UNIVERSITY OF BERGEN

DEPARTMENT OF INFORMATICS

VISUALIZATION GROUP

Comparative Visualization Of Longitudinal PET/CT Data For Acute Myeloid Leukemia Treatment Analysis

Author

MATHIAS BØE

Supervisor

NOESKA SMIT

September 1, 2022

Scientific Environment

This study is carried out at the University of Bergen, Visualization Group. The work is supported by Mohm Medical Imaging and Visualization Center (MMIV) and their project regarding AML treatment research and analysis.

Acknowledgements

I would personally like to thank my supervisor Noeska Smit for her continued support and guidance both in this thesis and in life. This thesis would, without question, not be possible without her support.

The visualization group at UiB has also been a good resource for discussion of different topics, being in the same room as leading researchers within the field has been inspiring, thank you all for valuable discussions and your time.

I would also like to thank all my friends and family for supporting me, it would not be possible without our album ratings or physical activities. Motivation for a project such as this is key to succeed, you gave me that.

A special thanks to Sondre Aasemoen for his rubber ducking skills and help when the non-caffeinated and tired brain sets in late at night.

Mathias Bøe

Glassburet, Bergen

01.06.2022

Abstract

The comparison of temporally separated multimodal imaging data is time consuming work. This thesis presents a novel approach for comparing multimodal imaging data, that supports different time points.

The Comparative visualization is a field of study that attempts to understand and explore the differences between two or more sets of data. While this area has been researched thoroughly in the past, it can still be applied to a wide range of new problems today. Especially in the medical field, where patients rely on these new approaches and technologies improving over time.

Comparative visualization is used in this thesis to compare two time-separated scans with each other in hopes of finding a new and faster way of comparing scans, reducing the time cost for the patient.

This thesis explores methods and techniques for comparing images and present a new application for an existing approach to comparative visualization. The proposed visualization, called The Sliding Window, is implemented in many websites and blogs but remains unexplored in the medical field.

The Sliding window approach allows the user to slide between different imaging datasets to see different states of the same subject or different subjects altogether. The approach is similar to juxtaposition mixed with superposition, where multiple images are layered on top of each other or side-by-side. This allows the power of human perception to be leveraged to perform a visual comparison. Sev-

eral blend modes are available that allow image fusion to be combined with the Sliding Window approach.

The implementation is then evaluated by using a System Usability Scale, based around a five point Likert scale. The results are promising, although the discovered usecase of being able to turn off/on PET layers in a PET-CT dataset was more valuable than the intended usecase according to vocalized opinions and evaluation scores.

Contents

Chapter 1

Introduction

Visualization is a field of study that handles how to convey data to the user efficiently. Visualization falls under the computer science umbrella in which the user uses a computer to solve problems and gather data. The data is then used to create visualizations enhancing human understanding and perception of the dataset. With good visualizations, the user will understand the data better, make better decisions, and find solutions to problems that might be hard to solve with algorithms alone.

Visualization is often targeted toward the visual channel of the user because its the highest bandwidth information channel of the human-perceptive system. Comparative visualization especially does this, sometimes to a fault, flooding our visual system with more information than what our brain can handle, resulting in bad visualizations or visualizations where the user will need extensive training or practice to efficiently spot differences.

Visualizations can unlock powerful data pattern processing, However, the designer of the visualization must be aware of several perceptual illusions that may cause viewers to map visual representations back to their original numerical values incorrectly. One example of such an illusion of perception is happens by changing the Y axis scale, resulting in a larger/smaller change than the user of the visualization might expect and making skewed decisions based on the false information obtained [1]. This leads to our eyes, as fast as they may be at gathering information, can gather the wrong information.

Our brains can rapidly perform a visual comparison and differentiate between colors, shapes, and sizes. Comparing multiple things at the same time takes more

time, our eyes might be able to gather the information at a sufficient speed, although our brain might not be able to process it at the same rate. This is where comparative visualization techniques come in. The goal of most comparative visualization techniques is to limit the information overhead that occurs when our eyes are looking at, and, taking in new information (i.e. unnecessary visualization components and clutter), or enhance visualizations with data that is hard to compare without help.

1.1 The Problem

Acute myeloid leukemia (AML) is an aggressive blood cancer. AML is characterized by infiltration of the bone marrow (BM), blood, and other tissues with proliferating leukemic cells, which rapidly lead to BM failure and eventually death if left untreated. Even though the disease was incurable 60 years ago, doctors around the world have made significant progress in curing AML. In 2015 35-40% of adult patients, age 60 and younger, were cured of AML [2]. 60 years and older still have depressing survival rates as chemotherapy, the standard treatment, has severe side effects, especially for the elderly. In people 60 and older, the survival rate is only 5-15%, with a median survival of only 5-10 months. Even though the survival rates are low, they are luckily trending upwards after hard work from many different fields and institutions [3].

Today patients are initially treated as a one-size-fits-all with chemotherapy, with additional targeted treatment based on mutational data, even though some progress has been made that suggest we will have options in the future [4]. Failure to fully or partially treat patients leads to a high risk for later relapse and increased mortality, making finding non-responders at an early stage that much more important. Currently, there are no clinically in vivo non-invasive methods available for monitoring AML treatment response, a problem the overarching study for this thesis is trying to find better solutions for or solve entirely.

1.2 Comparative Visualization

Pagendarm and Post[5] discuss comparative visualizations and how important they can be. Pagendarm and Post defines comparative visualization as: “Data from two or more different sources are visualized with the intention to show similarities and differences.“

Usually comparative visualization is done by placing two images side by side and letting the eyes and brain do the comparison. There is a limit to how effective this can be, as a person can only be trained to see the appropriate differences to a certain extent. New technology can however unlock new possibilities that the human brain wont be able to compute or see. The Sliding Window is not a tool for directly comparing medical images, but rather tries to enhance the experience by overlapping two separate visualizations and allow the user to slide between them to perform a more precise comparison.

1.3 Motivation

The motivation for the overarching study this thesis is a part of is to find non-responders as quickly as possible without the use of invasive methods. This thesis, however, will focus on the comparative visualization of the different datasets and resulting scans. The goal is that these visualizations will aid the process of finding non-responders at an earlier stage, which will allow more time for doctors to find additional or alternative treatments if necessary.

1.3.1 Alignment

One of the main issues with comparing scans especially time-separated scans is alignment and image registration. Adding to this, there are visualizations available to radiologists that are not directly comparing scans but rather viewing them side by side and letting the brain do the processing. Software exists that overlays images on top of each other, but none of them has the same interaction as the Sliding Window.

Directly comparing the different scans, in this case, is a hard problem to solve since we are talking about temporal data where patients do not necessarily lay in the same position every time. Even though patients try their best to be in the same position, making sure the scans are 1:1 is practically impossible with the techniques for gathering data available today.

Even if you were to synchronize the viewports at a certain ROI (Region Of Interest) in the transverse plane, it would only stay synchronized for a small distance, maybe even giving false information as you think the viewports are synced, but in reality, you are looking at different points within the body. This happens because even though the patient is laying in the same position; the spine will have a

different curve every time, leading to inconsistencies in the length between vertebrae, making the viewports immediately out of sync with each other. This, as well as the difference in slice width, can both lead to a false impression of something not being similar, while you are just looking at different slices of information. Solutions to this image registration issue have not been solved entirely, but are still being worked on. Some of the solutions proposed consist of deforming the resulting images to align them [6]–[8]. The problem with this being only the ROI will be readable and not the entire image. Machine Learning and Deep Learning might be able to deform the images in a way that does not affect the surrounding tissue and other relevant parts of the data sets in the future, but currently, implementing such methods into the Sliding Window is outside the scope of this thesis.

Deforming an image is not optimal, especially when relying on PET-CT images for comparing metabolism in cells, the ROI will potentially be minimal, and the CT image will be larger. One of the issues that came up when discussing this with medical professionals was that anatomical structures and guides for where you are looking within the body are important for the PET scan. Meaning deforming the CT scan makes it harder to compare scans as the anatomical data might be lost.

Alignment, also called image registration, of scans is a hard problem to solve and many different fields are already doing research on this[9]. Significant research has been done in recent years when it comes to using machine learning (ML) or deep learning (DL) to do image registration. Fu et al.[10] does a good job of collecting over 150 papers and their results using DL for automatic image registration, although they see good progress in the field, their conclusion is that it still needs work and research to be a viable solution to this problem. Considering DL-based medical image registration is a relatively new field of research, it shows enough promise to consider as a future solution to aligning medical scans. This applies not only to this Sliding Window application but also to comparative visualization as a whole and many other non-related fields.

1.4 The Sliding Window Approach

This thesis presents the Sliding Window, a web application, and its interface for comparing multiple scans of the same patient over a specific time period. It can also facilitate comparison between any two datasets that are encoded in the sup-

ported standards.

The data consists of multimodal medical imaging data from cancer patients. The patients get scanned multiple times, assessing their disease progression and treatment options while under the care of doctors. The data gathered from these scans are then compared and reviewed by medical professionals to figure out the best course of action on a patient-by-patient basis. The Sliding Window is, however, a tool primarily aimed at researchers exploring treatment options and separating treatment responders from non-responders.

The Sliding Window technique proposed in this thesis has been widely used online by many newspapers and blogs to compare before-after images of locations/people/houses/images of photographs. It has yet to be implemented or tested in the medical field. The closest thing available is the *"magic lens"* an approach where you have an area around the cursor that shows some other part of the dataset, or a completely different dataset altogether [11]. In addition, the *"checkerboard"* approach is closely related, where the images are overlayed on top of each other and you form a checkerboard pattern. Black squares being from one data set and white squares from the other. This allows users to compare the two data sets and check the correctness of the image registration [12].

The Sliding Window approach has many names online but usually contains something like *compare-slider, compare-window* and so on. In this case, a Sliding Window is more accurate as you are sliding between two different windows, or in this case, viewports.

When comparing scans today, a medical professional will have to open up the different datasets in multiple large windows. Furthermore, the open views have no connection to each other, such as overlaying the scans is not a feature that is available in most of the current software, especially applying filters and changing opacity is not something that is within the current workflow. Screen real estate is not necessarily large in every medical office, as well as bringing this information to meetings or patients will have to rely on non-interactive prints or images on a screen.

The Sliding Window attempts to solve both of these issues with its web focused view, making it possible to open up two different scans in the same window in the browser. Since this is a web-application it would be possible to open this web-app on tablets or phones as well, meaning the user could bring an interactive window with them. Portability means the user can bring the visualization to meetings for

discussing treatment or alternative procedures or bring the data to the patients to improve doctor-patient communication.

Automatic image registration is not implemented in the Sliding Window; it does, however, try to alleviate the alignment issue by providing good tools adjust the alignment while going through the slices. By overlapping two time-separated scans over each other where you can slide back and forth and compare a set position in the two scans, for example, a bone structure at a point close to the point of interest or other anatomical landmarks such as the pubic symphysis[13], that way, you can make sure alignment is as accurate as possible. The Sliding Window also provides different blend modes and an opacity slider to further help comparing and aligning scans. Automatic alignment of the viewports and scans is not within the scope of this thesis, which makes it important to provide good tools to align them manually.

Diepenbrock, Hermann, Schäfers, *et al.* [14] describes some of the issues they had comparing two arteries in mice. They produce atherosclerotic plaque in one carotid artery by a constrictive cuff and the other carotid artery serves as a control. One of their mentioned shortcomings is the missing ability to view both the left and right carotid artery at the same time with sufficient zoom levels. The solution proposed in this paper somewhat mitigates this and could, in theory, be used in later studies to show the difference between healthy and normal structures versus abnormal structures in PET/CT imaging. Especially combined with other visualizations such as Angelelli and Hauser [15] technique for straightening tubular flow or Daae Lampe, Correa, Ma, *et al.* [16] which is the base for the straight tubular flow technique.

1.5 Objectives

The overarching objectives of this thesis are:

- To design an application that allows for interactive comparison between image scans.
- To develop an easy-to-use interface for both experienced and inexperienced users.
- To evaluate the suitability of Sliding Window technique for medical imaging data.

1.6 Contribution

This thesis present visualizations that mitigates the issue of two regions of interest being too far apart for one single image.

An interactive web application with a sliding window that allows the user of the application to selectively change between different temporal dimensions is presented.

The web application is evaluated by five domain experts through a System Usability Scale to demonstrate the value of the Sliding Window in medical imaging.

The Sliding Window attempts to solve the issue of bringing data with you to where you need it as well as comparing scans interactively and exploratively in a familiar and intuitive way.

The Siding Window further tries to solve the issue described by [14], where regions of interest are too far apart to compare in one image.

The last goal of the sliding window was to reduce the time spent comparing two different data sets and to facilitate further research and exploration of blend modes.

1.7 Notable Terms

- **Sliding Window**: The web app developed for this project.
- **Production environment (prod)**: Code that is deployed on servers and served to end-user.
- **Development environment (dev)**: Code that is deployed on servers, used to test the deployment of code before sending it to production servers. Usually served to developers and testers.
- **DICOM**: Digital Imaging and Communications in Medicine. A DICOM simply explained, is both a network protocol deciding how a given modality can transfer an image to another, and how it can browse the content of remote modalities [17] and a wrapper for image files that wrap the images in meta-data related to the image.

- **DIMSE**: DICOM Message Service Element. A protocol for sending and receiving DICOM messages [18].
- **Orthanc**: A lightweight, RESTful DICOM server described in the work by Jodogne, Bernard, Devillers, *et al.*[17].
- **REST-ful**: REpresentational State Transfer. REST-ful represents a style of API based on transferring a representation of state between client and server. [19]
- **ROI**: Region Of Interest.
- **MIP**: Maximum Intensity Projection.
- **API**: Application Programming Interface. Its job is to serve a simple interface for computers or software to *speak* to each other.
- **Window Level Window Width**: Window Width controls contrast, Window Level controls brightness.

Chapter 2

Background

A large part of the Sliding Window approach is based on the data gathered from different medical equipment. The first section of this chapter describes the format of the data, while the rest goes further into the data gathering aspect.

2.1 DICOM

DICOM has over the years become the *de facto* standard for encoding medical images [20]–[22]. The protocol started with a point-to-point approach in mind. Resulting in a tedious to manage network at hospitals since the different modalities of DICOM might only be supported by certain private actors within the medical field. Usually, this results in hospitals having intermediate servers for the DICOM files that handle the *translation* between the different modalities. One such server or *translator* is Orthanc.

Another used in hospitals is the PACS, which also handles the translation between different DICOM modalities. Depending on the load at the entire hospital, PACS might get overloaded and unable to handle the workload, resulting in an immediate effect on the clinical routine of the entire hospital. This could be solved by implementing many DICOM bridges that work independently from the PACS [17].

2.2 Picture Archiving And Communication Systems (PACS)

PACS are standard components of medical imaging these days. For routine applications, they provide a secure and easy-to-use interface to clinical imaging data. Sometimes a researcher needs more flexibility for the medical data, thats where research-PACS have been invented, a much less used standard that only some hospitals have as of 2022.

2.2.1 Research-PACS

PACS work well within the hospital and for clinical imaging data. However, the workflow required to keep personal information and data out of the wrong hands make it really hard to work with the PACS. Thats why research-PACS have been invented, as less secure options for researchers to use. The data in the research-PACS are generally anonymized by default both for the researchers sake, but also for security reasons [23].

2.3 Medical Imaging

This section will describe the medical imaging techniques used to get the data for the Sliding Window visualization. The description of these are based on the work done in the STAR (State Of The Art Report) by Aladl and Peters [24], and the survey on multimodal medical data visualization performed by Lawonn et. al. [25].

2.3.1 Single Photon Emission Computed Tomography (SPECT)

SPECT imaging uses a gamma camera to acquire multiple projection 2D images from multiple angles [24]. The camera captures a radioactive tracer material injected into the patient. The tracer liquid gets injected into the patient and lights up areas of interest, such as areas with high blood flow and higher metabolism. The latter is vital for this study, as cancer cells have a different lipid metabolism than other cells [26]. It is also possible to use the information from these images

to create a 3D volume.

2.3.2 Positron Emission Tomography (PET)

PET scanning is similar to SPECT, except that the isotopes emit positrons and not gamma-rays. The gamma-ray is then generated when the positron is annihilated by an electron, which is then detected in a similar way to SPECT. The PET scan itself is not that useful without being fused with anatomical data from CT or MRI [24]. The reason is that the PET scans are low resolution, and hard to make out what you are looking at based on anatomical data, whereas CT and MRI present a much more readable anatomical map of locations within the body.

2.3.3 Computed Tomography (CT)

Also known as CAT (Computed Axial Tomography), CT reveals both bone and soft tissues. CT is done by collecting a large series of two-dimensional X-ray images on a single axis. Through the use of reconstruction software, it is possible to generate a 3D volume from this data as well [24]. CT images can and is used in the treatment planning phase of many medical issues, including calculating dose distribution based on electron density values. CT is not well suited for soft tissues compared to the superior soft tissue images of MRI [27] [28].

2.3.4 Magnetic Resonance Imaging (MRI)

MRI is based on the principles of nuclear magnetic resonance (NMR) which is used to obtain microscopic chemical and physical information about molecules. [24] It is based on the alignment of the nuclear magnetization of (usually) hydrogen nuclei of water in the body. Radiofrequency fields are then used to systematically alter the alignment of this magnetization. The hydrogen nuclei then produce a radio frequency signal that the scanner can detect and interpret to construct an image of the body.

2.3.5 PET-CT

PET-CT is an example of multi-modal images. PET-CT was proposed by Siemens in 1998 and clinically evaluated at the University of Pittsburgh. It is a matured

and well used tool to diagnose a plethora of diagnoses [29]. PET is used to enhance colors/contrast in areas where metabolic activity differs, not visible in CT alone. The goal of the creation of the PET-CT was to scan PET and CT within the same device so one could acquire a full anatomical and functional scan in a single session. Removing the need for a patient to do multiple scans. This also has the added benefit of capturing multiple modalities at the same time, meaning aligning these scans is way simpler than if done asynchronously [30].

