BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

COMPARISON OF THE EFFECT OF FINGERPRINT IMAGE COMPRESSION ALGORITHMS ON THE QUALITY OF MATCHING ALGORITHMS

POROVNÁNÍ VLIVU ALGORITMŮ KOMPRESE OBRAZU OTISKU PRSTU NA KVALITU POROVNÁVACÍCH ALGORITMŮ

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **MAREK VARGA**
AUTOR PRÁCE

SUPERVISOR **prof. Ing., Dipl.-Ing. MARTIN DRAHANSKÝ, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Varga Marek**
Program: Informační technologie
Název: **Porovnání vlivu algoritmů komprese obrazu otisku prstu na kvalitu porovnávacích algoritmů**
Comparison of the Effect of Fingerprint Image Compression Algorithms on the Quality of Matching Algorithms
Kategorie: Počítačová grafika

Zadání:

1. Seznamte se s kompresními algoritmy pro obrazová data a algoritmy porovnávání otisků prstů.
2. Navrhněte různé kombinace kompresních algoritmů (např. jpeg, wsq), jejich parametrizaci a použití různých filtrů (např. prahování) pro původní obrázek otisku prstu.
3. Implementujte nástroj, který pro daný obrázek nebo celou sadu otisků prstů umožní použití navržených postupů z předchozího bodu.
4. Otestuje implementovaný nástroj za využití databáze různých obrázků otisků prstů a nově získané obrázky otisků prstů porovnejte s původními pomocí dostupných porovnávacích aplikací.
5. Zhodnoťte a diskutujte dosažené výsledky.

Literatura:

- JAIN, Anil K.; FLYNN, Patrick; ROSS, Arun A. (ed.). *Handbook of biometrics*. Springer Science & Business Media, 2007.
- GOLJAN, Miroslav, et al. Effect of compression on sensor-fingerprint based camera identification. *Electronic Imaging*, 2016, 2016.8: 1-10.
- CADD, Samuel, et al. Fingerprint composition and aging: a literature review. *Science & Justice*, 2015, 55.4: 219-238.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Drahanský Martin, prof. Ing., Dipl.-Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 31. října 2019

Abstract

This thesis aims to assess the effect of various compression options and image processing techniques for different types of fingerprint images. Also, various fingerprint matching techniques will be tested to measure the similarity between the processed and the original fingerprint image. The work evaluated the performance of JPEG, PNG, and the WSQ compression, as well as the minutiae-based, cross-correlation, and the PSNR matchers. The fingerprint image processing techniques included image normalization, binarization, depth change, noise removal, and resizing. As a result, the PNG compression recorded the best average scores for all fingerprint types according to all tested matchers. Furthermore, the JPEG compression registered the best average compression times, while the WSQ compression produced the smallest compressed file sizes. Moreover, the fingerprint image processing techniques did not improve the matching scores, but only made them worse.

Abstrakt

Táto práca si kladie za cieľ zhodnotiť vplyv rôznych možností kompresie a techník spracovania obrazu pre rôzne typy snímok odtlačkov prstov. Kvalita kompresných metód sa otestuje pomocou rôznych techník porovnávania odtlačkov prstov na meranie podobnosti medzi spracovaným a pôvodným obrázkom odtlačku prsta. Práca hodnotila výkonnosť kompresií JPEG, PNG a WSQ, ako aj porovnávače založené na markantoch, krížovej korelácií a PSNR. Techniky spracovania odtlačkov prstov zahŕňali normalizáciu obrazu, binarizáciu, zmenu bitovej hĺbky, odstránenie šumu a zmenu veľkosti obrazu. Výsledkom bolo, že kompresia PNG zaznamenala najlepšie priemerné skóre pre všetky typy odtlačkov prstov podľa všetkých testovaných porovnávačov. Okrem toho kompresia JPEG zaznamenala najlepšie priemerné kompresné časy, zatiaľ čo kompresia WSQ priniesla najmenšie komprimované veľkosti súborov. Techniky spracovania snímok odtlačkov prstov nezlepšili skóre zhody, iba ich zhoršili.

Keywords

fingerprint, compression algorithms, JPEG, PNG, WSQ, fingerprint matching, compression effect, image quality

Kľúčové slová

obraz odtlačku prsta, metódy kompresie, JPEG, PNG, WSQ, porovnanie odtlačku prsta, vplyv kompresie, kvalita obrazu

Reference

VARGA, Marek. *Comparison of the Effect of Fingerprint Image Compression Algorithms on the Quality of Matching Algorithms.* Brno, 2020. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor prof. Ing., Dipl.-Ing. Martin Drahanský, Ph.D.

