Dynamic Optimization for Fast Image Transfer and Visualization for Mobile and Stationary Devices: A Performance Evaluation Using Animati Viewer

José Eduardo Venson[1,2], Fernando Bevilacqua[1], Carlos Edmilson da Silva Maia[3], Marcos Cordeiro d'Ornellas[4]

[1]Universidade Federal da Fronteira Sul (UFFS), Chapecó, SC, Brazil

[2]Instituto de Informática (INF), Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, RS, Brazil

[3]Animati - Computação Aplicada à Saúde, Santa Maria, RS, Brazil

[4]Laboratório de Computação Aplicada (LaCA), Universidade Federal de Santa Maria (UFSM), Santa Maria, RS, Brazil

jevenson@inf.ufrgs.br, fernando.bevilacqua@uffs.edu.br,

{edmilsonmaia,marcosdornellas}@gmail.com

Abstract. *A medical application running outside the workstation environment has to deal with several limitations, such as reduced available memory and low network bandwidth. Adaptations and novel approaches are required to make applications overcome such problems. This paper presents an approach that uses a combination of client- and server-side procedures to dynamically optimize the data flow for fast image transfer and visualization on mobile and stationary devices. The main goal of our approach is to minimize the amount of data transferred to and used in the host device without sacrificing the user experience. Our approach was implemented and validated using a real use case, the application Animati Viewer, which is a web visualizer for diagnostic images. The evaluation was measured using metrics such as the accumulated amount of network transferred data and the amount of memory used in the host device. The results show that our approach is feasible and, in one of our tests, it transferred only 7.73% of the amount of data downloaded by the OsiriX mobile.*

1. Introduction

Mobile technology has contributed to the management of chronic diseases, alerting people on medication schedule and improving the efficiency of health systems [West 2012]. For the usage on picture archiving and communication system - PACS, the access to images on mobile devices has broken physical barriers. The software and hardware evolution on portable devices with screens that have ever-increasing resolutions has created new possibilities for diagnostics. A wider access within mobile devices means less time between the exam execution and the medical report, which improves the diagnosis workflow.

This paper presents as its main contribution an approach that uses a combination of client- and server-side procedures to dynamically optimize the data flow for fast image transfer and visualization on mobile and stationary devices. The main goal of our

approach is to minimize the amount of data transferred to and used in the host device without sacrificing the user experience. Despite the resource limitation in the device, the user is still able to perform complex medical actions, such as image windowing. Another contribution is the implementation of our approach in the mobile application Animati Viewer as a use case. It is part of the Animati PACS, a system for distribution and image diagnostics developed by the company Animati - Computação à Saúde from Santa Maria/RS - Brazil.. A set of test cases based on real medical behavior were used to test our approach. One of such tests was performed in the Animati Viewer and the OsiriX mobile, as a simplified way to compare them.

2. Related Work

There are several initiatives to make the diagnostic procedures available beyond the workstation environment. In order to identify such initiatives, we conducted a search using the string "DICOM viewer", selecting works whose results were viewable on desktop and/or mobile devices. Correa et al. (2008) presents a novel approach to extend the access of medical images on mobile devices. They developed a distributed system composed of a web server, responsible for processing computer aided diagnostics (CAD) algorithms on the images, and a mobile device client, which is able to the visualize the results of such process. The combination of both ends provide physicians with a solution containing extra information regarding the diagnosis, not only a visualization tool.

Similarly Pasha et al. (2012) explores the use of mobile and stationary devices as a platform for collaborative discussion of medical images among hospital personnel. The patient images are uploaded from the workstation to a web server, which makes them available to mobile devices in the network. Using a mobile application, which access the web server, the doctor can view DICOM images and collaboratively discuss the diagnosis with colleagues. The problem of limited memory available in mobile devices is mitigated by compressing large DICOM images to JPEG before sending them to the portable devices.

Kaserer (2013) presents a DICOM web viewer able to display uncompressed DICOM files. Even though the viewer is able to run on desktop and on mobile platforms, it has some limitations, such as no integration with PACS systems and the ability to display only uncompressed DICOM images. A more complete and widely used mobile application, however, is the OsiriX mobile, whose implementation is described by Choudhri and Radvany (2011). The application has tools for viewing and processing DICOM images, allowing users to perform several operations as zoom and rotation. All images are downloaded to the application as uncompressed DICOM files, then processed locally for all subsequent actions.