2.4 Multimodal Medical Datasets

Multimodal medical datasets consist of several scans of the same subject using the various acquisition methods described in section 2.3. The common workflow for obtaining a multimodal dataset is gathering anatomical scans from higher resolution devices such as CT, MR and MRI scanners. The data is then combined the data with data from nuclear medicine devices such as PET or SPECT scans. PET and SPECT depict functional processes, such as metabolism in a lower resolution, but combined with the data from the other sources it becomes a really valuable asset when doing research or diagnostics. Tracer uptake is also important for the nuclear modalities as it is the main feedback from the data as it will diverge from the expected values. Getting the dosage right and using the right technologies is therefore vital to the resulting visualization. Any visualizations made have to assume that the tracer uptake, dosage, and data are correct to perform well.

Chapter 3

Related Work

Lawonn et. al. [25] provides an extensive survey on multimodal visualization techniques for multimodal imaging data, some of which are mentioned in section 2.4.

3.1 Visualization

Many visualizations have been proposed through the years on how to do medical visualization. This section will go through some of the methods and visualizations that have survived the test of time and are still in use today as well as some novel approaches that deserve another look, either in the medical field or related fields.

3.1.1 Multimodal Data Visualization

Multimodal data visualization within the medical field refers to using multiple data sets (scans) from different sources to combine them into one image or interface for user interpretation. In the medical field, multimodality is closely intertwined with image fusion, as image fusion is often used to process medical scans, creating new and easier-to-read results. With many multimodal visualizations, users are able to interact with the visualization to switch between the multiple modalities to enhance the ROIs, such as with PET-CT, where the PET scan is often used to guide the user to ROIs within the CT data. Image fusion is either achieved through asynchronous post-processing or fused automatically by acquiring the different modalities simultaneously [29]. In this way, the Sliding

Window can be considered interactive multimodal image data visualization. The main purpose is to increase the context of information, allowing for a fast and more intuitive comparison of the data.

3.1.2 Comparative visualization

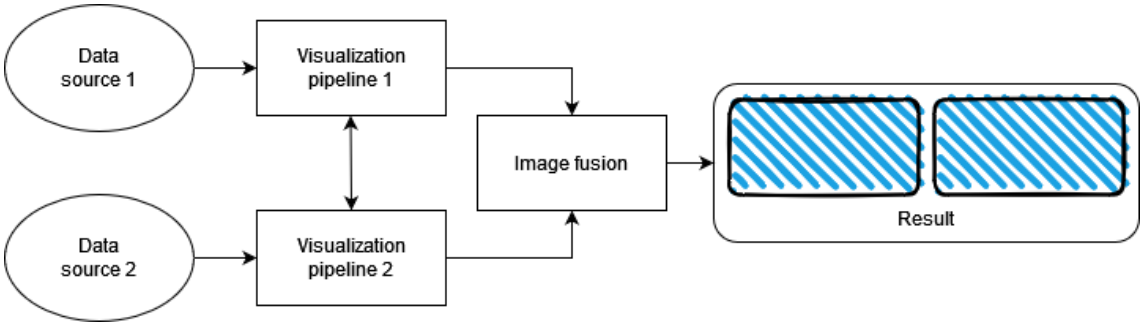
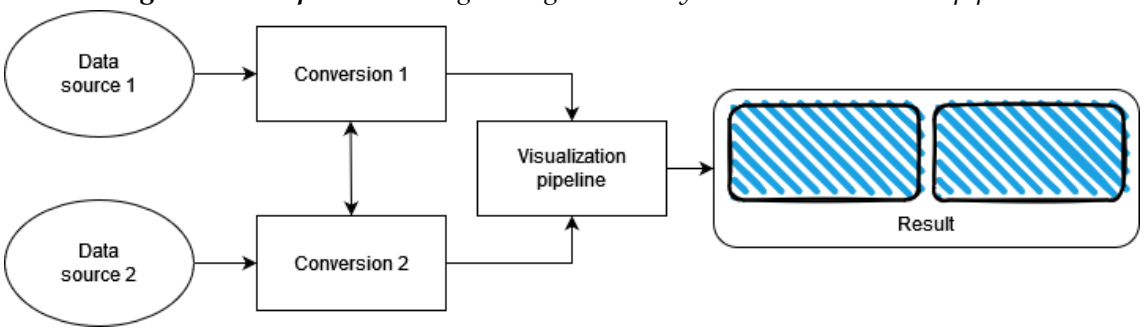


***Image level comparison**: Images are generated by its own visualization pipeline.*



***Data level comparison**: Data from two sources are transformed to a common representation, then fed into the same visualization pipeline.*

Figure 3.1: Two approaches to comparative visualization, figure recreated with inspiration from [5]

Pagendarm and Post[5] write that in comparative visualization, two approaches can be identified (see Figure 3.1): *image level comparison* and *data level comparison*. Image level comparison involves two visualization pipelines. The goal of the pipelines is to output comparable images. There are two main approaches to these visualization pipelines. They both use different techniques to achieve the goal, one is the optical visualization technique, the other is physical [5], [31], [32].

Examples of optical techniques

- **Shadowgraphs**, a technique used to reveal non-uniformities in transparent media such as water, air or glass. It captures light that the human eye cant

obviously see, the refractions through the transparent medium is the visible part (think of the shadow of the warm air above a fire pit when the sun shines through it).

- **Schlieren photographs**, very similar to shadowgraphs, but the light source is intercepted (in the shadowgraph example it was the sun, but in reality it would be a controlled light source) in such a way that only the relevant refracted light hits the image capturing device/medium.
- **Interferograms**, a technique that uses multiple light sources and capture the intereference between the two. Similar to taking a picture of interfering waves in a pool.

Examples of physical techniques are photographs of patterns revealed by adding tracer materials such as: *smoke, dye, powder, and medical tracer reagents*

Comparative visualization is often, if at all done by placing images side by side and letting our brain do the processing and comparison [5]. Pagendarm and Post Further list some of the ways that visual differences may present themselves:

- Different physical phenomena
- Different conditions (experimental or numerical)
- Measurement artifacts: Noise, sampling resolution, interference with the phenomenon.
- Numerical inaccuracies
- Different mathematics or logic
- The visualization process.

Pagendarm and Post goes further into the issues with this and some solutions to these issues, such as juxtaposition and superimposing images on top of each other using image fusion techniques to highlight the differences, similar to what is being done with blend modes in The Sliding Window web app.

Comparing scans directly like Dzyuback et al. [33] by overlaying scans from follow-up sessions to the first scan and applying them as a "filter" is an interesting idea and would be useful to implement. Dzyuback et al. assume that image registration is done with pre-processing, the same way as in the case of the Sliding Window. They also mention their use of bones as a basis for image

registration. The Sliding Window has multiple tools (6.1.4) for manual image registration if needed, as well as the possibility to simulate the visualization proposed by Dzyuback et al. This is achieved through the use of CSS filters (blend modes) such as *difference, multiply, overlay, screen* to simulate the experience suggested and aid manual image registration if appropriate for the dataset. Manual image registration might be necessary if the dataset is large and the user needs to re-align other parts of the dataset that were not intended to be looked at. More blend modes were implemented to allow for explorative visualization and testing purposes. Kokalj and Somrak [34] further advocates for the use of blend modes and layering them multiple times to achieve an easy-to-read image.

Diepenbrock, et al. [14] describes some of the issues they had comparing two arteries in mice. They produce atherosclerotic plaque in one carotid artery by a constrictive cuff and the other carotid artery serves as a control. One of their mentioned shortcomings is the missing ability to view both the left and right carotid artery at the same time with sufficient zoom levels. The solution proposed in this thesis somewhat mitigates this and could, in theory, be used in later studies to show the difference between healthy and normal structures versus abnormal structures in PET/CT imaging. Especially combined with other visualizations such as Angelelli and Hauser [15] technique for straightening tubular flow or Daae Lampe, et al. [16] which is the base for the straight tubular flow technique.

Pagendarm and Post goes further into the issues with this and some solutions to these issues, such as juxtaposition and superimposing images on top of each other using image fusion techniques to highlight the differences. Juxtaposition and superimposing images are further explained in Gleicher et. al. [35] great taxonomy on comparative visualization that boils down all comparative visualization approaches to three different main categories. *Juxtaposition*, i.e. placing images side by side. *Superposition*, i.e. placing images on top of each other. *Excplicit encoding*, i.e. nesting the visualization within another visualization. Depending on how you argue, The Sliding Window approach can be placed in all three of those categories. Superposition by overlaying the two viewports, Juxtaposition via the side-by-side view (see chapter 6.1.3) and explicit encoding by being able to simulate fused PET-CT images (see Figure 6.8).

3.1.3 Volumetric Visualization

Volumetric visualization and its applications go hand in hand with the advancements in computer graphics. Computer graphics have gotten better fast over the last years with the arrival of graphics cards with increased processing power as well as having special cores just to handle ray-tracing. This means that visualizations that previously were considered under-performing are now viable. Volumetric visualizations have been a valuable tool for medical professionals either for planning surgeries or for getting a better understanding of spatial relations within the body.

Volumetric visualizations are nothing new within the medical field or otherwise. Arguably the most popular way of doing volumetric visualization is through Direct Volume Rendering (DVR). DVR is used to render volumetric data with specific properties without extracting any geometric surfaces [36]–[38].

Papers such as [39] describe an early use of raymarching. Which is one of the techniques used today to build a volume from the scanned data. Raymarching is an implementation with an image order approach, meaning that the volume is directly sampled from screen-space viewing rays using raymarching to build the volume.

The other way to implement volume rendering through DVR is by using an object order approach, meaning that the images are sampled in slices which are then reconstructed and blended together.

DVR is divided into three main steps: *reconstruction, classification, and shading*.

3D techniques used in medical imaging can give a quick overview over regions of interest and help localizing where the user is localized within the body on a given slice. A 3D visualization could further be benefitial for research and treatment planning. It does however suffer when overlapping volumes as they may get occluded by the others and thus not being of any real value to either of the volume visualizations [25].

Volume rendering is not used in the Sliding Window due to privacy concerns where researchers could potentially recognize the anonymized patients in a proper 3D render. As well as rendering multiple volumes in 3D also leads to occlusion and visual clutter [25], therefore, we focus on a 2D slice-based approach in this thesis.

3.1.4 Image Fusion and Blend Modes

Using blend modes, alone, in medical visualizations is not a field that is well explored at the time of writing. Blend modes is not to be confused with image fusion as blend modes can be a part of image fusion as filters, while image fusion is combining information from multiple images into a single composite image [40] [34]. Kokalj and Somrak [34] present several different blend modes for visualizing archaeological and geomorphological data. The same principles can also be applied to medical data. In example, the standard way of creating a fused PET-CT is to overlay the PET data, with an appropriate color map, over the CT in grayscale. Thus highlighting the metabolic data within the CT data providing more information to the observers.

Blend modes are a simple way to manipulate images, with layer opacity being the only parameter making them easy to implement and understand. Image fusion has been extensively researched as it is an important step in diagnosing and enhancing features within the ROI. The blending of images used in CT, PET, and MRI can be compared to the standard blend modes implemented in image processing software, as well as the CSS filters implemented in the Sliding Window web app [41]. Although rendering of such images is often accompanied by shaders and smarter algorithms than just blend modes [42]. Image fusion has been extensively researched and proven to improve the interpretability of results, reliability, and diagnostic accuracy. [34] [43]–[47].

Further Kokalj and Somrak focuses on five main blend modes (A is the active layer, B is the background layer):

- **Normal**: Normal is the default, and does nothing with the image, it is simply an image placed on top of another image.
- **Screen**: $1 - (1 - A) * (1 - B)$
  Treats every color channel separately and multiplies the inverse of the two layers, then inverts the produced image. In simple terms, it makes black transparent and white lighter and higher opacity. This process always results in a lighter image.
- **Multiply**: $A * B$
  Multiply is, simply put, the opposite of screen. The resulting image is always a darker image.
- **Overlay**: Overlay combines the two previous approaches. The overlay

blend mode determines if the bottom layer is darker than 50% gray, then applies multiply to darker colors and screen to the lighter colors. This blend mode is non-commutative, meaning the order of the layers influences the resulting image.

- **Luminosity**: Luminosity keep the perceived brightness of the top layer and blends it with the hue and saturation of the bottom layer. This results in the bottom layer colors replacing the top layer colors while keeping the texture and shadows of the top layer.

For illustrations on how this works in practice, see Figure 3.2.

3.2 Situated Visualization

Situated visualization is a method to present data in context, its main feature being the presentation of data representations close to the user viewing the data [48]. Currently situated visualization is most apparent in Augmented Reality (AR), but there are also other use cases for situated visualization other than AR, examples of usecases can be seen in Figure 3.3.

Bressa et. al. name five key perspectives of situated visualization:

- **Space**: Situated visualization as a problem of spatial representation implies that the data has spatial properties such as *location, proximity, distance, and physical structure*. The space perspective focus on the spatial organization and the relationship between the physical environment and the situated visualizations.

- **Time**: With a temporal dimension the situated visualization is a result of the relationship between when the data was recorded and when it is presented. How this is solved varies on a case-by-case basis, depending on if the temporal data is linear, circular, or a variety of other non-linear ways.

- **Place**: Visualizations become situated if they fit into and represent not only relevant data but also the unique features of a particular place, such as the data collected by residents or local cultural heritage.

- **Activity**: The activity perspective of situated visualization implies that the visualizations are embedded and connected to places where humans can interact with the visualization.

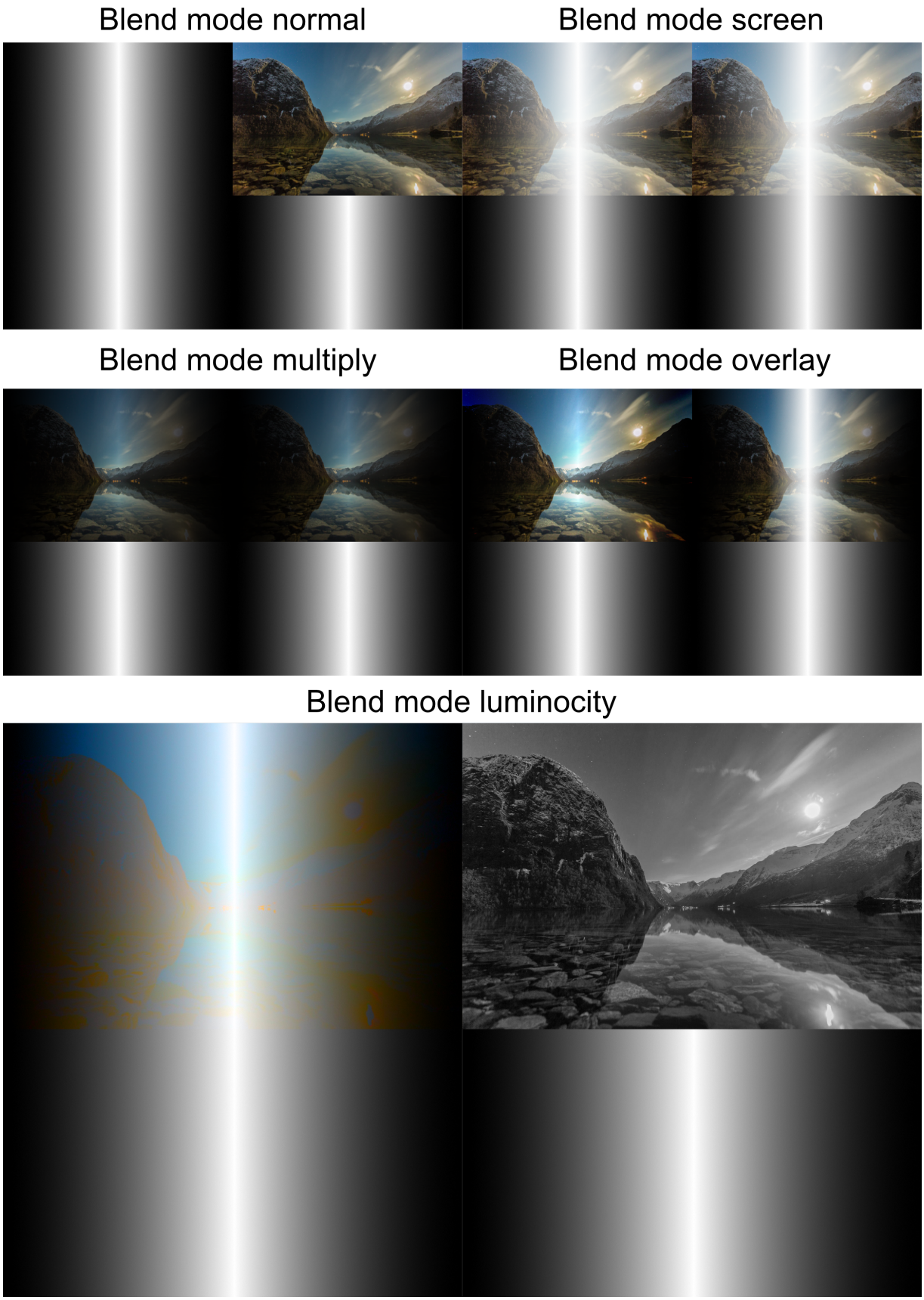


Figure 3.2: Side by side view of the blend modes and the effects it has on the pictures. Left side of each example has the photograph as the top layer, right side of each example has the gradient as the top layer. Photographs by Mathias Bøe.