Rozšírený abstrakt

Identifikácia osôb na základe odtlačkov prstov je jednou z najzákladnejších biometrických techník. Nato, aby mohla byť osoba úspešne identifikovaná, sa musí v čase porovnania, zhodovať jej odtlačok prsta s porovnávaným snímkom odtlačku v databáze. Takáto databáza, ktorá obsahuje množstvo obrázkov odtlačkov prstov, musí byť niekde umiestnená. Starať sa o takúto databázu môže byť nákladné jednak z časového hľadiska nahrávania a sťahovania obrázkov, a jednak z hľadiska platenia za takéto úložisko. Riešením môže byť použitie kompresie. Skomprimované obrázky zaberajú menej miesta a prenos takýchto obrázkov zaberie menej času. Táto práca sa zameriava na porovnanie rôznych kompresných algoritmov, ich parametrov, a ich vplyvu na rôzne typy obrázkov odtlačkov prstov. Každý skomprimovaný obrázok je porovnaný na zhodu so svojim originálom podľa rôznych porovnávacích algoritmov. Pred kompresiou možno upraviť odtlačok prsta pomocou filtra, ktorý môže zlepšiť výsledok porovnania, alebo urýchliť čas kompresie. Za účelom tejto práce bol implementovaný nástroj s grafickým užívateľským rozhraním, ktorý dokáže načítať obrázok odtlačku prsta, alebo celý priečinok odtlačkov prstov. Následne je možné použiť jeden z filtrov pre odtlačky prstov, konkrétne ide o normalizáciu obrazu, binarizáciu s lokálnym prahom, odstránenie šumu, zmenu bitovej hĺbky, alebo zmenu veľkosti obrazu. Nato je možné použiť jednu z podporujúcich kompresií, konkrétne JPEG, PNG, alebo WSQ. Pre kompresie JPEG a PNG sa dá zvoliť kompresný pomer, a pre kompresiu WSQ sa dá zvoliť rýchlosť kódovania. Skomprimovaný obrázok možno porovnať s jeho originálom na základe porovnania podľa markantov, krížovej korelácie alebo PSNR. Výsledná aplikácia bola implementovaná v jazyku Java s využitím knižnice JavaFX pre tvorbu užívateľského rozhrania. Toto užívateľské rozhranie nie je nevyhnutné spustiť, pretože aplikácia podporuje načítanie súboru typu XML, ktorý obsahuje informácie o zdrojovom priečinku s obrázkami odtlačkov, kompresiami a filtrami, ktoré aplikovať, a aj porovnávacie techniky, podľa ktorých budú následne obrázky porovnané. Výsledky práce ukázali, že kompresia PNG zaznamenala najlepšie priemerné skóre pri všetkých druhoch algoritmov porovnania. Avšak, pre reálne a syntetické odtlačky všetky kompresie s použitými filtrami, ale aj bez nich, zaznamenali, na základe porovnania podľa markantov, priemerné skóre, ktoré je minimálne dva krát vyššie, ako odporúčaná hodnota pre zhodu. Pre ostatné typy odtlačkov, choré, poškodené a falošné, zaznamenali iba kompresie JPEG a PNG, bez použitých filtrov, priemerné skóre, ktoré by sa dalo považovať za zhodu. Na základe porovnaní pre krížovú koreláciu zaznamenala iba kompresia PNG priemerné skóre, ktoré by indikovalo zhodu. Aplikovanie rôznych techník pre spracovanie obrazu, alebo filtrov, neprinieslo očakávané výsledky. Priemerné skóre pre varianty, pre ktoré neboli použité filtre, je oveľa vyššie, ako pre varianty, pre ktoré boli použité filtre. Naproti tomu, aplikácia filtrov pozitívne ovplyvnila priemerný čas kompresie a priemerné veľkosti skomprimovaných obrázkov. S použitými filtrami klesol priemerný čas kompresie aj veľkosť skomprimovaných obrázkov. Najviac sa o to zaslúžili filtre, ktoré zmenili hĺbku obrázku na 1 bit, alebo zmenšili obrázok. Ďalej sa zistilo, že hocijaká kombinácia kompresie, jej parametrov a filtra dokáže vyprodukovať nulovú zhodu s pôvodným obrázkom. To môže byť spôsobené nízkou kvalitou vstupného obrázku odtlačku prsta, ale aj spracovaním snímku odtlačku. Algoritmy pre porovnanie majú vo všeobecnosti problém s odtlačkami v nízkej kvalite, alebo s odtlačkami, pre ktoré nevedia nájsť dostatočné množstvo záchytných bodov, markantov. Taktiež, pre implementáciu techník spracovania obrazu boli použité funkcie knižnice OpenCV. Konverziou reprezentácie obrazu v jazyku Java na reprezentáciu obrazu pre OpenCV mohlo dôjsť k strate údajov a teda optimálna funkcionalita týchto filtrov nemusela byť zaistená.

Comparison of the Effect of Fingerprint Image Compression Algorithms on the Quality of Matching Algorithms

Declaration

Hereby I declare that I have developed this bachelor's thesis autonomously under the guidance of Mr. Drahanský. All the relevant information sources, which were used during the preparation of this thesis, are properly cited and included in the list of references.

.....
Marek Varga
June 3, 2020

Acknowledgements

First and foremost, I would like to thank my supervisor prof. Ing., Dipl.-Ing. Martin Drahanský, Ph.D. for supervising, and providing his insights over the course of this work. Last but not least, I would like to say thank you to my mother, my girlfriend, and all my other friends and family for encouraging and supporting me during these stressful months.

Contents

Chapter 1

Introduction

"A database is a collection of information that is organized so that it can be easily accessed, managed and updated"[1]. In terms of finances and time, managing and updating such a database can be costly. The storage space required for this database may be expensive. Also, uploading and downloading high-quality images to and from this storage space is time-consuming. To deal with this problem a compression can be used and make the images smaller. The compressed image can be uploaded to the storage space while requiring less time for transfer and occupying less space. When the image is downloaded it can be recreated to match the original image. Using a compression means that some information may be lost, and the image will not be the same when recreated. Also, the compression and decompression take some time which may no be pleasant if many images are compressed/decompressed.

This work talks about different compression algorithms that can be used when dealing with the database of various fingerprint images. The diverse quality of fingerprint images may require different compressions to obtain satisfactory results. These results are produced by comparing the original and the recreated fingerprint image with a fingerprint matching algorithm. Furthermore, for each compression, various compression parameters can be used. Using the different values of compression parameters may affect the compression time, compressed file size, and also the comparison score.

An image can be enhanced through the application of various filters. If the image is enhanced before the compression, the process is called the pre-processing. The pre-processing can also alter the compression time, compressed file size, or comparison result. For the fingerprint images, well-known enhancement techniques include image normalization, image binarization, noise reduction, and thinning.

1.1 Goal of the thesis

The goal of this thesis is to test how various filters, compression algorithms, and compression parameters affect different kinds of fingerprint images. Specifically, an application will be proposed, which will offer the user a chance to load either a fingerprint image or a whole directory of fingerprint images. Then the user will have a chance to apply some image preprocessing technique, then compress the image with one or more compression algorithms provided. Then each compressed image can be decompressed, and compared to the original fingerprint image using some fingerprint matching algorithm. For each compressed image,

[1] https://searchsqlserver.techtarget.com/definition/database

file size and compression time are measured which can be used to calculate the performance of the compression method and the quality of the reconstructed image. The fingerprint matching algorithm will provide the equivalence between the original and the reconstructed image.

1.2 Document structure

There are 5 chapters in this thesis.

Chapter 2 provides a summary of the current state. It presents fundamental information about fingerprints such as their composition, pattern, and a process of how a fingerprint is taken. Furthermore, the ideas of how to enhance a fingerprint image through various filters are presented in 2.1.2. These filters aim to reduce the compression time, reduce the compressed file size, or improve the matching score produced by the fingerprint recognition algorithm. These fingerprint recognition algorithms are described in 2.1.3. Additionally, the compression process as well as the JPEG, the PNG, and the WSQ compressions are outlined in 2.2. Finally, the measurement of the compression efficiency by PSNR and MSE is explained in 2.2.5.

Chapter 3 talks about the projected functionality of the application that will be implemented. The functionality described in 3.1 will let the user to load a fingerprint image or a whole directory, apply a filter or a collection of filters to the loaded image(s), to compress the loaded image(s), or the compare the compressed image with the original one. Section 3.2 describes the implementation specifications, and finally in section 3.7, issues encountered during the implementation are specified.