Our approach focus on delivering a visualization solution that works outside the workstation environment and is consistent on a variety of platforms, such as desktops and low- to high-end mobile devices. Our approach constantly uses compressed images in order to minimize data transferring, as opposed to some of the previous works. We also introduce a novel approach that uses a buffer to download a set of images on-demand, favoring JPEG compressed files instead of uncompressed DICOM images when needed to overcome network bandwidth limitations.

3. Mobile Diagnostics

Several technical requirements are needed to make a mobile diagnostic application able to be used for interpretation of medical images. Compared to the environment of a workstation, a mobile application must deal with several constraints such as limited processing power, fewer memory and lower network bandwidth.

Despite such adversities, mobile diagnostics is a valid and trending solution. As pointed by Chandratilleke and Honeybul (2013) , the use of mobile tools can reduce the time gap between the image acquisition and the moment it is viewed by the medical team, decreasing the time a patient has to wait for a diagnosis. De Maio et al (2014) presented the accuracy of mobile diagnostics related to intra-articular knee pathology, concluding that an iPhone DICOM Viewer can be used and it is similar to that of a conventional radiology workstation. Bhatia et al. (2013) also presented that mobile devices such as the Apple iPad can display adequate resolution of CT and MRI sequences to accurately diagnose acute central nervous system injuries and other non-acute pathology. John et al. (2012) suggest that the emergency conditions commonly encountered in CT and MRI can be diagnosed using a portable device DICOM viewer with good concordance to the workstation evaluation. Choudhri et al. 2012 presents similar results regarding the evaluation for acute appendicitis on abdominal CT studies using a mobile device.

In order to create a tool that allows the interpretation of medical images on mobile devices, one has to structure the application and its workflow in such a way to minimize the environment constraints. Compressed images can be used to reduce memory consumption and the download time. The hardware differences among mobile devices and the limited bandwidth, e.g. poor connection in a 3G network, requires the application to adapt to the available computing resources, balancing tasks and strategies in favour of usability and/or performance. The next section presents our approach to make a mobile diagnostic application able to overcome the previously mentioned adversities, which makes it capable of running on different scenarios and devices.

4. Architecture Overview

Our approach to creating a mobile DICOM viewer tries to reduce to a minimum the amount of information downloaded and processed by the application. This goal is achieved by orchestrating two modules: the on-demand downloader manager (ODM) and the dynamic windowing module (DWM). Both modules use compression and cache strategies as well as a combination of client- and server-side processing to reduce the amount of transferred data.

The ODM controls the flow to fetch data from the server, deciding how many images should be downloaded when the user clicks a serie or uses the image selector, for instance. The DWM allows the user to perform windowing operations (e.g. adjust contrast) on the currently selected image, deciding if this operation is best executed locally (client-side) or remotely (server-side). The following sections describe in detail how those two modules work.

4.1. On-demand Download Manager

The on-demand download manager (ODM) acts as a new level in the memory hierarchy. When the user clicks a series, ODM fetches the corresponding images of that series from

the server. After the first one is downloaded, the user can navigate through the images using a slider. In order to save on data transferring and meet any memory constraints, the ODM module will fetch just a set of the images from the server, downloading them as JPEG files (compression quality 75). The number of slots in the download set is based on the ODM buffer size, which depends on the device available memory. According to Wang et al. (2011) the size of a browser cache is about 300MB for desktops, several times bigger than 6MB available in the Android Gingerbread browser cache, for instance. For that reason, the ODM will adjust its buffer size to meet the memory constraints of the host system.

As the user moves the image selector, which changes the currently active image on the screen, called the pivot, the ODM will calculate and decide new fetches. If the pivot is about to exceed the ODM download set boundaries, which means that the user wants to see an image that was not downloaded yet, the ODM will fetch more images from the server. The already downloaded images that are in the set, but far from the pivot (the ones at the beginning of the set, for instance), will be removed from memory by the ODM in order to make room for the upcoming images being downloaded and added to the set. The ODM will always try to make the pivot the central image in the buffer, which means it will fetch images to the left and to the right of the pivot until the buffer is full. As a consequence, the user will be able to navigate back and forth in the image selector while the ODM controls the download process in background. This approach was used to ensure the Animati Viewer would meet the memory constraints of the system it is running on. A CT exam, for instance, typically has 500 images in a study, which accounts for 250MB of data [Pianykh 2009]. Without the ODM, the viewer would have to download all images of the study, which could be impractical on mobile devices, for instance.