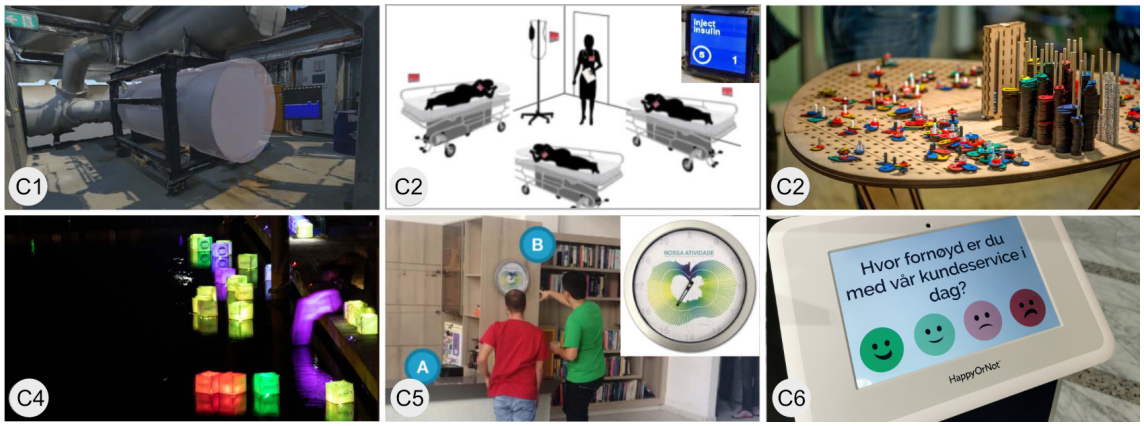


Figure 3.3: A collection of images collected by Bressa et. al. [49]. Case studies: ***C1*** *[50], ©2021 A. Prouzeau;* ***C2*** *Situated Glyphs [51], [52], ©2012 IEEE;* ***C3*** *Cairn [53], ©2021 P. Gourlet;* ***C4*** *Chemicals in the Creek [54], ©2020 W. Campbell/IEEE;* ***C5*** *Activity Clock [55], ©2020 IEEE;* ***C6*** *Public Polling Displays [56], [57].*

- **Community**: The community perspective puts emphasis on the community of people who are the audience and/or co-creators of the visualizations centered around local issues and shared concerns.

The Sliding Window utilizes some of these perspectives. Time, by the nature of the data having a temporal dimension due to follow-up scans. Activity, via the interactive viewport. The Sliding Window arguably also find itself using the community perspective by being available on all devices supporting a web browser. The data could be presented to fellow researchers or to patients through doctor-patient consultations.

Chapter 4

Methodology

The following chapter explains the design choices made in the Sliding Window, the general methodology, application requirements, as well as the methods used to get to the current point in the software.

4.1 Concept Development

The first meetings with Cecilie Brekke Rygh, the project lead for the AML project, were conducted as informal meetings with the goal of identifying problems or hardships within the current medical software or workflow at MMIV. One of the problems that stood out as capable of being improved by visualization was the comparison of time-separated scans. Today this is time-consuming and puts limitations on screen real estate. Screen real estate can be solved by adding more screens and windows, but using the Sliding Window, the hypothesis is that this will take less time as well as reduce the screen real estate required.

This information was then discussed with the supervisor of this thesis to confirm that visualization is a good fit for this issue. From these discussions, the Sliding Window approach to comparing time-separated medical scans was born. There were many intermediate steps before reaching the result, the first one being sketching it out with the Five design Sheet methodology (Figure 4.1).

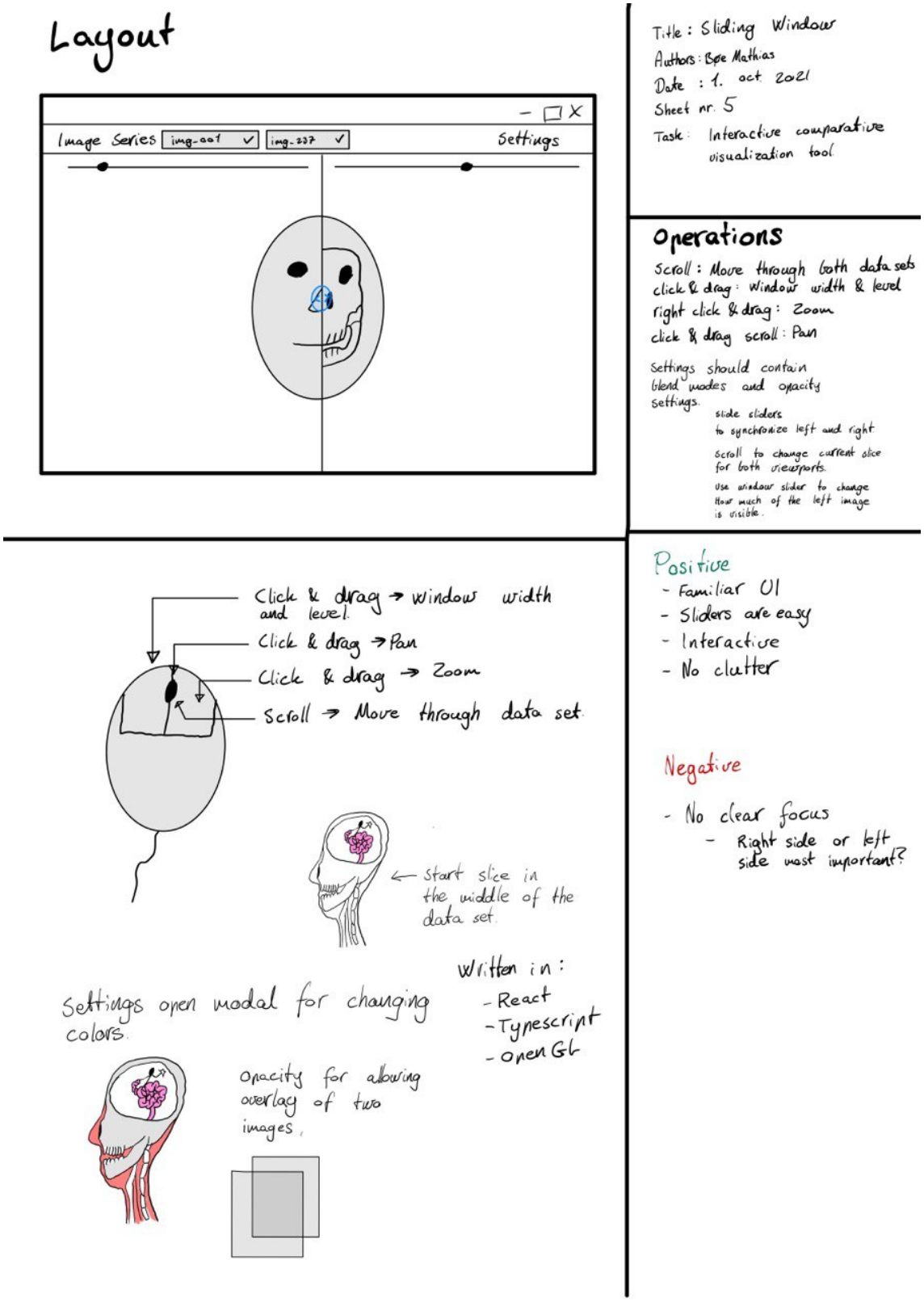


Figure 4.1: The last page of the FdS sketches

4.1.1 Five Design Sheet Methodology

The work by Roberts et al. [58] was the main inspiration for starting the brainstorming process. In the paper, a structured way of brainstorming is proposed called the Five design Sheet (FdS) methodology. The first sheet is for generating ideas, similar to normal brainstorming where all ideas are considered, feasible or not, then filtering out the most promising ones, categorizing them and combining and refining similar ideas, then asking the question: *Do the proposed solutions meet the task requirements*? Some of the more valuable ideas that were put forth were:

- Sliding Window
- Magic Lens
- Volumetric visualization
- Wep app
- Native app

Usually, these design sheets are discussed with the end-user, but for the Sliding Window, the sheets were used more for planning the initial UI and getting an overview of the web app. The reason for this is that the UI itself is primarily dictated by the functionality of the Sliding Window itself, leaving little to no room for changing the UI.

According to Elmqvist and Yi[59] the FdS is a *paper baseline* pattern. In which the researcher gives the end-user a rough sketch of what the result will look like, allowing them to make suggestions and comments on the design before moving on to the next sheet, and implementing the ideas and comments from the end-user. The FdS methodology also contain certain aspects of *do-it-yourself* pattern, as the researcher, in this case, has the final word on the final design of the prototype.

The *do-it-yourself* pattern was, therefore, the base pattern in the development of the Sliding Window, although discussions with Cecilie and the thesis advisor were had throughout the development.

After the initial process of sheet one was done, phase two could start. In sheet two, three, and four the appropriate meta-information and layout were defined. Operations were added and discussions about the advantages and disadvantages

of the core idea of the current iteration were performed. Ideas that were put forth in the second, third and fourth sheets:

- **Layout**: Final sketch of the layout with a navbar and sliding window.
- **Operations**: Scrolling through image series, panning with right mouse button, zoom by pressing and dragging the scroll wheel, changing window levels with left mouse button.
- Central idea of them all being the Sliding Window, where additional information about the data could be accessed at the corners of the viewports.

4.2 Requirements

This section will outline the overall requirements for this application.

R1 - Integrated View For Multiple Image Stacks

There are many websites online that offer many different medical visualizations (i.e. Open Health Imaging Foundation). Several of them are interactive, but none of them put forth a Sliding Window solution like the one proposed in this paper. Interactive viewports are essential to facilitate ease of use as and to make it possible for the user to explore the data set. Therefore, the main requirement of this thesis is to allow the user to interact with The Sliding Window without sacrificing the performance of the application and to further facilitate the visualization of two or more interactive image stacks that enable comparative visualization.

R2 - Support Viewing DICOM Data

The data set consists of scans gathered from the different medical imaging machines in different modalities mentioned in 2.3. It is important for the Sliding Window app to display the data *as is* and not try to change the data set in any way. This is achieved by only parsing the data sets into readable images. Changing the data will also change the visualization; the Sliding Window, therefore, has to assume that the data is accurate. The Sliding Window should also be able to display DICOM files with embedded colors, allowing users to customize colors and visualizations from other image processing software and only make the comparison within the Sliding Window.

R3 - Visual feedback at all times

The frontend of the application should not handle time-consuming work such as handling the DICOM images themselves. It is important to move those tasks to the backend to reduce delay for the user. In the event that loading data or other tasks and processes end up taking a lot of time, proper loading indicators should be displayed. In the event that files are missing or issues appearing, feedback on these errors should be put forward to the user to be able to ask for help. Selected image series should be defined with separate colors to ensure the user knows what data is currently being viewed. The same goes for *color maps, opacity, and blend modes*.

R4 - Portability

The app should be portable to allow any device to view the users data at any point as long as it has access to internet and a web browser. This means that the web-app should be stable enough to open on any device, not that any device should be able to connect to the backend and view all the data.

R5 - Color Blending

The web app should be able to blend colors from multiple datasets to achieve some form of image fusion.

4.3 Development

At the end of the the FdS process, development of an early prototype began. This was done quickly by using normal `.jpeg` images as the sliding window was the central and most vital part of this application. Adjustments for each design step deviating from the original plan in the development process were done as development progressed and new insights were gained. After the initial prototype with a working Sliding Window was complete a new meeting with Cecilie was scheduled to check if this would fit the criteria of the task and discuss options.

The first prototype was accepted as an idea that might help to compare ROIs. There were some additional requests to add more settings, such as changing the currently viewing data set as well as adding support for more viewports. Additional viewports were not added but could be added in the future without much

refactoring. The reason this was omitted is purely due to time constraints. It was also discussed whether or not this tool could be helpful in doctor-patient communication. The feedback on that was positive, considering the app is portable.

Since the feedback was positive, the second iteration of the Sliding Window could start development. The second iteration was mostly setting up the backend and cleaning up bugs and adding small usability upgrades, such as better menus and the ability to scroll through the different data sets. The second iteration of the app is the current state of the Sliding Window.

4.4 Post-Development

After development of the second iteration a case study was performed. The results of this study is discussed in more depth in section 6.3. A pilot study was performed with a domain expert (E0) to gain insight into what would be important to test from the users point of view as well performing a think-aloud evaluation. The study was then conducted on 4 more domain experts using the quantitative scale of software usability, the System Usability Scale (SUS)[60].

4.5 Validation

Validation patterns employed in this project are displayed in Table 4.1. None of the patterns used in this thesis were followed in full, instead, components of various approaches were combined to fit the project and scope. They are put in the categories proposed by Elmqvist and Yi [59].

Paper Baseline was used as a do-it-yourself pattern to make design choices for the development and during the design process, time did not allow to schedule meetings with the domain experts while development was ongoing. Expert reviews were used for the evaluation process through interviews, a pilot study preceding the SUS study with E0, and finally the SUS study itself. The SUS study will be explained in more detail in section 6.3. Early prototypes were also developed to confirm the need for the visualization in the first place, development iterations were done on top of those prototypes and new features were added along the way as needed.

Validation patterns should not be confused with Munzner [61] nested model for visualization design and validation (shown in figure 4.2). For the rest of this

Table 4.1: Validation patterns employed in this thesis

Category	Pattern	Type
Exploration	Do-It-Yourself	Qualitative
	Expert Review	Qualitative
Generalization	Paper Baseline	Quantitative
Validation	Pilot Study	Both
	Prototype	Qualitative

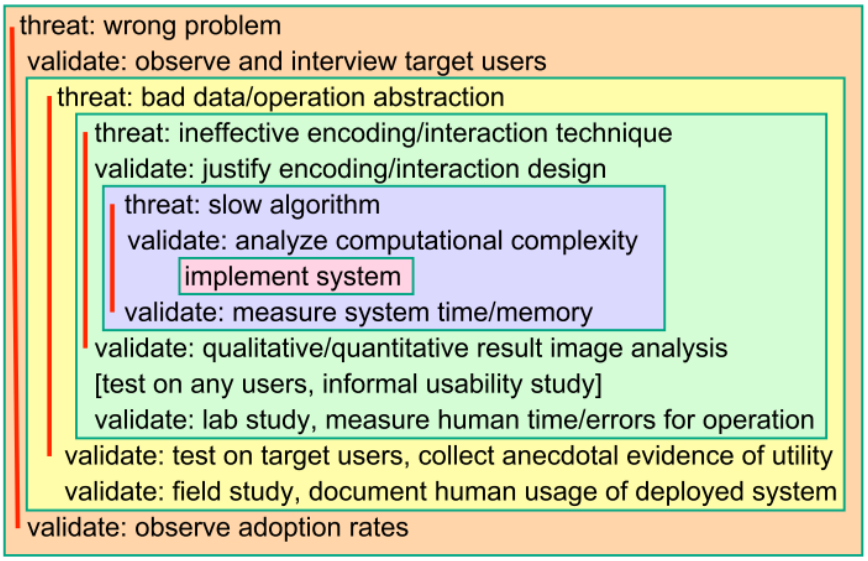


Figure 4.2: The framework proposed in Munzner[61]

section Munzners term is the one used.

Munzners nested model for visualization design was used to help explicitly point out the negative impact of the of bad design choices early on. It also served as a guide and motivation when developing The Sliding Window, making sure development didn't get stuck fixing minor issues instead of solving the main goals of The Sliding Window. The nested model also helped to make sure evaluation of work done, was done throughout the process and not just as a final study at the end.

4.5.1 Threats and Validation

Significant efforts were made to not answer the wrong questions by observing current workflows and talking with domain experts on what issues they face in the current workflow. This was not completely avoided, this is discussed in further depth in chapter 6.

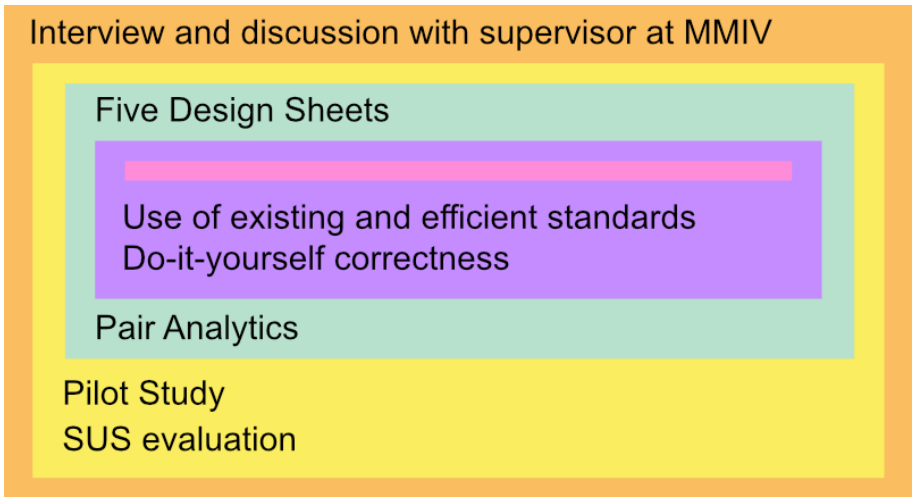


Figure 4.3: Validations performed in each step of the nested model for visualization validation.

The data was already gathered, so the Sliding Window as a solution has to assume that the data is correct. An overview of validations performed can be seen in figure 4.3.

4.6 Situated Visualization

The Sliding Window web app can be considered a situated visualization in many ways. First is the perspective of time, as the change cannot be presented to the user before the second scan is done. The result is then based on the temporal difference between the scans and the treatment response. The Sliding Window web-app also falls under the *place perspective* [49] as one can bring this visualization anywhere on a mobile, tablet, or computer.