Chapter 4 outlines how the application was tested and provides the results for different kinds of fingerprint images. Each type of fingerprint image was compressed with a miscellaneous compression algorithm with various compression parameters. Before the compression, a filter or a collection of filters was used to see if better matching results can be obtained.

Chapter 5 talks about whether the aims of this thesis were accomplished and discusses future work.

Chapter 2

Summary of the current state

2.1 Biometrics

Biometrics is "the measurement and analysis of unique physical or behavioral characteristics (such as fingerprint or voice patterns) especially as a means of verifying personal identity"[1]. Fingerprint recognition is the most common and one of the most successful biometric techniques. Fingerprint is "an ink impression of the lines upon the fingertip taken for the purpose of identification"[2]. From a more complex point of view, fingerprint contains the substances from the epidermis, the secretory glands, and intrinsic components (such as grease or food contaminants) [5]. It is believed that there are no two identical fingerprints [25]. Even twins have different fingerprints [15]. Therefore, it provides a great platform for people identification. Fingerprint recognition has been used in forensic science for a long time. The first mention of fingerprints usage comes from 1858 when Sir William Hershel used fingerprints of workers in the Indian Civil Service to document and verify their identity when collecting their wages [6]. Since then, fingerprints have been used in criminal investigations to identify the perpetrators. Many features can be told from the fingerprints, such as donors age, gender, and race [5]. More recently, they have found a new usage in smartphones, when the phone is allowed to be unlocked by its owner's finger.

2.1.1 Fingerprint

Another definition of the fingerprint, in contrast to the Merriam-Webster, is: "A fingerprint is the pattern of ridges and valleys on the surface of a fingertip" [11]. The uniqueness of each fingerprint is characterized by the relationships of the ridge characteristics. Most fingerprint matching algorithms which compare these local ridge characteristics, work with only two types of ridge characteristics. These characteristics, also called *minutiae*, consist of ridge ending and ridge bifurcation [11]. However, in forensic science, there are many more ridge characteristics used which are derived from the basic types [6]. Other minutiae include double bifurcation, triple bifurcation, crossover, delta, enclosure, dot, or an island. Some of the most common minutiae patterns can be seen in figure 2.1.

"A ridge ending is defined as the point where a ridge ends abruptly. A ridge bifurcation is defined as the point where a ridge forks or diverges into branch ridges." [11]

Each fingerprint image is unique and depending on the fingerprint and image quality, a fingerprint image of good quality can contain 60 to 80 minutiae [24]. A fingerprint quality

[1] https://www.merriam-webster.com/dictionary/biometrics
[2] https://www.merriam-webster.com/dictionary/fingerprint

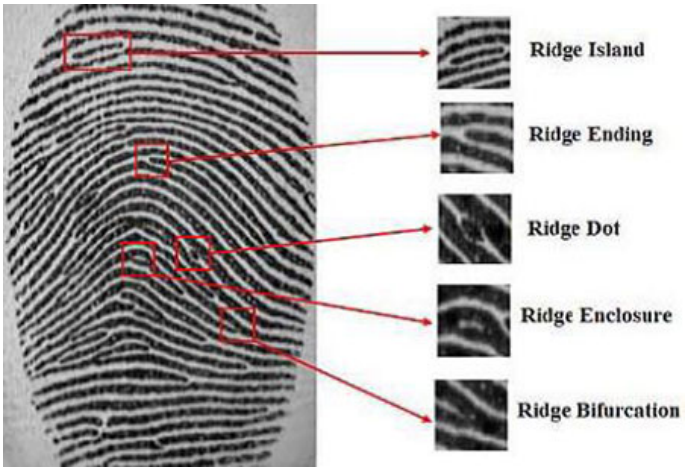


Figure 2.1: Minutiae patterns. Image taken from https://www.bayometric.com/minutiae-based-extraction-fingerprint-recognition/

is subject to numerous factors. Each fingerprint goes through two stages, the deposition and the aging [5]. The deposition stage takes place when the fingerprint is taken. During this time the donor's age, health, medication, race, gender, diet, and deposition conditions affect the resulting fingerprint. Depositions conditions include contact time, pressure, angle, and the substrate. The substrate is used to enhance the composition of the resulting fingerprint. Depending on the donor's characteristics various substrates and deposition methods can be used [5]. The second stage of a fingerprint is the aging stage which begins when the fingerprint is successfully deposited. A long aging stage can harm the composition of the fingerprint. The right substrate choice during the deposition stage can slow fingerprint degradation over time. However, other conditions also affect the fingerprint quality. The most influential conditions on the fingerprint quality include temperature, humidity, and light levels [5]. Depending on the time elapsed in the aging stage, some enhancement techniques can be used to alter the quality of the fingerprint. These enhancement techniques include chemical or physicochemical methods, for example the effect of aluminum powder or indanedione can have a significant effect on the fingerprint composition [5]. However, the impact of these methods depends on the age of the fingerprint. It has been suggested that the humidity has the largest influence on the quality of the enhancement. The higher the humidity the lower the quality of the ridge detail. As time passes, the water evaporates from the fingerprint and the chemical enhancements to the fingerprint have almost no effect. The evaporation of water can lead to large changes in thickness [5].

To prevent identification a fingerprint image can be forged. These forged fingerprint images can be produced by fake fingerprints or altered fingerprints [27]. The fake fingerprints are fingerprints fabricated from materials such as silicone [8]. Their main aim is to counterfeit the real fingerprint and fool the fingerprint matching device [8]. On the other hand, the altered fingerprints are real fingers. The altered fingerprint may be ill or otherwise obfuscated fingerprint. The ill fingerprints are fingerprints that have been produced by a finger that bears scars, scratches, burns, illnesses, or other skin damage[10]. As mentioned in [10], any damage done below the epidermis can cause the alteration in the fingerprint pattern permanently. The obfuscated fingerprints are fingerprints that have been intentionally altered to prevent identification [27]. It has been proven that some people [10] may intentionally try to change their fingerprint patterns to prevent identification. Fingerprint

images can also be artificially generated to imitate the real fingerprints [18]. The goal of generating such fingerprints is to "avoid collecting databases of real fingerprints"[18] which can be time-consuming [18].

2.1.2 Fingerprint enhancement

The fingerprint matching algorithms depend on the comparison of the ridge characteristics [11]. Therefore, the algorithms heavily depend on the quality of the fingerprint image. In an ideal fingerprint image, the ridge characteristics are well-defined, meaning "ridges and valleys alternate and flow in a locally constant direction and minutiae are anomalies of ridges, i.e., ridge endings and ridge bifurcations" [11]. However, in the real world, these minutiae are not always very well-defined and therefore they cannot be easily detected. The easier to detect these local ridge characteristics the more precise the matching algorithm can be.