4.2. Dynamic Windowing Module

A windowing action can be seen as a contrast adjustment performed in the images of the study being analyzed by the user. It is used to enhance a specific tissue in the images, e.g. bones or muscles [Bourne 2010]. The dynamic windowing module (DWM) allows the user to perform such actions.

The module was designed to minimize the amount of information downloaded during windowing actions, without sacrificing the user experience. The module will work alongside the ODM module performing its actions based on three main variables: waiting time between windowing actions, size of the uncompressed version of the images and available memory in the device. The DWM allows the user to input windowing parameters to adjust the contrast of the currently active image. The user is not allowed to navigate the image selector while the DWM interface is in place. As the user inputs windowing values, the DWM will constantly update the currently active image to present the new contrast selected by the user, so the user can input values until the desired windowing configuration is achieved. The update process can be performed locally (on the client's browser) or remotely (on the server). The DWM will choose between the local and the remote windowing approach based on the size of the uncompressed (raw) version of the image being windowed. If the raw image fits the memory constraints, the DWM will choose a local approach, using the raw image to calculate the adjustments, which allows the application to instantly render all updates as they are inputted by the user. In that case, after the raw image is downloaded, all windowing calculations are performed locally and

the result is instantly rendered to a canvas element placed in front of the active image. A CT study, for instance, has an average size of 512KB for each raw image [Pianykh 2009], so the DWM will perform all windowing adjustments locally for CT studies. Even though the user has to wait for the download of the raw image before inputting windowing values, any adjustment made after the raw is downloaded will be instantly rendered by the DWM. It avoids network traffic while the windowing is in place and allows the user to use a finger or the mouse cursors to adjust contrast dynamically, performing hundreds of windowing operations with no network cost.

If the size of the raw version does not fit the memory constraints, the DWM will choose the remote windowing. In that case, the DWM transmits to the server every windowing value the user inputs. All windowing operations are performed server-side. When the server finishes the current operation, the DWM downloads the result as a JPEG image and presents it to the user. The process is repeated while the user decides which windowing configuration to use. The DWM uses the remote windowing for mammographies, for instance, whose size of a raw image is about 58MB [Pianykh 2009]. After the windowing process is over, the user is allowed to navigate the image selector and use any visualization tools again. From that moment on the new windowing configuration is stored and used to fetch images from the server, so the user is able to analyze the study using the selected windowing. Every time a windowing parameter changes, the DWM has to inform that to the ODM, making it re-download its JPEG compressed images to match the newly defined windowing configuration, otherwise the user sees images featuring the old windowing configuration. The user can input several windowing values before the desired configuration is achieved, so the DWM was designed to trigger the ODM re-download only after the user is done inputting values, not after every interaction. Since the user is not able to interact with the rest of the application when the windowing interface is in place, the DWM can precisely decide when the windowing is over, triggering the ODM re-fetch. The re-fetch is expensive because the ODM will remove from memory all already downloaded images whose windowing configuration is different from the one the user just inputted and re-fetch them using the newly inputted windowing parameters.

5. Results

This section presents the results of our approach, which was implemented and integrated into the Animati Viewer. All data were collected using Google Chrome Developer Tools, which profiled the application while it was running on a desktop and a mobile device. The tests were organized as a set of test cases designed to stress the ODM, forcing it to eventually download all images of a study. Each test case was classified as ”less likely”, ”likely” and ”more likely” to happen according to empirical information provided by medical specialists. The test cases are presented in Table 1.

Case I, classified as likely to happen, occurs when the user opens a study by mistake. Cases II, III, IV, V and VII represent cases where the download set must be refetched several times. Finally case VII happens when the user directly selects a particular image without navigating through all images in the image selector, which is likely to happen depending on the study type. According to our empirical analysis, cases I, III and V are more likely to happen compared in a common medical workflow because the user often inspects all images of a study before providing a diagnosis.