Chapter 5

Implementation

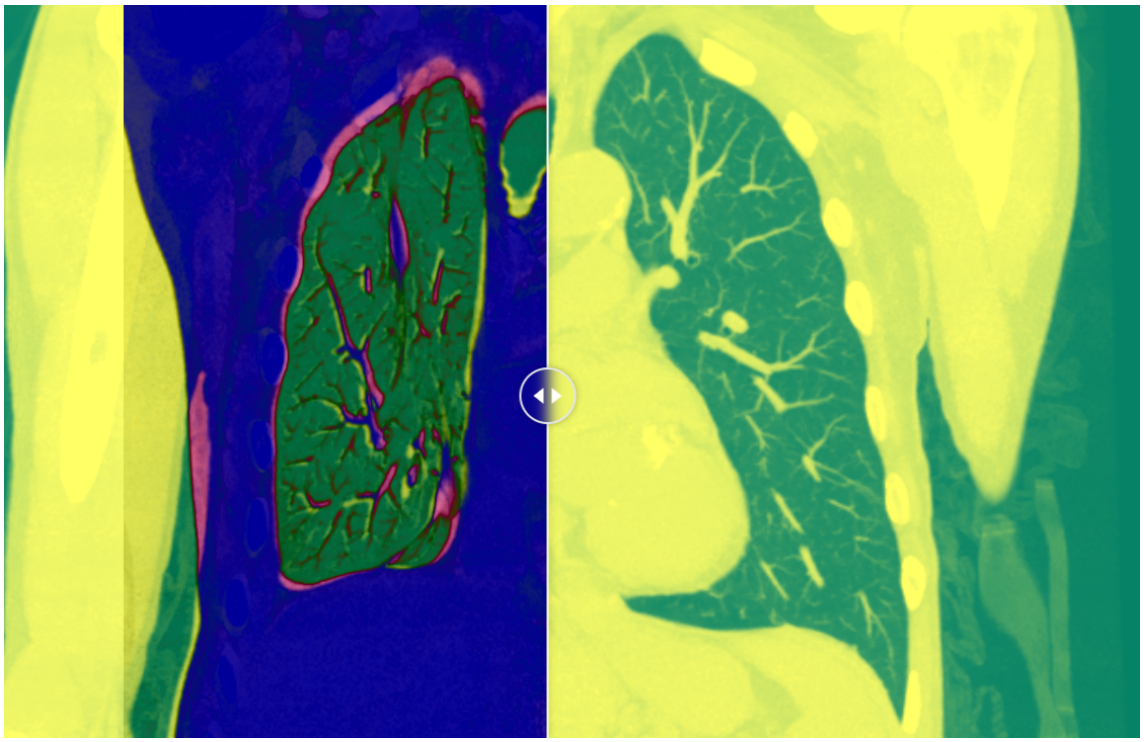


Figure 5.1: The viewport of the Sliding Window web-app

This chapter describes the architecture, technologies, and implementation employed in the Sliding Window and discusses the reasoning behind the choices.

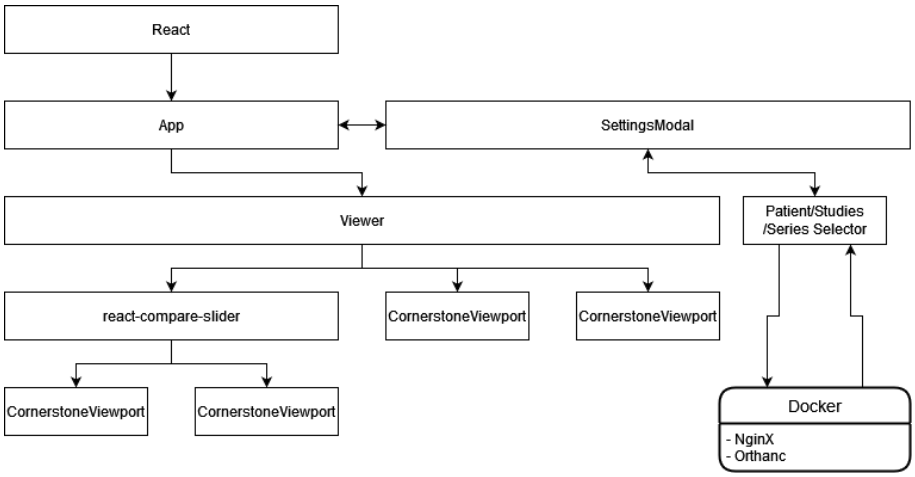


Figure 5.2: Rendering pipeline of the Sliding Window code. The whole web-app is handled by React. It then renders the App component which in turn handles the Viewer component handling the logic.

5.1 Software architecture

Choosing between developing for web or native was not an easy choice for this project. Native has the versatility of developing a high-performing app for either your chosen system or a multi-platform app that can be run on any computer. A web app has one target, the web, which already has multi-platform built-in and the possibility to deploy the app on every device at the same time. Especially developing towards mobile use as well as desktop use is a challenge that is not completely solved yet although functional. Multi-platform development would be a possibility through technologies such as flutter[1], QT[2], electron[3], React Native[4] and more. It could be argued that native apps have better performance as they are not relying on the browser and its capacity, but rather on the machine the app is deployed to. Due to the Sliding Window being a lightweight interface that relies on a backend server, performance is not an issue. This results in portability and multi-platform being the largest criteria for choosing technologies.

System differences are less of a concern, as it is dependent on what browser you use and not the platform you are on. There are still some issues developing for multiple screen sizes. It is a problem that is not completely solved, but the existing solutions are more than good enough for the purpose of the Sliding Window.

[1] https://flutter.dev/
[2] https://www.qt.io/
[3] https://www.electronjs.org/
[4] https://reactnative.dev/

In the early stages of development, *mobile first* [62], [63] was important for developing the frontend. The purpose of mobile-first is to make sure the web app being developed scales well for phones, tablets, and smaller screens. Mobile first works by first scaling the CSS and design for smaller screens, then upscaled with CSS to support larger screens. As development went on, the mobile-first was abandoned in favor of a quicker development workflow for desktop. The web app should still work the same on mobile devices, but the interface might be harder to use.

The code itself has a component-based structure [64], which is considered the gold standard when building web apps with technologies such as React[5] or Angular[6]. The component-based structure is similar to Object-Oriented Programming (OOP) in a purely structural sense, meaning the props get passed around and manipulated in their own *components*. The components themselves are not similar to OOP as they are a combination of abstract data types and functions. Classes could also be of use in some cases, but in the Sliding Window application classes are not used as functional components is the new standard to follow. Functional components (Example in Figure 5.3) were previously considered stateless, but with the addition of Hooks in React 16.8, it is now suggested to use functional components alongside hooks for state management in favor of class-based components (Example of a class-based component can be seen in Figure 5.4).

OOP is widely used in large codebases, although some people have criticized it for being slow and outdated [65]. Johnson[65] has an overview over the positive and negative sentiments regarding OOP, Cook [66] goes further into comparing OOP with abstract data types. React and Angular does not fit into this discussion as it is only similar to OOP in the structural sense and not in the actual code.

In the end, this app ended up being a web app that can run on most modern phones and computers regardless of the operating system. There are several reasons for making the Sliding Window a web app and using the web as its platform. First and foremost is portability. The ability for anyone to pull up the app on their phone without having access to either the data or the app itself, other than through the specified website. The other is familiarity with the web. Most people today know how websites work and how to use them. The third is the community, React and web development has a huge community with many bril-

[5] https://reactjs.org/
[6] https://angular.io/

```
function HelloWorld() {
    return (
        <h1>Hello, World!</h1>
        );
}

// Depending on the use of the component, exporting them
// will be necessary.
// It can either be exported as default or as its name
// Further information on this is available in the docs
export default function HelloWorldDefaultExport() {
    return (
        <h1>Hello, World!</h1>
    );
}
```

Figure 5.3: Functional React component

```
import React from "react";

class HelloWorldClass extends React.Component {
    render() {
        return (
            <h1>Hello, World!</h1>
        );
    }
  }

export default HelloWorldClass;
```

Figure 5.4: Class based React component

liant heads always looking to help others or have solved the same problem before you, making developing for the web faster and sometimes easier. A drawback to this could be the security of an app such as this, for this to be deployed to the public, extra security measures must be done, but integrating this web app as a viewer within already existing web solutions for medical data should, in theory, pose no further risk to owners of the data, as this app is merely a viewer of data and does not manipulate the data in any way.

All of this makes the web a suitable platform for this task [67], [68]. This application is making use of Orthanc, an open-source DICOM server that handles the processing and the web protocol of the DICOM standard such that it will be able to talk to already existing systems at Haukeland Hospital. Orthanc is a powerful but lightweight tool that is able to convert DICOM to standard image formats where applicable but also serves the DICOM files with all the metadata, important when adjusting the slider in the Sliding Window, especially for the CT images. Using this client-server structure we make sure that even weaker phones are able to view the data properly without running into performance issues related to processing the data. The client code however is written in such a way that it should be easy to exchange the API calls to match their already existing solution for the DICOM-web standard. As long as they adhere to the official standard this should not be an issue. Should there however be a lot of custom functionality on top of this standard, the application might not be targeting the right endpoints and names.

If there is an issue with connecting the frontend (client) to the already existing server at the hospital, starting an instance of Orthanc in a docker container[7] will be able to handle this conversion. The code provided contains a barebones Orthanc image with no authentication, so if this were to be used in any official way there would have to be made some changes to the Orthanc docker image. The code also provides a simple configuration for an Nginx[8] image (web server) that works as a proxy for the frontend to connect to. This is to circumvent the *CORS* (Cross-Origin Resource Sharing) errors that would occur without it.

[7] https://www.docker.com/resources/what-container/
[8] https://www.nginx.com/

5.2 Chosen Technologies

When choosing technologies for any software project, it's important to carefully consider which programming languages, algorithms, frameworks, libraries, and other technologies to use. It's important to select the right ones to remove useless overhead in dependencies and to secure the performance of the resulting software. When choosing the technologies for the Sliding-Window these criteria were used, at least one of them had to be fulfilled, listed in order of importance.

1. The technology must allow for rapid prototyping and development.
2. The technology must be relevant and already present in hospitals. If not at a hospital, or in medical research.
3. The technology must be used in the real world, I.e, production environments (see Figure 5.5 for an example of a "Sliding Window" being used in production).

The technologies in the Sliding Window were chosen to make the development process as painless as possible as well as to allow for fast prototyping. The project is about trying an already existing visualization in the medical field to see if it could be a valuable tool. That is why it was opted for already existing solutions for rendering DICOM images as well as the sliding window itself. The sliding window is a solution that is already used in many websites to display changes in static images. This project is using that same approach to medical imaging, where the goal is to figure out the progression of treatment for AML patients.

The following technologies are the ones used in the final version of the visualization.

- Programming Languages
  - JavaScript
  - Node.js
  - yaml
- Frameworks and libraries
  - React
  - Cornerstone.js

Det 50 meter høye spiret man kan se på før-bildet ble torsdag borte etter raset.
ØYVIND SANDNES/ROAR STRØM/NRK

Figure 5.5: Example of the Sliding Window approach being used in a real-world case by NRK [69].

  - cornerstone-tools
  - react-compare-slider
  - Dicom-parser
  - Tailwindcss
- Other tools
  - Docker
  - Orthanc
  - NginX
  - Yarn

5.2.1 JavaScript

JavaScript was chosen because the project works with libraries that do not have type signatures. TypeScript would have been a better choice if these libraries had proper support for it, but as is, the time cost would outweigh the advantages of adding types to all the libraries used. Since TypeScript was not used, PropTypes were used instead to alleviate some of the problems that come with JavaScript.

Especially since typesetting secures a better foundation for the code and makes further development faster and easier. The reason TypeScript would have been a better choice is the fact that it gives proper feedback to code and typings, and it does so when compiling. PropTypes does this as well, but not in the same way. The way PropTypes works is by sending an error message in the web-browser console, which is slower and not the best experience while developing. JavaScript does neither and will just run the code to the best of its ability, making bug fixing and refactoring harder.

Node.js and YAML were chosen based on the languages and technologies already present within the medical field, teams working within the field or doing any form of development will have no problem picking up this project and customizing it to further fit their needs. There is very little YAML in this project, it is only used for configuring the two docker images.

5.2.2 React

React was chosen for the component-based structure, its lightweight nature, and the large user base. It has proven it can do well in production as well as development environments. It is easy to exchange components in the future, meaning that future development and exchanging components for more custom and optimized solutions are pretty straightforward. Thanks to the large user base and developers using React, solutions for handling the scans and data sets already exist. One of these solutions is Cornerstone.js which is employed in this visualization. OHIF has already made an open-source component acting as a viewport that this visualization is based on. Making implementation quick and easy when prototyping. The API for the component is still a work in progress but is good enough to be deployed in the real world through the OHIF Viewer. Cornerstone.js is also the one requiring dicom-parser as a dependency, as you might expect, parsing the DICOM images can be an essential task of the cornerstone viewport to render the resulting image correctly if Orthanc fails to return the data as proper images.

React compare slider was chosen for the ease of integration and unopinionated approach. It's fully customizable and supports rendering components, and *divs*, meaning exchanging the CornerstoneViewport for a new and optimized solution in the future is fast and easy.

Tailwindcss is used for styling and CSS. It was chosen as it compiles small and

is easy to use in both production and development, allowing quick development and prototyping.

5.2.3 Dependencies

To make sure versioning of dependencies is correct and up to date `yarn` is used. This also makes the installation of dependencies on a new machine trivial as you can just run the command `yarn install` or `yarn` to install the dependencies for the project on the target computer. This is important to create a sustainable platform for future development.

There are also other tools available for this purpose, such as *NPM* (Node Package Manager). Both would work with the current config file, but using both at the same time may result in errors. The reason *yarn* was chosen over *NPM* is simply personal preference. In the past, while doing development *NPM* has been finicky, especially when considering environment variables. It was likely due to user error; ironing out those bugs was not worth it for this project, and `yarn` was chosen.

5.3 Backend

Keim, Andrienko, Fekete, *et al.* [68] highlights that we live in a world with an increasing amount of data. They also make the point that most of the time, when data is saved, it is stored without filtering and refinement. That is where Orthanc comes in handy. Orthanc is not the only solution; many hospitals and medical institutions use their own proprietary software or other solutions.

5.3.1 Orthanc

Orthanc is a lightweight, RESTful DICOM server for healthcare and medical research. Orthanc is an open-source project to work as a bridge between multiple DICOM modalities and facilitate research, development, and ease of use. This is important for the Sliding Window project as fast development time is essential. It also provides an easy-to-understand REST-API to make working with the DICOM modalities accessible from a development point of view.

Orthanc has been used by many health researchers and used by open-source projects such as OHIF (open health imaging foundation)[70] to simplify devel-

opment. One of the advantages of using Orthanc is that it can *speak* with remote DICOM modalities through the DICOM network protocol. Thus freeing developers from the low-level API of DICOM, cutting down the learning curve, and speeding up development. [17].

Orthanc was chosen based on its open-source and lightweight nature, making it a good choice when developing and prototyping new visualization solutions.

Orthanc will filter out files that are not supported (images not wrapped in the DICOM standard meta-data such as `.jpg` or `.png`). In its current state, Orthanc can decode raw DICOM files, JPEG DICOM files, and JPEG-LS DICOM files. The supported photometric interpretations are RGB, Greyscale2, and YUV if it is a JPEG derivative.

The backend is based around Orthanc and using Orthanc as a DICOM bridge between the different modalities of DICOM at the hospital. Orthanc can handle many conversions and have no problem using the data received for this project from the research-PACS, categorizing them in Patient, Study, and Series API endpoints.

Orthanc hosts the DICOM files and serves them through a fast and easy RESTful API, resulting in rapid development without requiring a deep understanding of the different DICOM modalities or the DICOM protocol itself. Orthanc provides a simple web interface where users can upload sets of DICOM images, resulting in a very easy-to-use workflow. However, for this project, it was opted to use Slicer-3D[9] to send the data from the research-PACS[10] to Orthanc through the DIMSE protocol since that process is mostly automatic. The process being automatic is very important when you are handling several gigabytes of data. Using the web interface Orthanc provides is limited to uploading one DICOM image series at a time; sending the images through DIMSE poses no such limitation.

5.4 Frontend

The frontend is based on React; React's component-based structure makes sense in this app as the same viewport is used more than once to create the sliding window. The viewports receive the correct data through the *useState* hook to feed the different datasets to the different viewports. The programming language for

[9] https://www.slicer.org/
[10] Research

the frontend is javascript (js). The reason for not using typescript here is that some of the larger libraries being used do not yet support typescript, and there are more issues than benefits to using typescript when the underlying library uses pure js. Strong types are very beneficial when doing any form of development as it makes the debugging process easier. Most of the type values come in the form of different states with either array of strings to the different DICOM image URLs, boolean statements to denote whether or not a patient has been chosen, or to open the settings menu where users can choose different series of images.

Using React for this implementation and YARN (Yet Another Resource Manager) simplifies controlling the versions of libraries installed and makes the installation on new machines relatively painless. This is especially important if this website were to be built on a server and deployed on the intranet at a hospital. There are, however, some issues with using React as the base framework. React has built-in functions and rules that can be hard to circumvent when integrating certain library functions. One of these is the layer functionality in *cornerstoneJs*. *CornerstoneJs* is the main viewport in the web browser and handles a lot of the issues with loading large image sets and caching these in the local storage of the browser, meaning you do not have to load already processed frames every time you change the slice position. Since we cannot use the layer functionality due to the limitations of the library, we are not able to layer PT scans on top of the CT scans, creating the precious multimodal PET-CT image using *cornerstoneJs* alone.

What *cornerstoneJs* does though is allow us to implement additional tools to the viewport fairlry easily by using the cornerstone-tools add-on shown in Figure 5.6.

DICOM images uploaded need pre-processing with software already existing or in use at the hospital to circumvent some of the limitations with the visualization. When exporting to research-PACS, the images will have to be processed to view the relevant data the user would want to compare. This also applies to MIP (Maximum Intensity Projections), which is a requested feature by Cecilie. This is not implemented right into the viewport due to *cornerstoneJs* being a 2D viewport while calculating MIP requires a 3D viewport. This could be solved by integrating something like *vtk-js* into the viewports. A workaround for this is also available by doing the MIP in pre-processing, encoding it directly to the DICOM files. Although changing slice width of the MIP is not possible.