To ensure that the minutiae are easily detected various fingerprint enhancement algorithms are used. Fingerprint image which has not been enhanced can produce problems, such as false minutiae, ignored minutiae or minutiae localization during fingerprint matching [11]. Therefore, it can be said that the main aim of the enhancement algorithms is to remove the false minutiae, clarify and improve the ridge structures which would be ignored or wrongly localized during the fingerprint matching process. Simply said, the fingerprint enhancement improves the quality of the fingerprint image [26].

The fingerprint enhancement algorithm can be run on either binary or gray-scale fingerprint images [11]. Each image type has its advantages and disadvantages. For example most of the fingerprint matching algorithms work better with binary fingerprint images. However, to convert an image from grayscale to binary can be time-consuming, even more, if the whole database has to be converted. Also, during the binarization, a lot of information may be lost and the binarization process may prove to be inefficient for low-quality images [25].

Nevertheless, the fingerprint enhancement does not only improve the fingerprint matching process but also fingerprint image compression can be improved. For example, when the false minutiae are removed before the compression, the compression time can be reduced and the size of the compressed image can also be reduced.

Normalization

The aim of the process of normalization is to diminish the variations in gray-level values along the ridges and valleys. Normalization removes the effect of sensor noise. Thus, normalization is used to change the intensity values of pixels [1]. "Normalization does not change the clarity of the ridge and valley structures" [11]. After the normalization, the image's colors are evenly spread throughout the image and the image is easier to compare [24]. Equation 2.1 [1] defines the normalized value for pixel located at (i, j) coordinate of the image:

$$N(i,j) = \begin{cases} M_0 + \sqrt{\frac{(V_0(I(i,j))-M_i)^2}{V_i}} & if I(i,j) > M \\ M_0 - \sqrt{\frac{(V_0(I(i,j))-M_i)^2}{V_i}} & otherwise \end{cases} \tag{2.1}$$

where $N(i,j)$ is the normalized value at (i,j) coordinate, $I(i,j)$ is gray level value at (i,j), M_0 is the desired mean, V_0 is the desired variance and $M_0 = V_0 = 100$ [1]. The

equations 2.2 [1] and 2.3 [1] define the variables M_i (mean) and V_i (variance) respectively.

$$M(I) = \frac{1}{L \times N} \sum_{i=0}^{L-1} \sum_{j=0}^{N-1} I(i,j) \tag{2.2}$$

$$V(I) = \frac{1}{L \times N} \sum_{i=0}^{L-1} \sum_{j=0}^{N-1} (I(i,j) - M(I))^2 \tag{2.3}$$

where $L \times N$ is an image dimension in pixels and $I(i,j)$ is the pixel intensity.

Binarization

The binarization aims to convert the fingerprint image to binary. Binary images have a pixel depth of one bit and are therefore black (0) or white (1). In a grayscale image each pixel is in the range from 0 (black) to 255 (white). When converting such an image to binary a threshold is chosen and each pixel that has its value less than this threshold is represented as 0 otherwise it is represented as 1. This threshold can either be a global or local/adaptive threshold. Global threshold means that a value is chosen before the binarization process starts and remains the same for the whole time. When the local threshold is chosen, for each pixel whose resulting value is determined, values of N-number of neighboring pixels are summed up and divided by N. The resulting value is used as the threshold. In grayscale images the pixel intensities vary. It has been suggested that it may be harder for some matching algorithms to distinguish between the ridges and the valleys and therefore the extraction of the key features used for verification proves difficult [24].

Thinning

This process aims to covert each ridge to be one-pixel wide [3]. Like the process of binarization, converting a fingerprint image to have thinned ridges is time-consuming, especially when the whole database is converted. However, fingerprint images that have thinned ridges do contain fewer redundancies and the performance of the compression algorithms may not be as time-consuming as they would normally be if the ridges were not thinned.

Noise reduction

As mention before, in practice, a fingerprint image is not ideal. There are elements of noise which corrupt the clarity of the ridge structures [3]. This noise produces the false minutiae and can hide the real one [1]. The noise is created by the variations in skin, namely scars, dirt, or bad contact with the fingerprint capturing device [3]. Thus, the noise reduction process enhances the definition of ridges against the valleys minimizing false minutiae detected. This noise reduction process can be done as a combination of normalization and Gabor filtering or Median filter, for example. Also, there are other suggested methods, such as Directional Fourier filtering [3].

Low-quality images come with noisy background. This background needs to be separated from the foreground during this process. By doing so, the future noisy background extraction from the foreground is avoided [2]. It is suggested that the fastest methods for this process are based on Gabor filters [2]. "Gabor filters are orientation-sensitive filters used for edge and texture analysis" [14]. They are mainly used for edge detection because the structure of the filter can easily detect edges in the images of various shapes and

sizes [14]. Simply put, these filters have very good localization properties that fit various images. By applying a Gabor filter to the fingerprint image, the foreground which contain ridges of different directions can be extracted from the background.

A more simple method of noise reduction is Median filtering. It does not detect the edges of an image and then extracts it from the noisy area, but it replaces pixel's value with the median of nearby pixels in a window [24]. Median filter method is simple, effective and it keeps the edges without blurring. However, in practice that is not always the case as "in the presence of the noise it does blur edges in images slightly" [24]. It is necessary to perform noise reduction, such as median filtering, before doing any further higher-level processing steps [24].

2.1.3 Fingerprint recognition algorithms

The main goal of the fingerprint recognition algorithm is to determine whether two fingerprint patterns have been produced by the same person [26].

There are various techniques used in fingerprint matching. According to [3] there are four fingerprint matching techniques:

- minutiae-based
- pattern matching
- correlation-based
- image-based

However, according to [20] there are three fingerprint matching techniques:

- correlation-based
- minutiae-based
- non-minutiae base

It can be seen that both of these classifications mention the minutiae-based and the correlation-based techniques. Still the classifications differ on the remaining items. According to [20], the non-minutiae based techniques "search for additional fingerprint distinguishing features, beyond minutiae" [20].

Minutia-based technique

The most commonly used technique today for scanners is the minutiae-based technique [3]. In this technique, a fingerprint is understood to be made of small local features such as ridge ending and bifurcation called minutiae. A template is formed by extracting the minutiae form the fingerprint [20]. "Matching essentially consists of finding the alignment between the template and the input minutiae sets" [20]. Therefore, the whole fingerprint matching problem is reduced to the point that two fingerprint images are considered equal if they have the same minutiae [3]. This technique requires high image quality for reliable minutiae extractions [25].