Table 1. Test cases related to image navigation

Case	Movement of image selector	Probability
I	None	likely
II	Navigate from the beginning until the middle	less likely
III	Navigate from the beginning until the end	more likely
IV	Navigate from the beginning until the middle, then back to the beginning	less likely
V	Navigate from the beginning until the end, then back to the beginning	more likely
VI	Navigate from the beginning until the end, then back to the middle	less likely
VII	Direct to a specific image	likely

The test cases were reproduced and validated against real and anonymized data from the Animati PACS database, arranged as three studies: study A, a CT containing 628 images, with an average size of 29.5KB per image; study B, a CT containing 3760 slices, with an average size of 38KB per image; and study C, a CT containing 40 slices, with an average size of 20.5KB per image. The average size of all images was calculated using the JPEG-compressed version of all slices. The amount of memory used in the host device is related to the average size of each image in the study, since the ODM will fetch and keep in memory just a subset of the study images. For that reason, we decided to measure the memory usage by analyzing study B (biggest one) with no specific test cases. When the ODM was disabled, the application downloaded and stored in memory all the study's images, which consumed a maximum amount of 134.5MB of memory in the host device. When the ODM was enabled, its strategy of downloading a few images at any given time instead of them all made the application use a maximum amount of 7.5MB of memory. The ODM re-fetching behavior ensures the Animati Viewer uses a constant amount of memory during the whole time the user is analyzing the study. The amount, in this case, is about 5% of the total size of the study, which is enough to allow the user to navigate the slices without having to wait for any download happening in background.

The accumulated amount of data transferred by the application was measured using different test cases. Figure 1 presents the results when a user performs test cases I, III and V while study A is active. Test case I, when the user opens a study by mistake, and test case III, when the user sees all images of a study, demonstrate that the ODM transferred an amount of data less than or equal to the total size of study, which is the expected result. In test case V, when the user navigates from the beginning of the image selector to its ends then to the beginning again, the ODM transferred more data than the total size of the study. It happens because the ODM will remove images from memory as it fetches new images closer to the pivot in order to honor the memory constraint limit as previously explained.

Test case V was performed again, but this time allowing the browser where the Animati Viewer is running to use its internal cache system, which is not controlled by the ODM. The results, presented as V* in Figure 1 show that the ODM can save even more on network transferring if the browser has the necessary memory to serve the ODM fetch

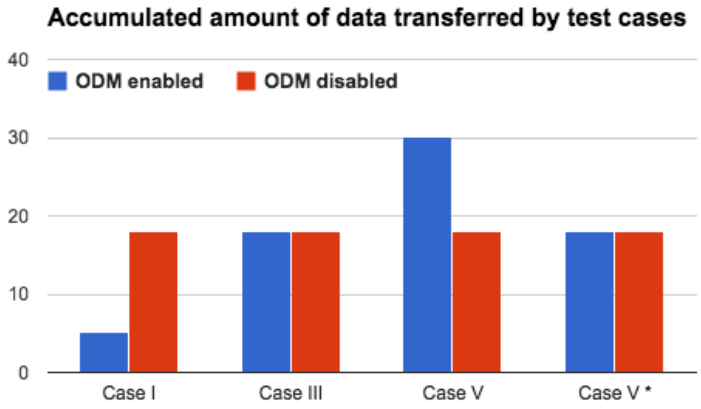

Figure 1. Accumulated amount of data transferred by the application. Y axis: amount of MBs transferred by each test case; X axis: test cases under Study A. Case V* is case V with the browser cache enabled.

requests from the device's cache. Even though test case V* is more efficient, it is dangerous according to our tests, because the browser might experience severe performance penalties due to the constrained amount of available memory.

Finally Figure 2 shows the results achieved by the DWM regarding the accumulated amount of data transferred by the application when the user performs a set of windowing actions, using both local and remove approaches, while study C is active.The remote (server-side) windowing approach causes the application to download an amount of 1.5MB as a result of 72 windowing actions inputted by the user. As previously mentioned, the remote approach requires the download of a new JPEG image from the server for each windowing action, which produces a linear growth in the amount of transferred data. The local approach (client-side), however, requires the download of a single file, which is the uncompressed version of the image being windowed. As illustrated in Figure 2, after the download of the uncompressed file, the accumulated amount of data transferred remains the same for all subsequent windowing actions. If the user performs up to 24 windowing actions, the remote approach transfers less data compared to the local approach, as indicated by intersection point A in Figure 2. After 24 windowing actions, the savings achieved by the local approach becomes evident as the user performs more windowing actions and the accumulated amount of data transferred remains the same.