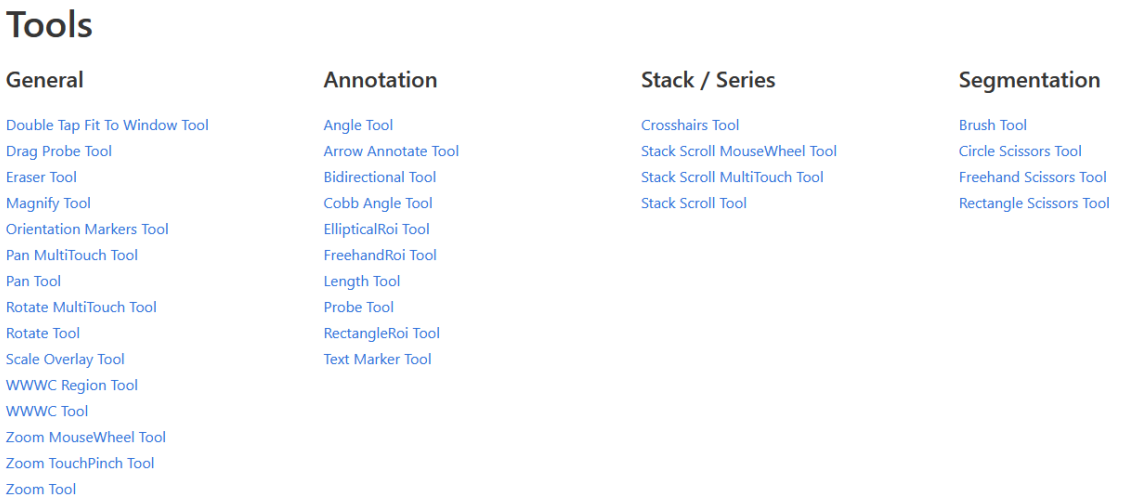


Figure 5.6: Available tools in the cornerstone-tools framework [71].

A shortcoming with this implementation is that changing the visualization after importing it into the Sliding-Window is limited to as-is images, with limited support for interactions and customizations. While this is possible, integrating it into React is very prone to bugs and hard to work with. Pre-processing is, therefore, the best option, at least until the *vtk-js* library matures and better integration with react.

Chapter 6

Results And Discussion

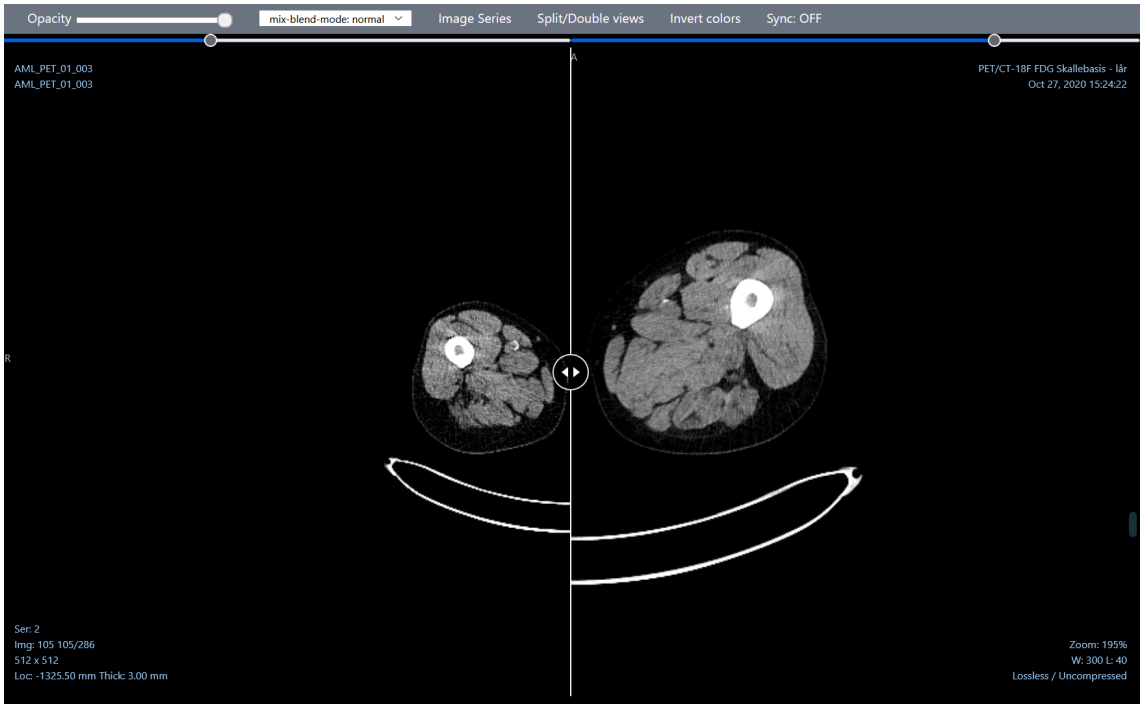


Figure 6.1: An overview of the entire web app

6.1 Interface And Workflow

This section describes the interface options of the Sliding-Window web-app as well as present the intended workflow for the web app.

6.1.1 Patient Selection

A patient can have multiple imaging studies associated with them with different modalities and different time points. Therefore, it is necessary to show a screen where the user of The Sliding Window can first choose which patient to look at. The first screen a user sees when visiting the page is an overview of the patients available from the Orthanc server (see Figure 6.2). Clicking on one of the patients will open the same modal[1] as clicking the Image Series button does, but automatically. When this modal opens up, multiple studies will be available for the patient.

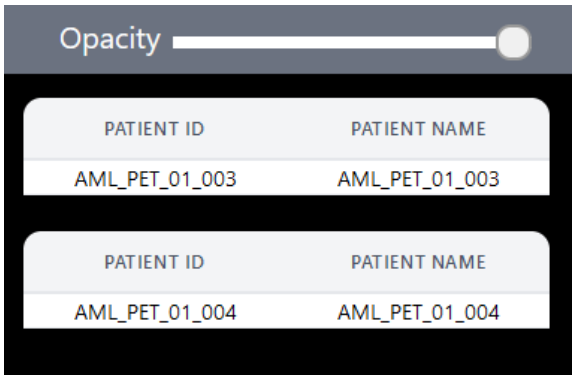


Figure 6.2: A section of the first screen meeting the user in the Sliding Window web-app.

One aspect of patient selection that might seem confusing at first is that as soon as a patient is selected for both left and right viewports it is impossible to change patient without refreshing the page. The reason for this is to minimize the possibility of comparing two different patients by accident. There is still a possibility for this to happen if you select a patient, then select an image series for one of the viewports, close the modal and select another patient, then select another image series for the other viewport. Since you would have to go out of your way to make that happen, fixing that issue with code has not been a priority for this prototype.

6.1.2 Study And Image Series Selection

After selecting the patient, you will be met with the study selection modal (see Figure 6.3). As a user, you could select two time-separated studies in order to look

[1] A modal, window or lightbox, is a web page element or component displaying information in front of other information while deactivating or hiding unneeded components during its lifecycle on the screen. Modals can usually be closed by users.

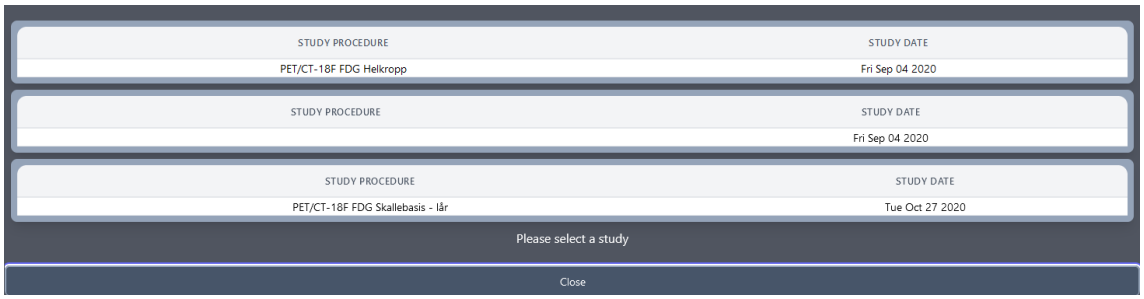


Figure 6.3: The study selection modal

Figure 6.4: The study selection modal when a study is selected. NOTE: This is at the bottom of the modal, study selection from 6.3 is still available to the user at the top of the modal.

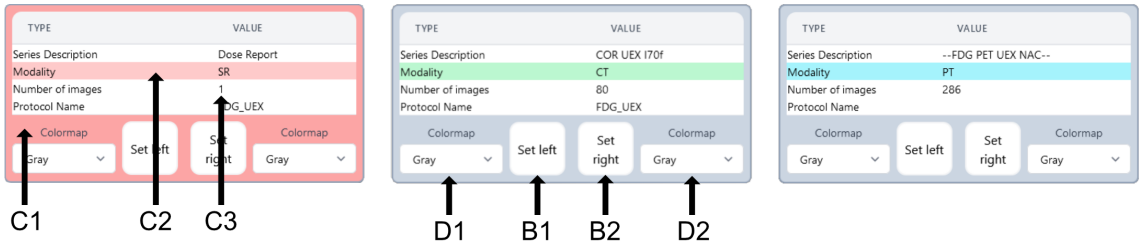


*Figure 6.5: The series cards and colors. **C1**: Background color of the card. **C2**: Modality of the image series. **C3**: Number of images in series. **D1/D2**: Dropdown menu for color maps. **D1**: Left viewport. **D2**: Right viewport. **B1**: Button to select image series for the left viewport, **B2**: Button for the right viewport.*

at the progress of the treatment. Once a study has been selected, more options will be available for selecting the image series of interest (see Figure 6.4. The cards have been color coded to allow the user to find interesting data faster, as well as the different modalities being color coded as well (see Figure 6.5). If the card is red, it means that the number of images in the image series is less than three. Usually the user does not want to look at these image series, but in some cases they may still contain relevant information for another image series the user wants to look at, in example *dosage reports*. The modalities of the image series is also colored to quickly get an overview of the different modalities of the different cards. CT is light green, PT (PET) is cyan, and everything else is red. This could be expanded at a later stage if more modalities of interest were added to the data set.

The intended workflow is to first select the study of interest to view in the left or right viewport, then change studies to get an overview of the image series for the next study, and select the image series to be viewed in the other viewport.

As soon as an image series has been chosen for both viewports, the modal will close, displaying both of the viewports within The Sliding window.

Due to the re-rendering problem with *cornerstoneJs*, if the user wants to change the color map, the user would have to click the *Image series* button, then select the color map of choice, and finally press the *Split/Double views* button in figure 6.6 twice to refresh the viewports. Selecting a new study, image series, or color map can be done at any point from the *Image series* modal.

6.1.3 Navigation

Figure 6.6: The navbar of the Sliding Window

The navbar contains many options for controlling the viewport:

- **Opacity**: This slider changes the opacity of the leftmost viewport of the Sliding Window.
- **Blend modes**: This drop-down allows you to change the blend mode of the leftmost viewport. The available blend modes can be seen in figure 6.8.

- **Image Series**: This button opens a modal that allows you to change which study and image series you are viewing in the viewports.
- **Split/Double Views**: A button that splits the combined sliding window into two separate views for an overview over both image series. If the windows are split it will combine them again. This button also serves a separate purpose of resetting the viewport settings as it resets alignment, window levels, pan, and zoom. This might be necessary from time to time as the current iteration of the application does not allow for changing color maps at runtime. To circumvent this drawback the user would have to press this button twice to re-render the viewport after changing color maps. Making a separate *reset* button could be an improvement, but was left out due to time constraints.
- **Invert colors**: Inverts all the colors in both viewports. This is also done through a CSS filter, this is to circumvent the drawback of using *cornerstoneJs* being that changing color maps or inversion of colors requires re-rendering of the whole component, doing this with CSS allows inversion of colors at any time without resetting the other controls such as zoom, pan, and window levels.
- **Sync OFF/ON**: This button synchronizes the sliders above the viewports. Normally you would use these sliders to align the different slices of a particular image series as they can be moved independently in contrast to the scroll function where you move through the slices of both viewports at the same rate. This button locks them together, meaning you can use the sliders instead of the scroll action to change the slice index of both viewports. The sliders themselves are located on top of the viewports right below the navbar, visible in figure 6.1.

6.1.4 Controls

The viewports currently support five different tools used for aligning and exploring the image series:

- **Zoom**: By clicking and dragging the right mouse button (button 2) the user can zoom in and out of regions of interest.
- **Pan**: By clicking and dragging the middle mouse button (button 3) the user can pan around to align the image series appropriately.

- **Window Width/Level**: By clicking and dragging the left mouse button (button 1), the user can change the window level of the image series. The user can change the window width with horizontal movement of the mouse, and window level with vertical movement.
- **Scroll**: By scrolling in the viewport both of the viewports will change indexes in the image series.
- **Sync sliders**: The synchronization sliders will allow the user to move the index of the viewport independently of each other. This tool can be moved asynchronously or synchronously depending on the state of the Sync button in the rightmost corner of the navbar 6.6.

6.1.5 Viewport

The viewports are rendered independently inside the react-compare-slider component. The vertical line in the middle of the viewport is the cutoff point for the left viewport. The left viewport is always on top in this implementation. At the top left corner of each viewport, the name of the patient is visible. The bottom left has information about the image series and the index of the current image. The top right contains the series description and date of the scan. The bottom left is the image data, window level, window width, and zoom level of the data set.

6.1.6 Blend modes

The blend modes in the Sliding Window will only affect the left viewport although allowing blend modes to be used for both viewports could be of interest. At the current stage of this application, it would not add to the goal of enhancing comparative visualization.

In the Sliding Window, you are able to make a *wax on, wax off*[2] PET-CT visualization, where you can slide between the different modalities of a PET-CT. By placing the PET scan in the right viewport with the color map *PET*, and the corresponding CT scan in the left viewport with the *luminosity* blend mode selected and a grayscale color map (see Figure 6.7). The resulting image is a PET-CT visu-

[2]Refering to the movie Karate Kid where Mr. Miyagi tells his student to put wax on and off on the car. Like the user does when interacting with the sliding window, sliding back and forth, turning the CT (left viewport), "on" and "off" again.

alization where the CT data gets its color from the PET data in the right viewport while keeping the right side as PET activity only.

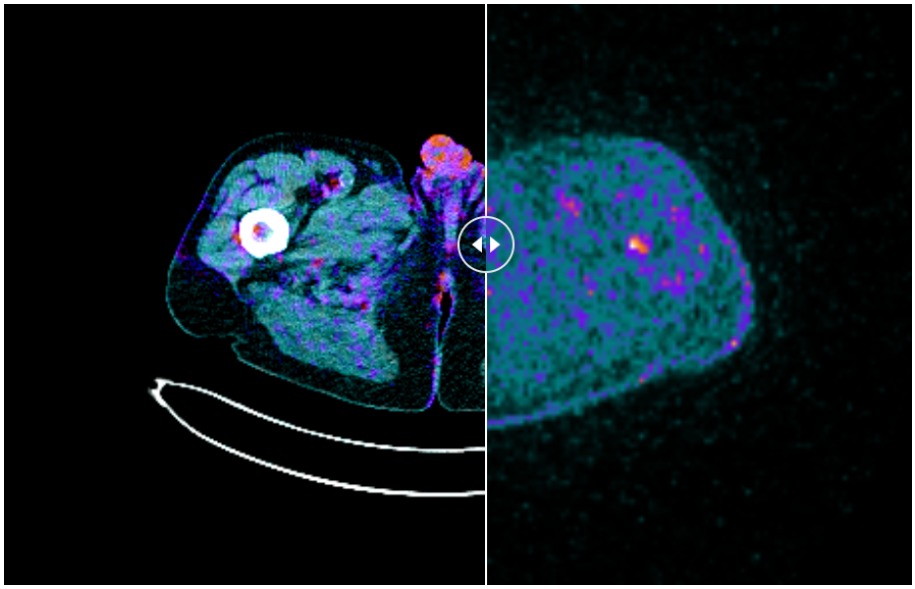


Figure 6.7: The simulated PET-CT visualization showing CT (grayscale and luminocity blend mode) on the left and PET (PET color map) on the right.

This approach does not make a perfect PET-CT as it does not follow the medical standard for this, but it is similar enough to get an idea of the possibilities with blend modes. To get an accurate PET-CT pre-processing is needed to make sure alignment is accurate enough, colors have the right values, and opacity. Currently, this PET-CT view does not follow industry standards other than mimicking it. The Sliding Window web app does assume that the DICOM files provided are the ones needed for diagnosis and evaluation of treatment. Properly pre-processed images can also contain the required colors for the visualization to display the needed information. This means that the color maps and blend modes are not required for the Sliding Window to be a valuable tool, but the option is there to explore novel image blending possibilities in addition to the standard visual representations. A full overview over the implemented blend modes can be seen in figure 6.8.