Correlation-based technique

The correlation is a measurement of similarity. The correlation-based technique takes two images and counts cross-correlation between them. It is a standard image similarity measurement technique. It takes a pixel intensity of the original image and the testing image, then it calculates their difference, then squares the difference [20]. Nevertheless, the original and testing fingerprints are not always ideally aligned, and so more complex algorithms have to be introduced to calculate the cross-correlation [20]. However, in fingerprint image recognition it can be quite computationally expensive [3], especially when trying to find a matching fingerprint image in a big database. Epidermis conditions can cause different fingerprint image effects, hence more sophisticated correlation algorithms are needed to increase technique accuracy [3].

Pattern-matching technique

Pattern matching technique is more content-based because unlike minutiae-based technique, it is based on a series of ridges [3]. These ridges form fingerprint patterns such as arch, loop, or whirl [13]. This technique compares these patterns in the original and the compared image [13]. Ridge can be affected by numerous effects such as finger placement on a scanning sensor [3] when the fingerprint is taken and therefore a positive match does not have to be found. This is the major drawback of the pattern matching technique. However, at the time of fingerprint deposition, minutiae points may be affected by wear and tear and minutiae-based technique would be prone to wrong results [3]. This is where pattern matching has an advantage over the minutiae-based technique because it is more efficient.

Image-based technique

Image-based technique is a newly emerging technique which can solve intractable problems [3]. In this technique, minutiae are not extracted from the fingerprint but whole fingerprint images are compared against each other. Image pre-processing is not required and a gray-scale fingerprint image may be used for matching [25]. Since pre-processing is not required, this technique has higher computational efficiency and can be used on images with low quality where minutiae or pattern-based technique would fail due to unreliable minutiae extraction [25]. Like pattern matching, this technique is very dependable on the orientation of the fingerprint in the image and therefore can produce misleading results.

Non-minutiae based technique

For low-quality fingerprint images, the extraction of minutiae may be complicated [20]. Therefore, the non-minutiae based technique must be introduced. This technique is based on the image texture, either local or global [20]. "Image texture is defined by the spatial repetition of basic elements, and is characterized by properties such as scale, orientation, frequency, symmetry, isotropy, and so on" [20]. This technique also depends on the right alignment of the original and the compared fingerprint images [20]. To ensure that images are correctly aligned, some minutiae may be used as anchor points [20].

2.2 Compression algorithms

The main goal of a compression algorithm is to reduce the information required to represent the image [12]. Decompression is the opposite process to the compression. The file created by the compression is called the *compressed file* and the file created after the decompression is called the *decompressed file*. The image created by the compression is called the *compressed image* and the image created by the decompression is called the *decompressed image*. The image compression comprises of two parts. The first part is called a compressor or an encoder which produces a compressed image from the original image. The second part is called a decompressor or a decoder and it creates a reconstructed image from the compressed image. Section 2.2.1 contains more information about the encoder and section 2.2.2 contains more information about the decoder. The basic steps of the compression are shown in figure 2.2 [21].

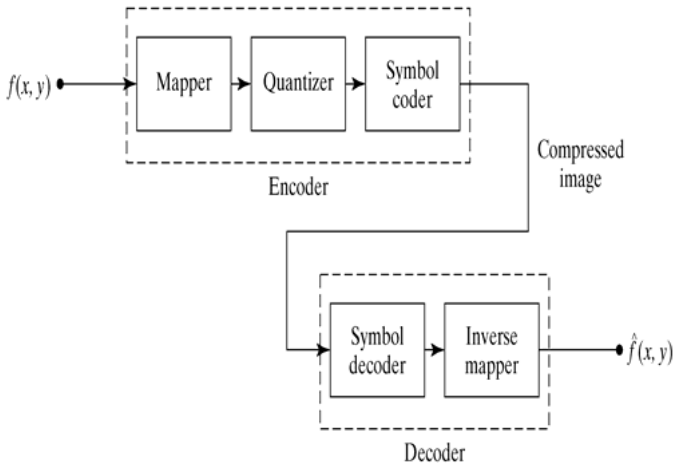


Figure 2.2: Figure show the basic compression and decompression steps. Image taken from [21]

There are two main compression methods: *lossy* and *lossless* [23]. When using the lossy compression, some of the information representing the original image is lost. Thus, allowing the compressed image to become smaller. On the other hand, the reconstructed image does not exactly match the original image. Meanwhile, the reconstructed image of the lossless compression is the replica of the original image but the size of the compressed image is not as small.

Regardless of the compression type, there are "three basic steps: transformation, quantization, and encoding" [23]. The most common way of compressing an image is to reduce the correlation or redundancy among the neighboring pixels. This redundancy is also called *spatial correlation*. Before the compression, some pre-processing techniques can be used to prepare an image for the compression, thus making the compression even more effective. It has been suggested, that reducing the correlation before the compression can improve the compression effect [12]. After the compressed image has been decompressed some post-processing techniques can be used to filter out artifacts gained during the compression [12]. JPEG compression is the most commonly known to leave the artifacts.

2.2.1 Encoder

The compression process starts with data reduction. Then "the second step is the mapping process, which maps the original image data into the mathematical space where it is easier to compress the data" [12]. Following the mapping process is the quantization, which puts the data from the mapping stage in the discrete form [12]. Lastly, the data are coded [12]. Depending on the compression algorithm, the compressor can consist of all these stages or only some of them [12], for example the lossless compressions may not require the quantization process.

2.2.2 Decoder

To decompress the image, reversible transformations are needed to be applied. The quantization process is not reversible. Therefore, no reversible process exists and some information is lost.

Decompression starts with reversing the coding [12]. It does so by mapping the codes to the quantized values [12]. Following the decoding is the "inverse mapping to reverse the original mapping process" [12]. After this, the reconstructed image is obtained.

2.2.3 Lossy compression

The most common compression algorithms used for images desiring lossy compression are based on the Discrete cosine transform (DCT) [12]. "The DCT works by separating images into parts of different frequencies" [4]. The less important frequencies are then discarded during the quantization. During the reconstruction process the image is only reconstructed from the non-discarded frequencies, hence the *distortion* occurs [4].

Compressed images produced by the lossy compression can be much smaller in size than the ones produced by the lossless compression.