As an additional way of checking the performance of our approach, the Animati Viewer was compared to the OsiriX mobile. The comparison was based on the accumulated amount of data transferred by the applications while viewing study A (CT with 628 slices) under test case III, followed by a windowing action. OsiriX mobile transferred 314MB at the end of the test, which is the result of downloading all images of the study as uncompressed files [Choudhri and Radvany 2011]. The Animati Viewer transferred 24.3MB at the end of the test, which corresponds to 18.1MB of JPEG compressed images downloaded while the user was analyzing the images, followed by an additional uncompressed image of 0.5MB when the windowing action was performed. After the user had finished the windowing action, the ODM download buffer was invalidated, triggering a re-fetch operation that downloaded additional 5.7MB of JPEG compressed images. The

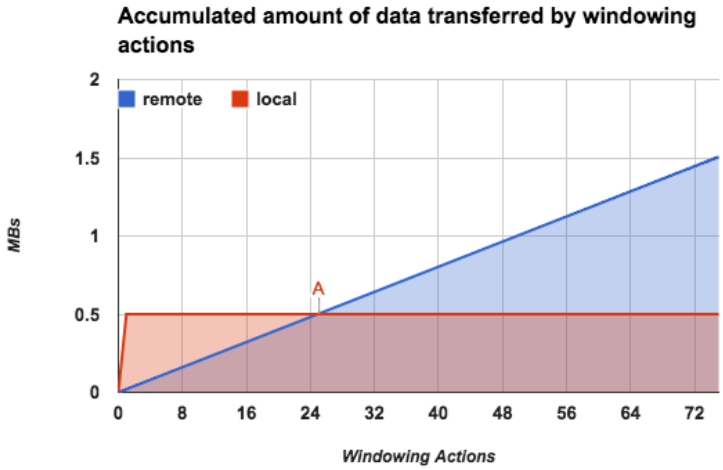

Figure 2. Accumulated amount of data transferred by the application during windowing actions under Study C, highlighting differences between local and remote approaches. Y axis: amount of MBs transferred; X axis: number of windowing actions performed on the image during the test.

re-fetch operation downloaded enough images to fill the ODM download buffer and then stopped, preventing the application from transferring unnecessary images. For this specific test, the Animati Viewer transferred an amount of data that corresponds to 7.73% of the amount transferred by the OsiriX mobile.

6. Discussion

Our results show that a hybrid approach using server- and client-side procedures is able to optimize the data flow. The two modules used in our design, the on-demand download manager (ODM) and the dynamic windowing module (DWM), work together to keep the application within the memory constraints, minimizing data transferring whenever possible. This combination proved to be useful, allowing the application to run on a variety of devices (including mobile ones) without sacrificing the user experience.

The ODM plays an important part in the process of controlling how much memory is used by the application. Its internal buffer dictates how many images will be stored in memory, which allows the application to estimate and adjust its memory consumption based on the available resources. As presented in the results, the amount of memory used by our implementation is constant, which is a key aspect to make the approach suitable to run on resource-constrained devices. The ODM also uses compressed JPEG images in the whole process, which saves on network transferring, making the application capable of being used in situations where the connectivity is not ideal, e.g. 3G networks. Even though the compression process might interfere with the image original data, Kim et al. (2011) shows that compressed medical images have already been used with good results in a system for rapid emergency care via mobile networks, using the JPEG2000 algorithm. The results are encouraging to support the idea of using JPEG images to minimize data transferring without sacrificing medical judgment.

At any moment, the user can view the uncompressed version of an image or adjust its contrast by using the windowing tool. When it happens, our approach relies on the

DWM, which analyzes the size of the uncompressed image and decides the better way to apply the windowing action. It ensures the user is always able to perform windowing actions, even when the network bandwidth is not suitable. As illustrated by the results, the accumulated amount of data transferred by the DWM can be adapted based on the circumstances. Under adverse conditions (e.g. poor connectivity) the DWM will perform all windowing actions in the server, downloading the result as a compressed JPEG, which is not ideal, but fulfills the user request. When conditions are ideal, the DWM will download the uncompressed version of the image and perform the windowing action locally. As previously mentioned, it allows the processing of virtually unlimited windowing actions without network transfers, except the download of the uncompressed image used by the DWM to start the process.

7. Conclusion and Future Work

This paper presented an approach that provides dynamic optimization for fast medical image transfer and visualization suitable for mobile and stationary devices. Our approach was implemented and validated using a real use case, the application Animati Viewer, which is a web visualizer for diagnostic images, part of the Animati PACS. A performance evaluation was conducted using mobile and desktop devices.