6.1.7 Color maps

The implemented color maps are taken from the CornerstoneJs library. These color maps were implemented to allow users to view the data in familiar colors. There is, however, a drawback to using these color maps as there is no legend

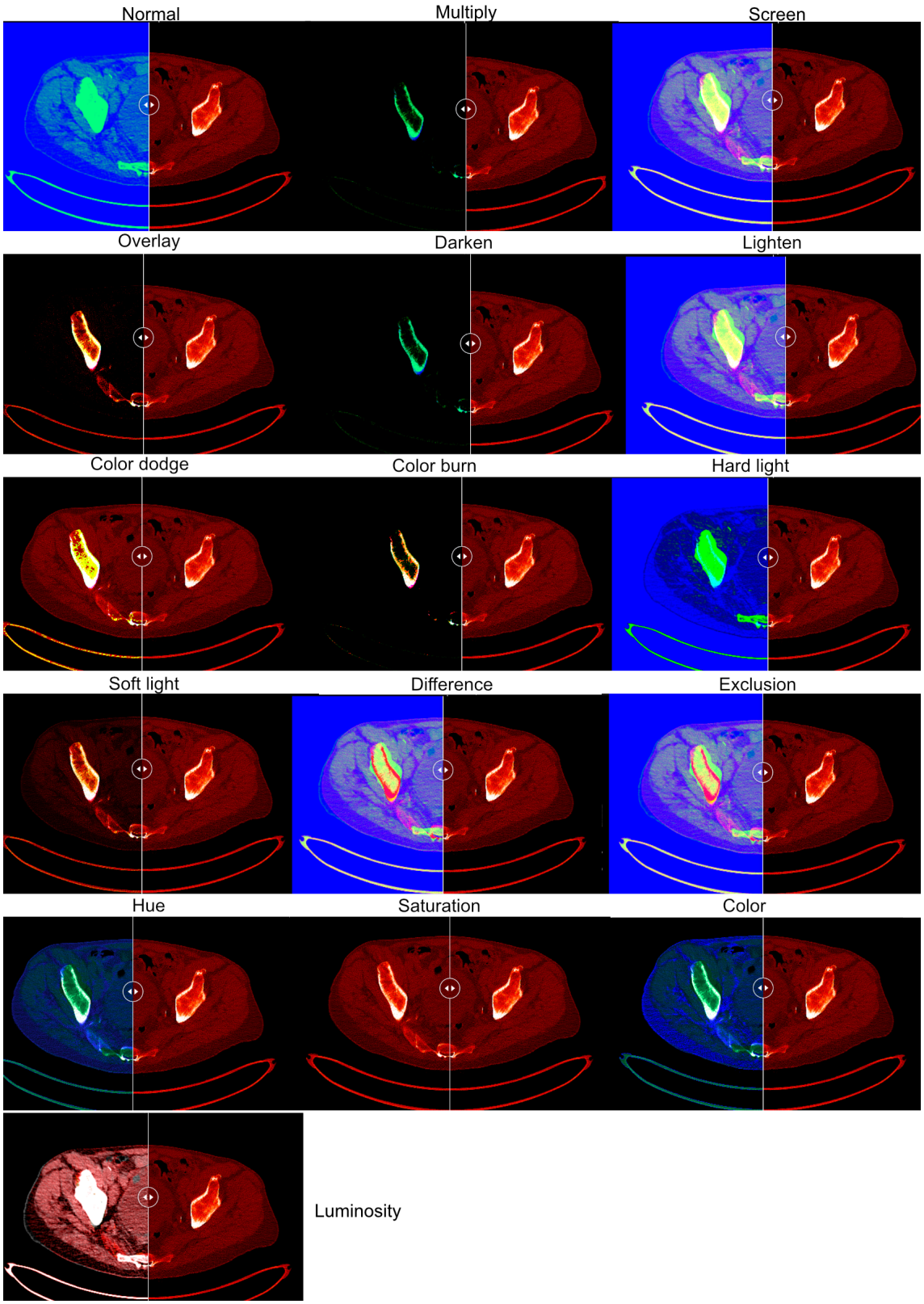


Figure 6.8: Different blend modes implemented with CSS in the Sliding Window web app. On the left, the color map winter was used; on the right, the color map Hot iron was used. This was done just as a visual reference to what the different blend modes do to the image.

implemented in the view. This is a problem that is especially prevalent with the use of rainbow or spectral color maps.

Many color maps are inappropriate for scientific visualizations [72] [34], especially rainbow color maps are troublesome. Even scientists who claim to have adapted to rainbow color maps perform significantly worse when relying on those color maps instead of more perceptually suitable color maps [73] [74]. However, a spectral color map is still implemented at the request of the end users due to their familiarity with this colormap, although it might not empirically offer the best performance in diagnosing patients.

A PET color map is also included with more separation between the colors to allow physicians to create a PET-CT even if the DICOM version of it was not provided in the data set. The PET-CT you can create within the Sliding Window will not be the same as one created with any approved medical software, as it lacks the necessary tools to make it as powerful as its professional counterpart when customizing transfer functions[3], but it is good enough as a proof of concept, especially to show off the possibility of sliding between CT and PET-CT.

6.2 Visualization

The resulting visualization is a robust visualization that can handle a lot of different scenarios and allow the user to explore the data as well as make close comparisons of the data. The Sliding Window visualization also considers the principles of constant visual feedback to make the experience of using the software a smooth one.

6.2.1 Comparative visualization

Recalling 3.1.2 listing the important ways differences can present themselves, this list expands upon that with how differences are most likely to present themselves in the Sliding Window:

- Differences in tracer uptake within cells.
- Patient position.
- Irregularities within the body.

[3] Transfer functions in this thesis refers to adjusting the colors more accurately to display information within the dataset

The visualization of the scans in the Sliding Window must not display differences that are not in the data set, as this can create the false impression of differences where there are none. It is also important that the visualization does not conceal differences. That is one of the reasons the Sliding Window does not have advanced views and has limited support for color maps. Color maps are helpful, but without proper legends and recommendations to prevent users from using bad color maps such as rainbow maps, it is essential to state that the color maps implemented are purely added for exploration of the data set and not necessarily for any diagnostic purposes.

In the Sliding window, a volumetric visualization would be more likely to conceal information as there is a lot more information on the screen. Finding the ROIs would also be more complex with a third dimension. This is not to say that it could be implemented appropriately, but without any automatic image registration to aid in finding ROIs, it would simply not be time-efficient.

The proposed solution in this thesis uses comparison in the way of support for fused PET-CT scans as well as a simulated PET-CT view (Multimodality). It also utilizes standard comparisons, such as the split view, that rely on eye to brain computation and multimodality in the form of allowing two different data sets to be viewed at the same time with tools to align or superimpose pictures on top of each other.

6.3 Evaluation

E0 has a background as a radiologist and is not mentioned as he was the pilot for this study and therefore directly affected the scores of the following participants. The study was first run with E0, resulting in criticisms and tips on how to run the other studies to get the needed results. Good pointers on how to set it up and present the evaluation form to the other experts were given and invaluable for the rest of the evaluation process.

Four additional experts from various fields and domains of computer science. Each expert has experience with either working with medical data or with software for the medical field. E1 has 6 years of experience working with medical images and a visualization researcher. E2 is a computer scientist with 20 years of experience working with medical images. E3 is a senior researcher in MR Physics with 12 years of experience. E4 is a Ph.D. in medical physics.

The study was carried out remotely through a video call in which participants first got an overview of the Sliding Window web app before sharing the controls remotely with the participant. Explanations about how the tools worked and buttons were not given at first, intentionally, to see how users interacted with the web app. They were monitored to see if they got stuck for any particular reason and to ensure that they could use the web app.

At the start of the survey, the participants were shown a simulated PET-CT image with the PET 20 stop color map applied, and the left viewport a CT image with a grayscale color map and the blend mode luminosity. Resulting in an image similar to Figure 6.7. The participants were then allowed to do as they please within the app, testing settings and changing image series. If the users did not know what to do, they got a hint that they could change colors, split view, zoom, etc. In most cases, the users found all the settings asked for in the survey organically while exploring the app. The requirement for this survey was that they had to get a feel for the Sliding Window, allowing them to answer all of the 22 questions in the survey. The users were, however, allowed to return to the Sliding Window if the questions were unclear. The participants never used that option.

Participants were also encouraged to think aloud to express their thoughts while using the web-app and ask questions or come up with suggestions for improvements.

After the initial use of the Sliding Window, they got access to the questionnaire containing 22 questions regarding different aspects of the Sliding Window application. The first 22 questions were based on the System Usability Scale (SUS) designed by Brooke [75]. Additionally, at the end of the questionnaire, there was a freeform evaluation field where the participants could voice suggestions and problems they had. The questions were all scored on a 5-point Likert scale where half of the questions are negatively formed.

6.3.1 Evaluation Results

The results of the survey are shown in Table 6.1. Questions marked with a * are the questions that were asked in the negative form but presented here in their positive form and inverted scores to make it easier to read and understand.

The feedback on the Sliding Window was overall positive, especially the verbal feedback was positive. The feedback was even better for using the Sliding Win-

dow as a diagnostic tool when combined with the PET-CT image, where you can slide on and off the CT layer. Multiple users said they wanted this approach in the software they use today, as it helps them maintain the anatomical position in the body. This is reflected in question number 22.

The feedback on the questions was also overall positive. The lowest average score was 2.5 in question 7, with respect to viewing regions of interest. This might be a result of the interaction of E1 and E2 with the sliding window, as both users did not know what they were looking for. This was explained to the subsequent two users when they asked for it, maybe resulting in a higher score. The highest score on this question is still only 4, meaning there is room for improvement. One aspect that needs improvement, in this case, is the ability to lock zoom levels and panning between the two viewports, as this was a suggested and wanted feature, also probably resulting in the low score of E1 and E2.

The highest averages were question 2 regarding interaction with the web app and question 16, regarding the use case of presenting findings to other people with the web app. One participant suggested animating the Sliding Window slider as this would remove the need for the user to do that when looking for differences or when presenting findings, allowing the user to focus on the visual search alone. It was also mentioned that changing the angle of the Sliding Window would be useful for some cases, as well as implementing a magic lens view.

Drawing in the viewport was also requested as a feature, and then uploading those images to the Sliding Window web app. It would be a nice feature, at least for a temporary drawing in the viewport. However, drawing to compare sizes of ROIs would have to be done in medical-grade software before exporting the DICOMs to the Sliding Window web app in its current iteration.

Most participants were positive about using the Sliding Window in the future to compare medical images, barring some more future developments adding usability features. This includes the ability to zoom, pan, and change window levels of both viewports at the same time.

The study revealed that the target purpose of the application, being able to slide between past and future scans, might not be the most valuable part of the application. Most of the participants in the user study found sliding between temporal dimensions less useful than being able to slide on and off the PET modality of a PET-CT scan. This is reflected slightly in the score difference between questions 22 and 18 and reinforced by the vocalizations of thoughts that the participants

expressed when testing the application. This means that the target function of this application might not be the most suited solution to solve the time issue of comparing scans.

One of the participants vocalized his opinion on the Sliding Window being a web app, finding it to be a neat solution but concerned about the feasibility considering the strict regulations around health data. Especially since this solution is essentially streaming data to your web browser or mobile device. This is a valid concern; a way for this problem to be solved would be to do rigorous testing and secure the data transmission to the device, as well as only allowing specific devices to get a response from the API. Another way to lock it down would be to only host the website on the hospital's internal internet and only allow specific people access to the site with authentication.

*Table 6.1: Response of the participants on a 5-point Likert scale, where **1:**strongly disagree, **2:**disagree, **3:**neither agree nor disagree, **4:**agree, **5:** strongly agree. The stars(*) denote which questions have been changed from their negative form to their positive form; the scores are also negated. The rightmost column is the average score. The bottom row is the SUS score of their respective column. Sus scores has a divergent color map on the score of 68 as this would be considered below average or an unacceptable score [75]–[77].*

	Statements		E1	E2	E3	E4	Average
1	I would like to use the Sliding Window web app for comparing medical images.		4	4	5	4	4.25
2	Interacting with the web-app is easy.	*	4	5	5	5	4.75
3	Differentiating between interesting and uninteresting image series is easy.		3	3	5	4	3.75
4	I was able to align scans easily with the tools provided.	*	4	5	5	4	4.5
5	I was able to use the Sliding Window without any help after the initial instructions.		5	1	5	5	4
6	I see this as a useful tool for doctor-patient communication.	*	2	5	4	5	4
7	The Sliding Window provides enough tools to view regions of interest.		1	1	4	4	2.5
8	The Sliding Window helps me find differences between the two scans more easily.	*	4	5	5	4	4.5
9	The Sliding window view is better than the split view for comparing scans.		3	2	4	5	3.5
10	The sliding window view allows me to zoom in on regions of interest.	*	1	3	5	4	3.25
11	The choice of different color maps was large enough		4	5	5	5	4.75
12	The Sliding Window is fast enough to be useful.	*	4	1	4	5	3.5
13	The tools provided are intuitive and easy to use.		4	3	4	5	4
14	The colors denoting modality were useful when selecting an image series. (C2)	*	4	5	1	4	3.5
15	The color of the card made it easier to find the relevant image series. (C1)		4	5	5	4	4.5
16	The sliding window is nice way to present findings in scans.	*	4	5	5	5	4.75
17	The Sliding Window being a web page, is useful for viewing data from anywhere.		4	5	3	4	4
18	The Sliding Window is useful when comparing two separate CT scans.	*	4	5	3	5	4.25
19	The screen blend mode is helpful for aligning scans.		4	4	5	4	4.25
20	The blend mode difference help me to spot differences in the scans.	*	4	5	5	4	4.5
21	The blend mode luminosity is useful for simulating PET-CT.		3	5	5	3	4
22	It is useful to be able to "turn on and off" the PET layer of a PET-CT scan.	*	3	5	5	5	4.5
	Sus score		62.5	73.9	85.2	85.2	76.7

6.4 Evaluation Conclusion

Overall the Sliding Window approach was well received by domain experts; the resulting application works and is slightly preferred over a split view, according to the case study in question 9 with a score of 3.5. The hypothesis is that the score would be increased if more viewports were added. Figure 6.9 shows an example of what this could look like.

The results point towards this being a valuable feature for medical software. The solution being web-based can be helpful for private hospitals as medical data is more often delivered to the patient, while public hospitals are more reserved about giving out their data for security reasons. Therefore the Sliding Window is a novel approach to a common problem in both sectors, although the solution would probably have to be implemented into existing software for public hospitals so as not to break any rules and regulations such as GDPR (General Data Protection Regulation) [78]. This, of course, counts for the private sector as well, but they are not mandated in the same way to save your personal data for perpetuity (in Norway)[79], allowing them to use more flexible solutions.

For this Sliding Window to be even more useful it was suggested that adding more sliders to change between the modalities would be nice, allowing the user to add their own sliders at different angles, further allowing the user to create their own interactive checkerboard pattern. Although this prototype is beyond the scope of this thesis, it is an interesting idea that warrants future exploration.

6.5 Discussion

The Sliding Window solves the issue with ROIs from two different scans being too far apart when comparing findings. However, it could serve more use cases if *vtkjs* and *itk* was used instead of cornerstone. Being able to explore the data set in three dimensions might be advantageous for other use cases, especially building MIP volumes/images. To get a MIP in the Sliding Window in the current iteration, the DICOMs provided will have to be created from software that can build volumes from the DICOM images and present them with volume rendering. Implementing *vtkjs* should, in theory, be a simple task as there are already guides within that library that guide you through the process of converting existing cornerstone viewports to a *vtkjs* viewport. However, the added development time for this to be a usable product would still increase and would be beyond the

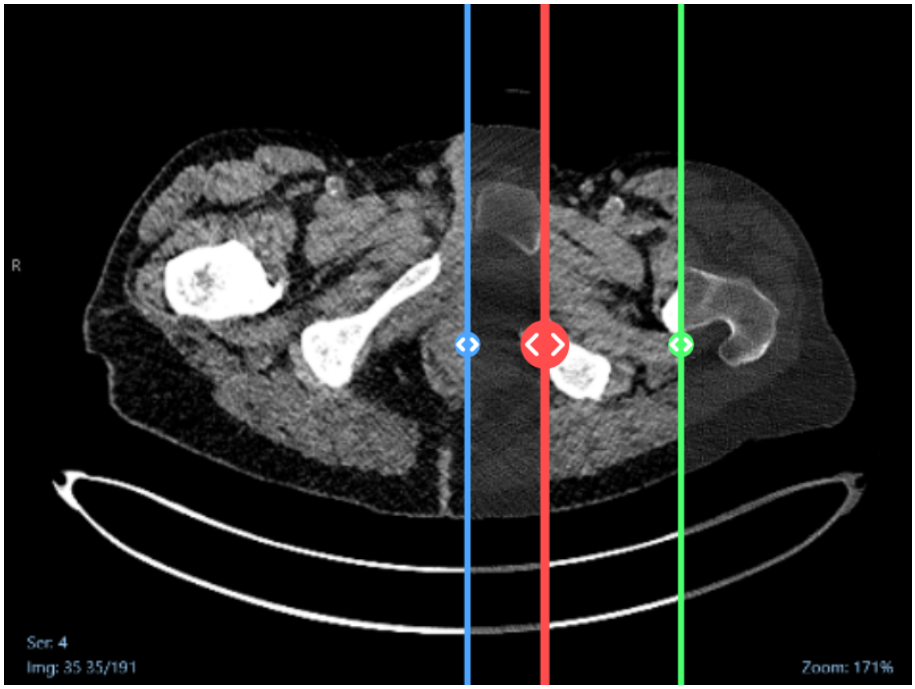


Figure 6.9: Example of how the Sliding Window could look with an additional viewport to compare both the temporal dimension and keep the functionality to slide on and off the second modality. In this figure, the red slider is equivalent to the one used in the current iteration of the Sliding Window, the blue is for the left viewport, and the red is for the corresponding right viewport. Making it a total of four viewports.

scope of this thesis.

There are arguments that can be made for the blend modes not to be included in the final prototype, especially all 16 of them. Blend modes are an area within medical visualization that has not yet been fully explored. Some mentioned papers have been written around the topic, but using the blend modes for different effects is an area that requires more research. That is also the reason why the blend modes are kept as is. As far as relevant blend modes, the screen blend mode has proven very useful when aligning scans while testing the software. It makes it easier to align ROI, such as pubic symphysis, a common registration point for aligning image series. Opacity could be used instead of the screen blend mode, but the blend mode screen has the advantage of reducing opacity only for darker areas and increasing it for lighter areas.