JPEG compression

JPEG compression is a compression scheme specified by the Joint Photographic Experts Group (JPEG) based on the Discrete cosine transform (DCT). It has been reported that JPEG is not suitable enough for fingerprint images compression because of the minutiae degradation and the relics of blocking artifacts [16]. To increase the accuracy of the fingerprint matching algorithms, artifacts of the JPEG compression need to be filtered out [9].

WSQ compression

It has been suggested that desiring a high compression ratio while maintaining high image quality, wavelets should be used [23]. Although wavelet compression is lossy, the reconstructed image contains differences that are barely visible to the human eye [23].

WSQ compression can be based on a Discrete wavelet transform (DWT). DWT analyses the signal in the time-frequency domain. DWT is made of wavelets which correlate with the signal [6]. These wavelets are defined in the finite time interval [6]. An example wavelet can be seen in figure 2.3. In contrast to Discrete Fourier transform and Discrete cosine transform, DWT has better reduction capabilities and better computation time [23]. "DWT can deliver better image quality on higher compression ratios" [23].

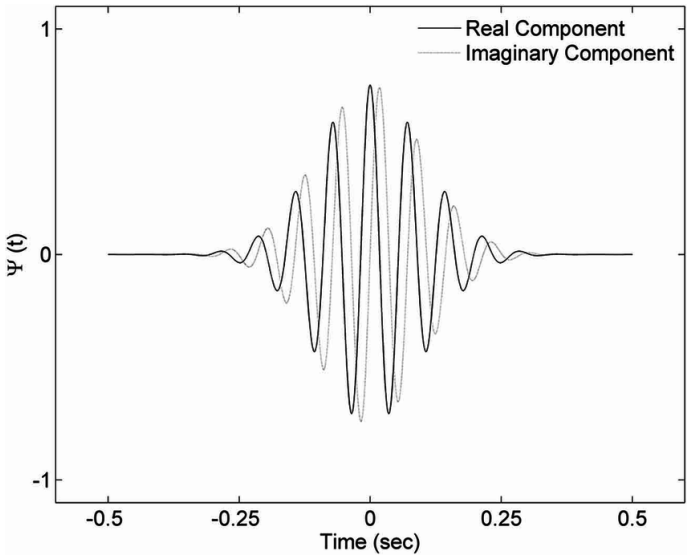


Figure 2.3: A wavelet example, Morlet wavelet. Image taken from https://www.researchgate.net/figure/Complex-Morlet-wavelet-with-central-frequency-f0-2-used-in-the-analysis-Presented_fig4_284246841

2.2.4 Lossless compression

Compressed images produced by the lossless compression can be perfectly reconstructed to their original image. Some images, for example, the ones containing the results from a medical examination, may need to be reconstructed perfectly otherwise they may lead to wrong diagnosis [12].

PNG compression

The PNG compression is done in two phases, filtering and compression [19]. The filtering is applied to make the image more compressible. For each pixel, the filter predicts the value based on the neighboring pixels and then subtracts from the current value [19]. The compression is based on the Deflate format, which is a combination of Huffman coding and LZ77 [19]. The LZ77 algorithm reduces the repetitive data [7]. "If the algorithm encounters a sequence of data that has been previously used in the file, it replaces it with a reference to the first sequence" [7]. Lastly, it reduces the character representation with the Huffman coding [7]. Thus, it can be said that if multiple repetitive sequences arise in the image then the PNG is a perfect choice.

2.2.5 Compression efficiency

There are many ways to measure the compression efficiency. One approach is to calculate the *compression ratio*, in equation 2.4 [12], which is calculated as a ratio of the size in bytes of the original file and the size in bytes of the compressed file.

$$compression \quad ratio = \frac{original \quad file \quad size}{compressed \quad file \quad size} \tag{2.4}$$

Another way to find out how one image matches the another is to measure the fidelity [16]. Fidelity expresses the difference between the original and the reconstructed image. This

difference is also called the *distortion*. If this distortion is small it means that the reconstructed image is very similar to the original one. To calculate this distortion, firstly one must calculate a simple form called *mean square error (MSE)*, defined in equation 2.5 [12] which can be used.

$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} (f(i,j) - g(i,j))^2 \tag{2.5}$$

where f is an original image, g is a reconstructed image, *f(i,j)* is the pixel value at (i,j) coordinate of the original image, *g(i,j)* is the pixel value at (i,j) coordinate of the recostructed image, and both images have $M \times N$ pixels. "This is a useful measure as it gives an average loss in the lossy compression of the original image" [12]. Another distortion measure between the original and the reconstructed image is the *peak signal to noise ratio (PSNR)*. PSNR is used to describe the similarity (in decibels). Equation 2.6 [16] defines PSNR:

$$PSNR = 20 \times log_{10}(\frac{255}{\sqrt{MSE}}) \tag{2.6}$$

where *MSE* is the mean square error calculated by equation 2.5. Thus, the smaller the errors the smaller the *MSE* the bigger the similarity. Two images are considered to be identical when the PSNR is higher than 40 dB [23].

Chapter 3

Proposed Application

This chapter provides the layout of the projected functionality of the implemented application as well as some implementation specifications.

3.1 Projected functionality

The final application will serve as a tool for manipulation with fingerprint images. It will allow the user to compress and decompress the fingerprint image or a whole directory of fingerprint images. After the fingerprint image(s) has been decompressed, the user can compare them with the original image(s).

This application will support both compression types, lossy and lossless. For lossy compression, the application will provide the compression based on the Discrete cosine transform (DCT) as well as the Discrete wavelet transform (DWT). For DCT, JPEG compression will be supported and for DWT the WSQ compression will be supported. For lossless compression, the PNG compression will be supported.

As outlined in [16], the whole compression/decompression can be made in the so-called *compression cycles*. The compression cycle is a process in which multiple compression methods are used to obtain the final compressed image. The compression method used during the compression cycle can remain the same or different compression methods can be used. The implemented application will also support the use of these compression cycles.

As mentioned in 2.2, the compression result can be altered with the use of some image enhancement techniques. Not only the compression results can be altered, but also the fingerprint matching results can be altered with the use of the image enhancement techniques. Thus, the application will provide the user the choice to normalize the image, to remove noise from the image(to discard the false minutiae and reveal the real one), to binarize the image(to decrease the compression time and improve the fingerprint matching result), to resize the image and to change the depth of the image(as normalization can only be done on gray-scale images).

For each compressed fingerprint image, the compression time and the size of the compressed file will be measured. For the comparison between the original and the reconstructed image, the application will support the different fingerprint matching algorithms as well as calculating the PSNR. The PSNR will be measured to calculate the distortion. These measurements will be taken to compare the efficiency of using different compression methods with and without the pre-processing techniques.

