for every organ type. Furthermore, the Cython API is much better at handling sparse data volumes [12], thereby possibly offering improvement to the solution where it is needed the most: in processing sparse segments.

Another way of decreasing sparsity, and thereby increasing S_{thr} is by using an oriented bounding box instead of axis aligned bounding box when storing segments and masks. An oriented bounding box (OBB) is the smallest cuboid volume which contains a given object. Since it contains less entries than the axis alligned bounding box (AABB) [30], yet the same amount of relevant data, it can be concluded that using a OBB would decrease the sparsity of segments and masks. However, computing an OBB is more costly than computing an AABB. Future research may determine whether the costs of using OBB instead of AABB will improve overall performance.

In this thesis, there are a number of shortcomings which should be addressed. Firstly, it does not address the performance when using segmentation which use different organs than the ones present in this dataset. However, it does contain the organ which is probably one of the most sparse, namely the bone. Therefore, it can be assumed that if an organ mask is less sparse than the average bone mask, the implementation is still viable. The same can not be said when regarding organs like the skin, which will have very high sparsity. Future research is therefore needed to determine if the solution is feasible for every organ type.

Related to the former point is the low availability of data for the brain. However, As the brain is a rather convexly shaped organ, and is contained in a skull which enforces the shape to be within certain constraints, it is likely that most of the masks defining the brain won't be very sparse and it can be assumed that there won't be a lot of variance within the dataset, and therefore, the solution will probably be feasibly for transmitting segments defined by the brain.

Lastly, a suggestion for anyone who would like to implement this solution in their own infrastructure. If one wants to use the benefits of the smart solution, but strictly does not want the total loading time to be longer than the naive implementation, the S_{thr} of the most sparse organ can be stored separately and the following condition can be used: Given the current transmission speed S_{cur} from PACS to viewer, if $S_{cur} > S_{thr}$, use the naive implementation, and if $S_{cur} <= S_{thr}$, use the smart implementation.

Bibliography

[1]

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

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

[4] Example of mammogram. http://www.rad.msu.edu/Course/RAD553/image_lib/body/BT14.htm.

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

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

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

[8] Over 75 million ct scans are performed each year and growing despite radiation concerns. https://idataresearch.com/over-75-million-ct-scans-are-performed-each-year-and-growing-despite-radiation
8 2018.

[9] Ehud Artzy, Gideon Frieder, and Gabor Herman. The theory, design, implementation and evaluation of a three-dimensional surface detection algorithm. *ACM SIGGRAPH Computer Graphics*, 14:2–9, 01 1981.

[10] National Electrical Manufacturers Association. NEMA PS3 / ISO 12052, digital imaging and communications in medicine (DICOM) standard. https://www.dicomstandard.org/current/, 2020.

[11] K. Shivakumar et al B. Rister, D. Yi. Ct-org, a new dataset for multiple organ segmentation in computed tomography. *Scidata*, 7:381, 11 2020.

[12] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science Engineering*, 13(2):31–39, 2011.

[13] Eyal Bercovich and Marcia Javitt. Medical imaging: From roentgen to the digital revolution, and beyond. *Rambam Maimonides Medical Journal*, 9:e0034, 10 2018.

[14] Peter Gøtzsche and Margrethe Nielsen. Screening for breast cancer with mammography. *Cochrane database of systematic reviews (Online)*, 358:CD001877, 01 2011.

[15] 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.

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

[17] Anna Khoreva, Rodrigo Benenson, Jan Hosang, Matthias Hein, and Bernt Schiele. Simple does it: Weakly supervised instance and semantic segmentation. pages 1665–1674, 07 2017.

[18] 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.

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

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

[21] S Mazziotti, F Arceri, Sergio Vinci, Ignazio Salamone, S Racchiusa, and I Pandolfo. Role of coronal oblique reconstruction as a complement to ct study of the temporal bone: Normal anatomy. *La Radiologia medica*, 111:607–17, 07 2006.

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

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. volume 9351, pages 234–241, 10 2015.

[24] 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.

[25] 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.

[26] I. Sechopoulos. A review of breast tomosynthesis. Part II. Image reconstruction, processing and analysis, and advanced applications. *Medical Physics*, 40(1):014302, January 2013.

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

[28] 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.

[29] 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`, 7 1984.

[30] C. Tu and L. Yu. Research on collision detection algorithm based on aabb-obb bounding volume. In *2009 First International Workshop on Education Technology and Computer Science*, volume 1, pages 331–333, 2009.

[31] Z. Wang and A. C. Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117, 2009.

[32] Z. Wang, Eero Simoncelli, and Alan Bovik. Multiscale structural similarity for image quality assessment. volume 2, pages 1398 – 1402 Vol.2, 12 2003.