Orthanc as a DICOM bridge for the research PACS has worked flawlessly, it has also proven itself in many real use cases, meaning that Orthanc is a good fit for this prototype. API calls were also designed to be modular, meaning it should be easy to swap them out for any proprietary solution a hospital is using. Although it is easy to swap out, I think using Orthanc in a hospital setting would work just as well, as long as it is used as a bridge and not as a replacement for existing solutions. Current services in use used to handle DICOM files should, in theory, be able to send the image series directly to Orthanc due to DICOM's built-in network protocol. This means that the server should work without any setup other than hosting the docker container, assuming that appropriate authentication methods such as 0auth [80], [81] were also implemented.

6.5.1 Flaws And Fixes

Patient Selection

The reason the user has to refresh the page to choose another patient is to minimize the possibility of comparing two different patients by accident. There is still a possibility for this to happen if you select a patient, then select an image series for one of the viewports, close the modal and select another patient, then select another image series for the other viewport. Since you would have to go out of your way to make that happen, fixing that issue with code has not been a priority for this prototype. Fixing it by adding extra logic to the patient select screen would have solved this issue and allowed the user to change patient on the fly

and remove the wrong patient from the viewport automatically.

Viewport

To see the information of each viewport, the user has to slide the window all the way to the left and right to get the full picture, this could be mitigated by adding a custom component that renders this information to the left and right of the viewports, in that way the information would always be visible. This is not implemented as of now due to the added screen real estate required. Especially on mobile screens. It would, however, be possible to conditionally render the custom component on mobile screens, but was omitted due to time restrictions.

State Handling

One of the biggest issues with the current iteration of the application is the handling of states. The states are handled by the `useState()` hook. Considering there are a multitude of states in this application, it would be worth it to change the states to use the `useReducer()` hook instead. The `useReducer` hook works similarly to the `useState()` hook, the main difference being that it is possible to handle more advanced logic within the `useReducer()` hook. This could further be integrated into the `Context` functionality of React, allowing the developer to refactor the states into a separate file, ultimately making the code more readable, clean, and modular without promoting *prop drilling*[4].

Visual Feedback At All Times

Another issue that might need be the fixed is the information being hidden by the Sliding Window. CornerstoneJs, the viewports in the Sliding Window web-app, *does* support custom components that display the information located in each corner of the viewports. Creating a custom component for this information makes sure the information is always visble and located at the corresponding left or right of the viewport. It would however be more useful in a 3D viewport as you would be able to adjust slice width, which is important information for the doctors. Slice width is not an option in the cornerstone viewport as it only displays the image series *"as is"* only changing the window level and window width. The custom component would be able to show the information at all times, which might be useful, but not vital information.

[4]Passing states into child components that keep passing them down to separate child components. `Context` allows you to skip children that do not need to use the prop

Chapter 7

Conclusions and Future Work

7.1 Conclusion

This thesis presented a novel approach to comparative visualization of temporal data. The presented solution uses two viewports to allow the user to slide between different modalities as well as the temporal dimension. It was explored and tested with the user in mind. The main goal of this thesis, to reduce time spent comparing scans, needs further testing. A study that compares the users time spent comparing scans would have to be performed to get definitive results. However, in the process, we did find a novel solution for keeping the anatomical context of PET scans by allowing the user to slide on and off either the PET or the CT portion of a multimodal PET-CT scan. The user study also points towards this being a valuable solutions and the overall sentiments to the visualization was positive. It is also possible to see from the results that while some scores are low on average, the overall scoring of the evaluation came out to 76.7, which is well above the average of 68.

Most comparisons of temporal data done in the medical field today are either pre-processed images such as the fused PET-CT scan images or side by side in different windows on a computer (rarely, even printed out on sheets). More work needs to explore the utility of interactive visual comparison techniques for multimodal medical imaging data.

Machine learning is a rapidly growing technology and is explored as a way to compare scans. ML might be the future, but approaches need to be trustworthy and transparent, and there will be a need for doctors and patients to verify results

and present them in a way that is easy to understand. Therefore, there will also be a need for visualizations that provide insight into the black-box nature of fully automated methods. Good visualizations *can* become more important as the current generation grows old and needs to be hospitalized, the current generation might not be as trusting with ML as newer generations will be, as they grow up with the technology. The older generation will always have trouble trusting new and unknown solutions to problems, meaning visualizations that explain the science will be important for at least a couple of generations more, if not always.

Advances in technology and especially designing tools allow for rapid prototyping, meaning future visualizations can be designed quicker and tested faster. Full implementations might not be necessary to test a prototype or hypothesis. Therefore, I urge the visualization community to *pick up the mantle* and explore the unexplored aspects of comparative visualization for medical images. There are great foundations to build on.

7.2 Future work

The Sliding Window web-app serves as a prototype to present a concept for how this type of visualization could work within the medical field. There are many directions for this to proceed, one of which is to implement *vtk.js* which would allow the application to render volumetric data. To implement *vtk.js* in this web application and using Orthanc as a DICOM bridge, it would also be necessary to implement *itk*. *itk* would allow the application to load the DICOM files directly from the server and build the volumetric dataset rendered by *vtkjs*.

Another route is to implement the Sliding Window into software already familiar and used by medical professionals. Implementing the Sliding Window into existing software is probably the fastest way to implement this approach in hospitals. New features are easier to get approved than new software in its entirety, integrating The Sliding Window into the research-PACS would be a great first step for making this a reality.

Color maps are also something that can be further improved on within the Sliding Window web-app. The color maps currently implemented are implemented in the same way as the *cornerstoneJs* library implements them. These color maps are not specifically designed for the purpose of diagnosing and evaluating the progress of AML treatment, meaning that with more focus on color maps it would

probably be possible to convey treatment information more appropriately with proper color maps designed for this purpose.

The *react-compare-slider* library used for the sliding window can also be easily switched out with a custom one where you can swap between what image lies on top. This is especially important for the blend modes that are non-commutative. Furthermore I would urge the visualization community to look further into using more blend modes other than the five mentioned by Kokalj and Somrak. Considering the usefulness of the *screen* blend mode when aligning scans it would probably be valuable to enable the screen blend mode as soon as you try to pan one of the viewports. Implementing this would not be difficult. If the user has changed the opacity of the left viewport, the *screen* blend mode might make it too difficult to see what is going on. The same case applies if the user is trying to create their own PET-CT image. Changing the blend mode while aligning might actually harm the user experience. Making this a smart blend mode that only activates at the proper times would make the most sense combine this with appropriate color maps with smart defaults and recommendations and you have a truly interesting topic for future work.

It would be interesting to see how this visualization would perform with sliding windows within the two different viewports. Adding the possibility to view PET-CT or just CT, and then slide between the two viewports to compare the different PET-CT scans while still allowing the user to change the modality of each side with a Sliding Window as well.

Bibliography

[1] S. L. Franconeri, L. M. Padilla, P. Shah, J. M. Zacks, and J. Hullman, "The science of visual data communication: What works", *Psychological Science in the Public Interest*, vol. 22, no. 3, pp. 110–161, 2021, PMID: 34907835. DOI: 10.1177/15291006211051956. eprint: https://doi.org/10.1177/15291006211051956. [Online]. Available: https://doi.org/10.1177/15291006211051956 (cit. on p. 1).

[2] *Acute Myeloid Leukemia | NEJM*. [Online]. Available: https://www.nejm.org/doi/full/10.1056/NEJMra1406184 (visited on 02/13/2022) (cit. on p. 2).

[3] *Acute Myeloid Leukemia - Cancer Stat Facts*, en. [Online]. Available: https://seer.cancer.gov/statfacts/html/amyl.html (visited on 05/06/2022) (cit. on p. 2).

[4] N. Harbeck and R. Wuerstlein, "Truly personalized therapy an end to the era of one size fits all", en, *Nature Reviews Clinical Oncology*, vol. 16, no. 2, pp. 77–78, Feb. 2019, Number: 2 Publisher: Nature Publishing Group, ISSN: 1759-4782. DOI: 10.1038/s41571-018-0165-1. [Online]. Available: https://www.nature.com/articles/s41571-018-0165-1 (visited on 06/07/2022) (cit. on p. 2).

[5] H.-G. Pagendarm and F. H. Post, "Comparative Visualization - Approaches and Examples", en, *Visualization in Scientific Computing*, p. 15, 1995 (cit. on pp. 2, 14, 15).

[6] B. D. de Vos, F. F. Berendsen, M. A. Viergever, H. Sokooti, M. Staring, and I. Igum, "A deep learning framework for unsupervised affine and deformable image registration", en, *Medical Image Analysis*, vol. 52, pp. 128–143, Feb. 2019, ISSN: 1361-8415. DOI: 10.1016/j.media.2018.11.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361841518300495 (visited on 07/26/2022) (cit. on p. 4).

[7] S. Oh and S. Kim, “Deformable image registration in radiation therapy”, *Radiation Oncology Journal*, vol. 35, no. 2, pp. 101–111, Jun. 2017, ISSN: 2234-1900. DOI: 10.3857/roj.2017.00325. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5518453/ (visited on 07/26/2022) (cit. on p. 4).

[8] S. Broggi, E. Scalco, M. L. Belli, *et al.*, “A Comparative Evaluation of 3 Different Free-Form Deformable Image Registration and Contour Propagation Methods for Head and Neck MRI: The Case of Parotid Changes During Radiotherapy”, en, *Technology in Cancer Research & Treatment*, vol. 16, no. 3, pp. 373–381, Jun. 2017, Publisher: SAGE Publications Inc, ISSN: 1533-0346. DOI: 10.1177/1533034617691408. [Online]. Available: https://doi.org/10.1177/1533034617691408 (visited on 07/26/2022) (cit. on p. 4).

[9] B. Zitová and J. Flusser, “Image registration methods: A survey”, en, *Image and Vision Computing*, vol. 21, no. 11, pp. 977–1000, Oct. 2003, ISSN: 0262-8856. DOI: 10.1016/S0262-8856(03)00137-9. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0262885603001379 (visited on 06/07/2022) (cit. on p. 4).

[10] Y. Fu, Y. Lei, T. Wang, W. J. Curran, T. Liu, and X. Yang, *Deep learning in medical image registration: A review*, Oct. 2020. DOI: 10.1088/1361-6560/ab843e. [Online]. Available: https://doi.org/10.1088/1361-6560/ab843e (visited on 06/07/2022) (cit. on p. 4).

[11] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose, “Toolglass and Magic Lenses: The See-Through Interface”, en, p. 8, (cit. on p. 5).

[12] D. L. G. Fill, D. J. Hawkes, Z. Hussain, S. E. M. Green, C. F. Ruff, and G. P. Robinson, “Accurate combination of ct and mr data of the head: Validation and applications in surgical and therapy planning”, en, *Computerized Medical Imaging and Graphics*, 3D Advanced Image Processing in Medicine, vol. 17, no. 4, pp. 357–363, Jul. 1993, ISSN: 0895-6111. DOI: 10.1016/0895-6111(93)90029-M. [Online]. Available: https://www.sciencedirect.com/science/article/pii/089561119390029M (visited on 07/26/2022) (cit. on p. 5).

[13] J. M. Balter, G. T. Y. Chen, C. A. Pelizzari, S. Krishnasamy, S. Rubin, and S. Vijayakumar, “Online repositioning during treatment of the prostate: A study of potential limits and gains”, en, *International Journal of Radiation Oncology*Biology*Physics*, vol. 27, no. 1, pp. 137–143, Sep. 1993, ISSN: 0360-3016. DOI: 10.1016/0360-3016(93)90431-T. [Online]. Available: https://www.sciencedirect.com/science/article/pii/036030169390431T (visited on 07/26/2022) (cit. on p. 6).

[14] S. Diepenbrock, S. Hermann, M. Schäfers, M. Kuhlmann, and K. Hinrichs, "Comparative Visualization of Tracer Uptake in In Vivo Small Animal PET/CT Imaging of the Carotid Arteries", en, *Computer Graphics Forum*, vol. 32, no. 3pt2, pp. 241–250, 2013, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12111, ISSN: 1467-8659. DOI: 10.1111/cgf.12111. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12111 (visited on 03/14/2022) (cit. on pp. 6, 7, 16).

[15] P. Angelelli and H. Hauser, "Straightening Tubular Flow for Side-by-Side Visualization", *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2063–2070, Dec. 2011, Conference Name: IEEE Transactions on Visualization and Computer Graphics, ISSN: 1941-0506. DOI: 10.1109/TVCG.2011.235 (cit. on pp. 6, 16).

[16] O. Daae Lampe, C. Correa, K.-L. Ma, and H. Hauser, "Curve-Centric Volume Reformation for Comparative Visualization", *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1235–1242, Nov. 2009, Conference Name: IEEE Transactions on Visualization and Computer Graphics, ISSN: 1941-0506. DOI: 10.1109/TVCG.2009.136 (cit. on pp. 6, 16).

[17] S. Jodogne, C. Bernard, M. Devillers, E. Lenaerts, and P. Coucke, "Orthanc - A lightweight, restful DICOM server for healthcare and medical research", in *2013 IEEE 10th International Symposium on Biomedical Imaging*, ISSN: 1945-8452, Apr. 2013, pp. 190–193. DOI: 10.1109/ISBI.2013.6556444 (cit. on pp. 7–9, 39).

[18] W. D. Bidgood Jr, S. C. Horii, F. W. Prior, and D. E. Van Syckle, "Understanding and Using DICOM, the Data Interchange Standard for Biomedical Imaging", *Journal of the American Medical Informatics Association*, vol. 4, no. 3, pp. 199–212, May 1997, ISSN: 1067-5027. DOI: 10.1136/jamia.1997.0040199. [Online]. Available: https://doi.org/10.1136/jamia.1997.0040199 (visited on 07/02/2022) (cit. on p. 8).

[19] R. T. Fielding, "In Information and Computer Science", en, p. 180, 2000 (cit. on p. 8).

[20] M. Onken, M. Eichelberg, J. Riesmeier, and P. Jensch, "Digital Imaging and Communications in Medicine", en, in *Biomedical Image Processing*, ser. Biological and Medical Physics, Biomedical Engineering, T. M. Deserno, Ed., Berlin, Heidelberg: Springer, 2011, pp. 427–454, ISBN: 978-3-642-15816-2. DOI: 10.1007/978-3-642-15816-2_17. [Online]. Available: https://doi.org/10.1007/978-3-642-15816-2_17 (visited on 07/01/2022) (cit. on p. 9).

[21] *1ăScope and Field of Application*. [Online]. Available: https://dicom.nema.org/medical/dicom/current/output/chtml/part01/chapter_1.html (visited on 07/01/2022) (cit. on p. 9).

[22] *Current Edition*, en. [Online]. Available: https://www.dicomstandard.org/current (visited on 07/01/2022) (cit. on p. 9).

[23] S. J. Doran, J. dArcy, D. J. Collins, *et al.*, "Informatics in radiology: Development of a research pacs for analysis of functional imaging data in clinical research and clinical trials", *RadioGraphics*, vol. 32, no. 7, pp. 2135–2150, 2012, PMID: 22929148. DOI: 10.1148/rg.327115138. eprint: https://doi.org/10.1148/rg.327115138. [Online]. Available: https://doi.org/10.1148/rg.327115138 (cit. on p. 10).

[24] U. E. Aladl and T. Peters, "Medical Image Registration", in *Multi Modality State-of-the-Art Medical Image Segmentation and Registration Methodologies: Volume II*, A. S. El-Baz, R. Acharya U, A. F. Laine, and J. S. Suri, Eds., New York, NY: Springer New York, 2011, pp. 227–245, ISBN: 978-1-4419-8204-9. DOI: 10.1007/978-1-4419-8204-9_9. [Online]. Available: https://doi.org/10.1007/978-1-4419-8204-9_9 (cit. on pp. 10, 11).

[25] K. Lawonn, N. Smit, K. Bühler, and B. Preim, "A Survey on Multimodal Medical Data Visualization: A Survey on Multimodal Medical Data Visualization", en, *Computer Graphics Forum*, vol. 37, no. 1, pp. 413–438, Feb. 2018, ISSN: 01677055. DOI: 10.1111/cgf.13306. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1111/cgf.13306 (visited on 03/21/2022) (cit. on pp. 10, 13, 17).

[26] C. R. Santos and A. Schulze, "Lipid metabolism in cancer", en, *The FEBS Journal*, vol. 279, no. 15, pp. 2610–2623, 2012, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1742-4658.2012.08644.x, ISSN: 1742-4658. DOI: 10.1111/j.1742-4658.2012.08644.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1742-4658.2012.08644.x (visited on 06/30/2022) (cit. on p. 10).

[27] G. L. Sannazzari, R. Ragona, M. G. Ruo Redda, F. R. Giglioli, G. Isolato, and A. Guarneri, "CTMRI image fusion for delineation of volumes in three-dimensional conformal radiation therapy in the treatment of localized prostate cancer", en, *The British Journal of Radiology*, vol. 75, no. 895, pp. 603–607, Jul. 2002, ISSN: 0007-1285, 1748-880X. DOI: 10.1259/bjr.75.895.750603. [Online]. Available: http://www.birpublications.org/doi/10.1259/bjr.75.895.750603 (visited on 05/24/2022) (cit. on p. 11).

[28] V. S. Khoo, A. R. Padhani, S. F. Tanner, D. J. Finnigan, M. O. Leach, and D. P. Dearnaley, "Comparison of MRI with CT for the radiotherapy planning of prostate cancer: A feasibility study.", en, *The British Journal of Radiology*, vol. 72, no. 858, pp. 590–597, Jun. 1999, ISSN: 0007-1285, 1748-880X. DOI: 10.1259/bjr.72.858.10560342. [Online]. Available: http://www.birpublications.org/doi/10.1259/bjr.72.858.10560342 (visited on 05/24/2022) (cit. on p. 11).