It has been suggested by the supervisor that the application should support loading of an XML which would contain information about source directory, destination directory, filters to be used, compressions with their parameters to be used, and matches to be used. The application would then execute the loaded XML. Such XML would also be used for testing purposes.

3.2 Implementation specifications

The application is implemented in the Java programming language with the GUI implemented in the JavaFX library with Maven being used for the application's building process.

The minimum Java version required to launch the application is 11. There is a simple reason for the version being set so high. The GUI requires the JavaFX library to be a part of the Java Development Kit (JDK). The library was supposed to be a part of the JDK until version 1.8. Unfortunately, that is not always the case, as experienced during the implementation. The library seemed to be included in the JDK but some files were missing, thus making it unable to launch the GUI. After the missing files were downloaded and copied to the required directory, the GUI launched successfully. There is a simple solution to prevent the problems with the JavaFX library. The JavaFX library will be downloaded[1] during the build process. One of the requirements for this workaround to work is having the Java version 11 or posterior.

3.2.1 GUI implementation

GUI works with JavaFX's scenes. Each scene is divided into three parts. At the top of each scene, a brief text description of the current scene is provided. The left part of the scene comprises of clickable buttons which allow the user to interact with the scene. These are explained below but they include actions such as loading an image, proceeding to the next scene, or going back to the previous scene. Also, the pre-processing scene contains a selection box, which allows users to select the filters of their desire. Finally, the center of the scene, composes of an area that shows loaded images. Figure 3.1 shows the pre-processing scene, and also it shows how the scene is divided. Furthermore, there is a pop-up window which is shown when an error is encountered or the selection of the compression parameters is required.

When the application is run with the GUI, an initial scene is set. This initial scene provides the user with an ability to load image(s) or load an XML.

When the user chooses to load image(s), options to start the compression process, or to start the matching process are provided. Selecting the start the compression process button navigates the user to a pre-processing scene where filters can be applied to an image. This scene offers buttons for loading an image, loading a directory of images, starting the compression, or going back to the previous scene. The figure 3.1 shows the pre-processing scene.

Starting the compression changes the current scene to the compression scene. The compression scene offers the user a choice to choose one of the supported compressions. If no images are loaded then a pop-up shows to inform the user that the compression cannot proceed further, otherwise a pop-up shows to display the selection of the available compression parameters for the selected compression. Section 3.2.3 talks about the possible

[1] https://openjfx.io/openjfx-docs/#introduction

compression parameters for each compression. Figure 3.2 shows the compression scene with the pop-up for selecting the compression parameters for JPEG compression.

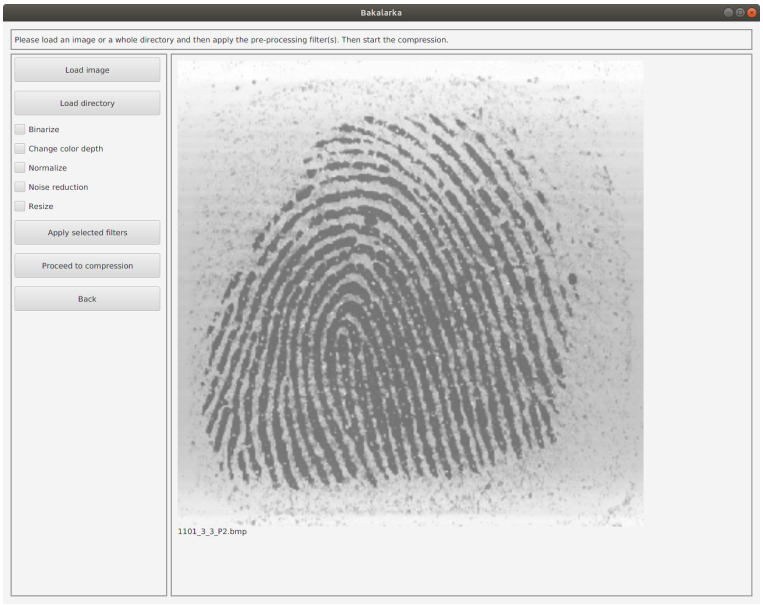


Figure 3.1: Pre-processing scene

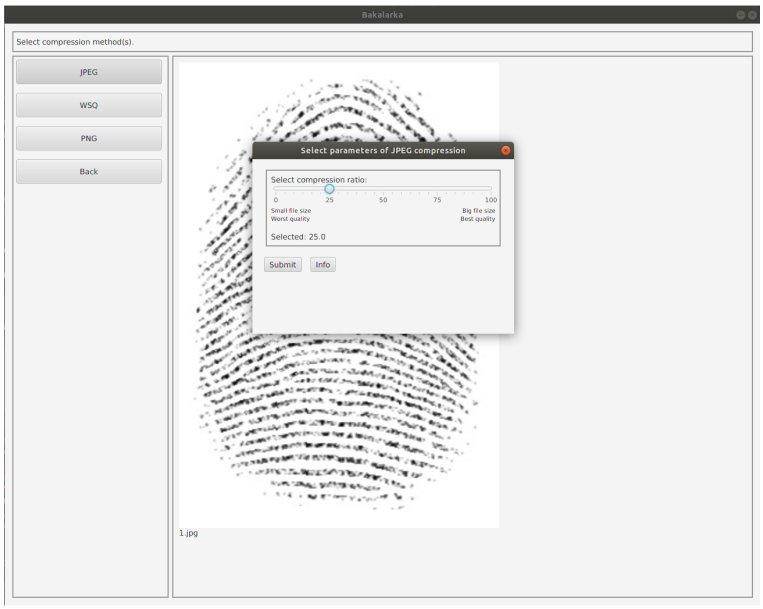


Figure 3.2: Compression scene with the pop-up for selecting the compression parameters

The matching scene is presenting options for users to load an original image, to load a comparing image, to start minutiae-based matching, to start cross-correlation matching, or to start PSNR matching. Both, the original and the comparing images have to be loaded to start the matching.

When the user chooses to load an XML, options to load an XML, or to execute the XML are provided. Section 3.2.5 provides more information about the XML which can be loaded.

At the end of the filter application, image compression, or image matching a pop-up is shown informing about the success or failure of the performed action.

With the initial scene, the *ImageManager* class is instantiated. This class is responsible for holding the image(s) that were loaded and are shown in the GUI. It holds a *List* of *MyImage* objects in *myImages* instance variable. By calling ImageManager's *display()* method, all MyImage objects in the *myImages* variable are displayed in the GUI. By calling *addImage(MyImage myImage)* method, MyImage object can be added to the list and by calling *clearStackPane()* all images that are currently being displayed in the GUI are cleared. MyImage class represents an image and holds an image as JavaFx *Image* object (used in GUI), *BufferedImage* object (used for image manipulation, e.g. filtering), and String containing the name of the image.