Our approach, mainly composed of the on-demand download manager (ODM) and the dynamic windowing module (DWM), proved to be useful to optimize the amount of information transferred by the application. The ODM is able to control how many images are downloaded to and stored in the device, working within its the memory constraints. The DWM allows the user to perform windowing actions, which can be server- or client-side based on the image size. The combined activities and orchestration of both sub-systems allow the application to run on resource-constrained environments, such as those with low network bandwidth or few available memory. The Animati Viewer, after equipped with our solution, was able to minimize the amount of information downloaded using the network, without sacrificing the user experience. Our approach of using a combination of compressed (JPEG) and uncompressed medical images during the process proved to be a feasible solution. In some cases, the user was able to perform virtually an unlimited number of windowing actions with instant feedback, all locally processed and with no network cost past the setup stage. For a specific test case, our approach downloaded only 7.73% of the amount of data downloaded by OsiriX mobile.

As future work we recommend further tests regarding the use of JPEG-compressed images in diagnostics tools. The use of compressed images is a key aspect to keeping the download rate acceptable in network-constrained environments. Another suggestion is a field test involving hospital personnel to evaluate how our approach performs in a medical context. We also recommend a more detailed comparison between our approach and the one used in the OsiriX mobile, since the former is widely used as a mobile diagnosis tool.

8. Acknowledgments

The study and applications of this research are part of the project Appification of Medical Reports - Distribuição de Laudos e Imagens por meio de Apps, developed in Animati - Computação Aplicada á Saúde and financed by the Brazilian Public Agency MCTI/SETEC/CNPq No. 17/2012 - RHAE, Pesquisador na Empresa - Conselho Nacional de Desenvolvimento Científico e Tecnológico/CNPq.

References

Bhatia, A., Patel, S., Pantol, G., Wu, Y.-Y., Plitnikas, M., and Hancock, C. (2013). Intra and inter-observer reliability of mobile tablet pacs viewer system vs. standard pacs viewing station-diagnosis of acute central nervous system events.

Bourne, R. (2010). *Fundamentals of digital imaging in medicine*. Springer Science & Business Media.

Chandratilleke, M. and Honeybul, S. (2013). Modifying clinicians use of pacs imaging. *Journal of digital imaging*, 26(6):1008–1012.

Choudhri, A. F., Carr III, T. M., Ho, C. P., Stone, J. R., Gay, S. B., and Lambert, D. L. (2012). Handheld device review of abdominal ct for the evaluation of acute appendicitis. *Journal of digital imaging*, 25(4):492–496.

Choudhri, A. F. and Radvany, M. G. (2011). Initial experience with a handheld device digital imaging and communications in medicine viewer: Osirix mobile on the iphone. *Journal of digital imaging*, 24(2):184–189.

Correa, B., Ishikawa, E., Ziviani, A., and Faria, M. (2008). Medical image analysis using mobile devices. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, pages 1380–1384, New York, NY, USA. ACM.

De Maio, P., White, L. M., Bleakney, R., Menezes, R. J., and Theodoropoulos, J. (2014). Diagnostic accuracy of an iphone dicom viewer for the interpretation of magnetic resonance imaging of the knee. *Clinical Journal of Sport Medicine*, 24(4):308–314.

John, S., Poh, A. C., Lim, T. C., Chan, E. H., et al. (2012). The ipad tablet computer for mobile on-call radiology diagnosis? auditing discrepancy in ct and mri reporting. *Journal of digital imaging*, 25(5):628–634.

Kaserer, M. (2013). *DICOM Web Viewer*. PhD thesis, Technische Universität Wien.

Kim, D. K., Kim, E. Y., Yang, K. H., Lee, C. K., and Yoo, S. K. (2011). A mobile teleradiology imaging system with jpeg2000 for an emergency care. *Journal of digital imaging*, 24(4):709–718.

Pasha, M. F., Supramaniam, S., Liang, K. K., Amran, M. A., Chandra, B. A., and Rajeswari, M. (2012). An android-based mobile medical image viewer and collaborative annotation: development issues and challenges. *JDCTA*, 6(1):208–217.

Pianykh, O. S. (2009). *Digital imaging and communications in medicine (DICOM): a practical introduction and survival guide*. Springer Science & Business Media.

Wang, Z., Lin, F. X., Zhong, L., and Chishtie, M. (2011). How effective is mobile browser cache? In *Proceedings of the 3rd ACM workshop on Wireless of the students, by the students, for the students*, pages 17–20. ACM.

West, D. (2012). How mobile devices are transforming healthcare. *Issues in technology innovation*, 18(1):1–11.