[29] L. Martí-Bonmatí, R. Sopena, P. Bartumeus, and P. Sopena, "Multimodality imaging techniques", en, *Contrast Media & Molecular Imaging*, vol. 5, no. 4, pp. 180–189, 2010, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cmmi.393, ISSN: 1555-4317. DOI: 10.1002/cmmi.393. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cmmi.393 (visited on 05/26/2022) (cit. on pp. 12, 13).

[30] O. Mawlawi and D. W. Townsend, "Multimodality imaging: An update on PET/CT technology", en, *European Journal of Nuclear Medicine and Molecular Imaging*, vol. 36, no. 1, pp. 15–29, Mar. 2009, ISSN: 1619-7089. DOI: 10.1007/s00259-008-1016-6. [Online]. Available: https://doi.org/10.1007/s00259-008-1016-6 (visited on 05/26/2022) (cit. on p. 12).

[31] W. J. Yang, *Handbook Of Flow Visualization*, en. Routledge, Dec. 2018, Google-Books-ID: pMuCDwAAQBAJ, ISBN: 978-1-351-44261-9 (cit. on p. 14).

[32] W. Merzkirch, *Flow Visualization*, en. Elsevier, Dec. 2012, Google-Books-ID: DJCKI5qQdiAC, ISBN: 978-0-08-050658-6 (cit. on p. 14).

[33] O. Dzyubachyk, J. Blaas, C. P. Botha, *et al.*, "Comparative exploration of whole-body MR through locally rigid transforms", en, *International Journal of Computer Assisted Radiology and Surgery*, vol. 8, no. 4, pp. 635–647, Jul. 2013, ISSN: 1861-6429. DOI: 10.1007/s11548-013-0820-z. [Online]. Available: https://doi.org/10.1007/s11548-013-0820-z (visited on 06/07/2022) (cit. on p. 15).

[34] Z. Kokalj and M. Somrak, "Why not a single image? combining visualizations to facilitate fieldwork and on-screen mapping", *Remote Sensing*, vol. 11, no. 7, p. 747, 2019 (cit. on pp. 16, 18, 50, 61).

[35] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts, "Visual comparison for information visualization", *Information Visualization*, vol. 10, no. 4, pp. 289–309, 2011. DOI: 10.1177/1473871611416549. eprint:

https://doi.org/10.1177/1473871611416549. [Online]. Available: https://doi.org/10.1177/1473871611416549 (cit. on p. 16).

[36] N. Max, "Optical models for direct volume rendering", *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99–108, Jun. 1995, Conference Name: IEEE Transactions on Visualization and Computer Graphics, ISSN: 1941-0506. DOI: 10.1109/2945.468400 (cit. on p. 17).

[37] M. Levoy, "Display of surfaces from volume data", *IEEE Computer Graphics and Applications*, vol. 8, no. 3, pp. 29–37, May 1988, Conference Name: IEEE Computer Graphics and Applications, ISSN: 1558-1756. DOI: 10.1109/38.511 (cit. on p. 17).

[38] R. Fernando, Ed., *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*, English, First Edition. Boston: Addison-Wesley Professional, Mar. 2004, ISBN: 978-0-321-22832-1 (cit. on p. 17).

[39] K. Perlin, "~ ComputerGraphics,Volume23, Number3, July 1989", en, p. 10, (cit. on p. 17).

[40] H. B. Mitchell, *Image Fusion: Theories, Techniques and Applications*, en. Springer Science & Business Media, Mar. 2010, Google-Books-ID: D7DXAX6eH2oC, ISBN: 978-3-642-11216-4 (cit. on p. 18).

[41] Y. Ma, "The mathematic magic of Photoshop blend modes for image processing", in *2011 International Conference on Multimedia Technology*, Jul. 2011, pp. 5159–5161. DOI: 10.1109/ICMT.2011.6002127 (cit. on p. 18).

[42] M. Hadwiger and H. Hauser, *First steps in hardware two-level volume rendering*, 2002 (cit. on p. 18).

[43] F. L. Giesel, A. Mehndiratta, J. Locklin, *et al.*, "IMAGE FUSION USING CT, MRI AND PET FOR TREATMENT PLANNING, NAVIGATION AND FOLLOW UP IN PERCUTANEOUS RFA", en, p. 18, 2010 (cit. on p. 18).

[44] Z. Keidar, O. Israel, and Y. Krausz, "SPECT/CT in tumor imaging: Technical aspects and clinical applications", en, *Seminars in Nuclear Medicine*, vol. 33, no. 3, pp. 205–218, Jul. 2003, ISSN: 00012998. DOI: 10.1053/snuc.2003.127310. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0001299803700063 (visited on 05/22/2022) (cit. on p. 18).

[45] Sale, Deepali and Joshi, DM and Patil, Varsha and Sonare, Pallavi and Jadhav, and Chaya, "Image fusion for medical image retrieval", *Citeseer*, vol. 3, pp. 1–9, 2013 (cit. on p. 18).

[46] G. Förster, C. Laumann, O. Nickel, P. Kann, O. Rieker, and P. Bartenstein, "SPET/CT image co-registration in the abdomen with a simple and cost-effective tool", en, *European Journal of Nuclear Medicine and Molecular Imaging*, vol. 30, no. 1, pp. 32–39, Jan. 2003, ISSN: 1619-7070, 1619-7089. DOI: 10.1007/s00259-002-1013-0. [Online]. Available: http://link.springer.com/10.1007/s00259-002-1013-0 (visited on 05/22/2022) (cit. on p. 18).

[47] G. Antoch, J. Kanja, S. Bauer, *et al.*, "Comparison of PET, CT, and Dual-Modality PET/CT Imaging for Monitoring of Imatinib (STI571) Therapy in Patients with Gastrointestinal Stromal Tumors", en, p. 9, (cit. on p. 18).

[48] B. Marques, B. S. Santos, T. Araújo, N. C. Martins, J. B. Alves, and P. Dias, "Situated visualization in the decision process through augmented reality", in *2019 23rd International Conference Information Visualisation (IV)*, 2019, pp. 13–18. DOI: 10.1109/IV.2019.00012 (cit. on p. 19).

[49] N. Bressa, H. Korsgaard, A. Tabard, S. Houben, and J. Vermeulen, "What's the Situation with Situated Visualization? A Survey and Perspectives on Situatedness", *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 107–117, Jan. 2022, Conference Name: IEEE Transactions on Visualization and Computer Graphics, ISSN: 1941-0506. DOI: 10.1109/TVCG.2021.3114835 (cit. on pp. 21, 29).

[50] A. Prouzeau, Y. Wang, B. Ens, W. Willett, and T. Dwyer, "Corsican Twin: Authoring In Situ Augmented Reality Visualisations in Virtual Reality", en, in *Proceedings of the International Conference on Advanced Visual Interfaces*, Salerno Italy: ACM, Sep. 2020, pp. 1–9, ISBN: 978-1-4503-7535-1. DOI: 10.1145/3399715.3399743. [Online]. Available: https://dl.acm.org/doi/10.1145/3399715.3399743 (visited on 05/24/2022) (cit. on p. 21).

[51] F. Kawsar, J. Vermeulen, K. Smith, K. Luyten, and G. Kortuem, "Exploring the Design Space for Situated Glyphs to Support Dynamic Work Environments", en, in *Pervasive Computing*, K. Lyons, J. Hightower, and E. M. Huang, Eds., vol. 6696, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 70–78, ISBN: 978-3-642-21725-8 978-3-642-21726-5. DOI: 10.1007/978-3-642-21726-5_5. [Online]. Available: http://link.springer.com/10.1007/978-3-642-21726-5_5 (visited on 05/24/2022) (cit. on p. 21).

[52] J. Vermeulen, F. Kawsar, A. L. Simeone, G. Kortuem, K. Luyten, and K. Coninx, "Informing the design of situated glyphs for a care facility", in *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, ISSN:

1943-6106, Sep. 2012, pp. 89–96. DOI: 10.1109/VLHCC.2012.6344490 (cit. on p. 21).

[53] P. Gourlet and T. Dassé, "*Cairn*: A Tangible Apparatus for Situated Data Collection, Visualization and Analysis", en, in *Proceedings of the 2017 Conference on Designing Interactive Systems*, Edinburgh United Kingdom: ACM, Jun. 2017, pp. 247–258, ISBN: 978-1-4503-4922-2. DOI: 10.1145/3064663.3064794. [Online]. Available: https://dl.acm.org/doi/10.1145/3064663.3064794 (visited on 05/24/2022) (cit. on p. 21).

[54] L. J. Perovich, S. A. Wylie, and R. Bongiovanni, "Chemicals in the Creek: Designing a situated data physicalization of open government data with the community", *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 913–923, Feb. 2021, Conference Name: IEEE Transactions on Visualization and Computer Graphics, ISSN: 1941-0506. DOI: 10.1109/TVCG.2020.3030472 (cit. on p. 21).

[55] L. A. de Macêdo Morais, N. Andrade, D. M. Costa de Sousa, and L. Ponciano, "Defamiliarization, Representation Granularity, and User Experience: A Qualitative Study with Two Situated Visualizations", in *2019 IEEE Pacific Visualization Symposium (PacificVis)*, ISSN: 2165-8773, Apr. 2019, pp. 92–101. DOI: 10.1109/PacificVis.2019.00019 (cit. on p. 21).

[56] J. Coenen, P. Biedermann, S. Claes, and A. V. Moere, "The Stakeholder Perspective on Using Public Polling Displays for Civic Engagement", en, in *C&T '21: Proceedings of the 10th International Conference on Communities & Technologies - Wicked Problems in the Age of Tech*, Seattle WA USA: ACM, Jun. 2021, pp. 61–74, ISBN: 978-1-4503-9056-9. DOI: 10.1145/3461564.3461585. [Online]. Available: https://dl.acm.org/doi/10.1145/3461564.3461585 (visited on 05/24/2022) (cit. on p. 21).

[57] S. Claes, J. Coenen, and A. V. Moere, "Conveying a civic issue through data via spatially distributed public visualization and polling displays", en, in *Proceedings of the 10th Nordic Conference on Human-Computer Interaction*, Oslo Norway: ACM, Sep. 2018, pp. 597–608, ISBN: 978-1-4503-6437-9. DOI: 10.1145/3240167.3240206. [Online]. Available: https://dl.acm.org/doi/10.1145/3240167.3240206 (visited on 05/24/2022) (cit. on p. 21).

[58] J. C. Roberts, C. Headleand, and P. D. Ritsos, "Sketching Designs Using the Five Design-Sheet Methodology", *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 419–428, Jan. 2016, Publisher: IEEE

Computer Society, ISSN: 10772626. DOI: 10.1109/TVCG.2015.2467271. (visited on 02/02/2021) (cit. on p. 24).

[59] N. Elmqvist and J. S. Yi, "Patterns for visualization evaluation", en, *Information Visualization*, vol. 14, no. 3, pp. 250–269, Jul. 2015, Publisher: SAGE Publications, ISSN: 1473-8716. DOI: 10.1177/1473871613513228. [Online]. Available: https://doi.org/10.1177/1473871613513228 (visited on 07/02/2022) (cit. on pp. 24, 27).

[60] P. W. Jordan, B. Thomas, I. L. McClelland, and B. Weerdmeester, *Usability evaluation in industry*. CRC Press, 1996 (cit. on p. 27).

[61] T. Munzner, "A nested model for visualization design and validation", in *IEEE Transactions on Visualization and Computer Graphics*, Issue: 6, ISSN: 10772626, vol. 15, Nov. 2009, pp. 921–928. DOI: 10.1109/TVCG.2009.111. (visited on 04/20/2021) (cit. on pp. 27, 28).

[62] C. Mullins, "Responsive, mobile app, mobile first: Untangling the UX design web in practical experience", en, in *Proceedings of the 33rd Annual International Conference on the Design of Communication*, Limerick Ireland: ACM, Jul. 2015, pp. 1–6, ISBN: 978-1-4503-3648-2. DOI: 10.1145/2775441.2775478. [Online]. Available: https://dl.acm.org/doi/10.1145/2775441.2775478 (visited on 06/22/2022) (cit. on p. 32).

[63] G. Venkatesh and V. Sridhar, "Mobile-First Strategy for MSMEs in Emerging Markets", *IT Professional*, vol. 16, no. 1, pp. 58–61, Jan. 2014, Conference Name: IT Professional, ISSN: 1941-045X. DOI: 10.1109/MITP.2014.9 (cit. on p. 32).

[64] I. Crnkovic, S. Larsson, and M. Chaudron, "Component-based Development Process and Component Lifecycle", en, p. 7, (cit. on p. 32).

[65] R. A. Johnson, "The ups and downs of object-oriented systems development", en, *Communications of the ACM*, vol. 43, no. 10, pp. 68–73, Oct. 2000, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/352183.352205. [Online]. Available: https://dl.acm.org/doi/10.1145/352183.352205 (visited on 05/24/2022) (cit. on p. 32).

[66] W. R. Cook, "Object-oriented programming versus abstract data types", en, in *Foundations of Object-Oriented Languages*, G. Goos, J. Hartmanis, D. Barstow, *et al.*, Eds., vol. 489, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 151–178, ISBN: 978-3-540-53931-5 978-3-540-46450-1. DOI: 10.1007/BFb0019443. [Online]. Available:

http://link.springer.com/10.1007/BFb0019443 (visited on 05/24/2022) (cit. on p. 32).

[67] D. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann, "Mastering the information age: Solving problems with visual analytics", 2010 (cit. on p. 34).

[68] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon, "Visual Analytics: Definition, Process, and Challenges", in *Information Visualization: Human-Centered Issues and Perspectives*, A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 154–175, ISBN: 978-3-540-70956-5. DOI: 10.1007/978-3-540-70956-5_7. [Online]. Available: https://doi.org/10.1007/978-3-540-70956-5_7 (cit. on pp. 34, 38).

[69] J. Vissgren, *Se bilder før og etter raset på Veslemannen*, nb-NO, Sep. 2019. [Online]. Available: https://www.nrk.no/norge/se-bilder-for-og-etter-raset-pa-veslemannen-1.14690847 (visited on 06/22/2022) (cit. on p. 36).

[70] E. Ziegler, T. Urban, D. Brown, *et al.*, "Open Health Imaging Foundation Viewer: An Extensible Open-Source Framework for Building Web-Based Imaging Applications to Support Cancer Research", *JCO Clinical Cancer Informatics*, no. 4, pp. 336–345, Nov. 2020, Publisher: Wolters Kluwer. DOI: 10.1200/CCI.19.00131. [Online]. Available: https://ascopubs.org/doi/full/10.1200/CCI.19.00131 (visited on 08/13/2022) (cit. on p. 38).

[71] *Cornerstone Tools: Examples*. [Online]. Available: https://tools.cornerstonejs.org/examples/ (visited on 06/22/2022) (cit. on p. 41).

[72] M. Niccoli, "Geophysical tutorial: How to evaluate and compare color maps", en, *The Leading Edge*, vol. 33, no. 8, pp. 910–912, Aug. 2014, ISSN: 1070-485X, 1938-3789. DOI: 10.1190/tle33080910.1. [Online]. Available: https://library.seg.org/doi/10.1190/tle33080910.1 (visited on 05/24/2022) (cit. on p. 50).

[73] M. Borkin, K. Gajos, A. Peters, *et al.*, "Evaluation of Artery Visualizations for Heart Disease Diagnosis", *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2479–2488, Dec. 2011, Conference Name: IEEE Transactions on Visualization and Computer Graphics, ISSN: 1941-0506. DOI: 10.1109/TVCG.2011.192 (cit. on p. 50).

[74] N. Smit, *Rainbow Colormaps What are they good for? Absolutely nothing!*, en-US, Aug. 2012. [Online]. Available: https://medvis.org/2012/08/21/rainbow-colormaps-what-are-they-good-for-absolutely-nothing/ (visited on 03/21/2022) (cit. on p. 50).

[75] j. Brooke, "SUS: A 'Quick and Dirty' Usability Scale", in *Usability Evaluation In Industry*, Num Pages: 6, CRC Press, 1996, ISBN: 978-0-429-15701-1 (cit. on pp. 52, 54).

[76] P. Kortum, C. Z. Acemyan, and F. L. Oswald, "Is It Time to Go Positive? Assessing the Positively Worded System Usability Scale (SUS)", en, *Human Factors*, vol. 63, no. 6, pp. 987–998, Sep. 2021, Publisher: SAGE Publications Inc, ISSN: 0018-7208. DOI: 10.1177/0018720819881556. [Online]. Available: https://doi.org/10.1177/0018720819881556 (visited on 07/14/2022) (cit. on p. 54).

[77] S. C. Peres, T. Pham, and R. Phillips, "Validation of the System Usability Scale (SUS): SUS in the Wild", en, *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 57, no. 1, pp. 192–196, Sep. 2013, Publisher: SAGE Publications Inc, ISSN: 2169-5067. DOI: 10.1177/1541931213571043. [Online]. Available: https://doi.org/10.1177/1541931213571043 (visited on 07/14/2022) (cit. on p. 54).

[78] *General Data Protection Regulation (GDPR) Official Legal Text*, en-US. [Online]. Available: https://gdpr-info.eu/ (visited on 08/13/2022) (cit. on p. 55).

[79] *Slik brukes tjenestene på Helsenorge*, no, Jan. 2020. [Online]. Available: https://www.helsenorge.no/om-tjenestene/slik-brukes-tjenestene-paa-helsenorge/ (visited on 08/13/2022) (cit. on p. 55).

[80] J. Margulies, "Securing Cloud-Based Applications, Part 1", *IEEE Security & Privacy*, vol. 13, no. 5, pp. 96–98, Sep. 2015, Conference Name: IEEE Security & Privacy, ISSN: 1558-4046. DOI: 10.1109/MSP.2015.117 (cit. on p. 57).

[81] *Auth0: Secure access for everyone. But not just anyone.* en. [Online]. Available: https://auth0.com/ (visited on 08/13/2022) (cit. on p. 57).