3.2.2 Pre-processing implementation

For the previously supposed normalization, binarization, noise reduction, resize and change depth filters, few modifications had to be made.

First of all, the numerators in the normalization formula 2.1 cited in this work based on [1] do not match the numerators in the normalization formula:

$$N(i,j) = \begin{cases} M_0 + \sqrt{\frac{(V_0)(I(i,j)-M_i)^2}{V_i}} & if I(i,j) > M \\ M_0 - \sqrt{\frac{(V_0)(I(i,j)-M_i)^2}{V_i}} & otherwise \end{cases} \tag{3.1}$$

which was defined in [17] and cited in [1]. Both normalization formulas were tried producing different results. Figure 3.3 shows the original fingerprint image 3.3a, the resulting image 3.3b after the 2.1 normalization formula was used, and the resulting image 3.3c after the 3.1 normalization formula was used.

Fingerprint images which contain the Alpha channel cannot be passed to JPEG compression, or else an exception is thrown. So, an option for the user to remove the alpha channel by changing the color depth of an image was added. The user can adjust an image so that each pixel has the bit depth of either one, eight, twenty-four or thirty-two bits. If the user does not remove the alpha channel then it will be removed.

All of the implemented filters extend the abstract *MyFilter* class. This abstract class contains only the constructor and three methods, of which all are abstract. Then each filter class, either *Binarization*, *ColorDepth*, *NoiseReduction*, *Normalization*, or *Resize* implements these abstract methods. The most important method is the *filter(BufferedImage originalImage)* method. This method applies the specific filter to the original image. For example, by calling the Binarization's filter method, the original image will be binarized, while calling the NoiseReduction's filter method will remove the noise in the original image.

The binarization and the noise reduction (by using a Gabor filter) filters, were implemented with the use of the *OpenCV* library. For binarization, both the global and the adaptive threshold were implemented. But the global threshold produced unsatisfactory results, thus only the adaptive threshold was used later on. OpenCV's *adaptiveThreshold(src, dst, maxValue, adaptiveMethod, thresholdType, blockSize, C)* function being used for the adaptive threshold, and "the threshold value is a Gaussian-weighted sum of the

(a) The original fingerprint image. Image taken from the database for testing purposes containing the real fingerprint images

(b) Image normalized with 2.1

(c) Image normalized with 3.1

Figure 3.3: A figure shows the original fingerprint image, and resulting images after each of the normalization formulas were used

neighboring pixel values"[2]. To use the binarization, firstly an image must be converted to gray-scale if it is not already. To use the OpenCV's functions for the binarization, and the noise removal, an image must be represented by the OpenCV's *Mat* class. Thus, Java's BufferedImage must be converted to the OpenCV's Mat instance. After the filters were applied, then the Mat object must be converted back to the BufferedImage instance.

For Gabor filtering, firstly a Gabor kernel needs to be created with OpenCV's *getGaborKernel(ksize, sigma, theta, lambda, gamma, psi, ktype)* function. After creating such a kernel, it needs to be applied to the image with OpenCV's *filter2D(src, dst, ddepth, kernel)* function. The *theta* parameter "is the orientation of the normal to the parallel stripes of the Gabor function" [22] so it may be necessary to create more than one kernel for different orientations of the normal. Then, each kernel is applied to the source image resulting in several partial results which are added together to form a resulting image by OpenCV's *addWeighted(src1, alpha, src2, betta, gamma, dst)* function. In this case, the theta is set to 0.0, 45.0, 90.0, and 135.0 degrees respectively. Figure 3.4 shows fingerprint image 3.4f obtained after all partial images 3.4b, 3.4c, 3.4d, 3.4e were added together.

However, not every fingerprint image gets to be filtered as good as the one in figure 3.4f. Some fingerprint images have good partial results, but after they are added together, they produce unsatisfactory results, as seen in 3.5f. The reason behind this behavior remains unknown.

To resize an image a new instance of the BufferedImage class, with the percentage of the width and height of the original image is created. For example, if the original image has dimensions of 500 pixels wide and 320 pixels high, and the image should be resized to 50% of the original size, then the new image will be 250 pixels wide and 160 pixels high. After the new image class is created, then the *Graphics2D* is created, by calling the BufferedImage's *createGraphics()* method, to which an image can be drawn with the Graphics2D's *drawImage()* method.

[2] https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html

(a) The original finterprint image. Image taken from the database for testing purposes containing the fake fingerprint images

(b) Image produced with kernel's theta set to 0°

(c) Image produced with kernel's theta set to 45°

(d) Image produced with kernel's theta set to 90°

(e) Image produced with kernel's theta set to 135°

(f) Image produced by adding all partial images together

Figure 3.4: A figure shows the original fingerprint image, partial images produced with different theta parameter set for gabor kernel, and resulting image after each of the partial images were added together

The filter that changes the depth of an image works similarly. Firstly, a new instance of the BufferedImage is created with the desired image type. Then, the Graphics2D is created, to which the image is drawn.

3.2.3 Compression implementation

The compression is done by *ImageIO* class which is part of the Java Image I/O framework. It contains methods for locating *ImageWriter* class which fulfills encoding or *ImageReader* class which fulfills decoding. ImageWriter and ImageReader are abstract superclasses, and for example for PNG compression a *PngImageWriter* subclass is needed. ImageIO can locate subclasses used for basic compressions, such as JPEG, PNG, BMP, or TIFF. Unfortunately, ImageWriter's and ImageReader's subclasses needed for WSQ encoding and decoding are not part of the Java Image I/O framework. For this work, an open-source free

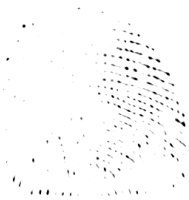
(a) The original finterprint image. Image taken from the database for testing purposes containing the real fingerprint images

(b) Image produced with kernel's theta set to 0°

(c) Image produced with kernel's theta set to 45°

(d) Image produced with kernel's theta set to 90°

(e) Image produced with kernel's theta set to 135°



(f) Image produced by adding all partial images together

Figure 3.5: A figure shows the original fingerprint image, partial images produced with different theta parameter set for gabor kernel, and the unsatisfactory resulting image after each of the partial images were added together

implementation of WSQ[3] encoder/decoder which is a part of the Machine Readable Travel Document standards specified by the International Civil Aviation Organization was used. The used WSQ implementation is licensed under the GNU Lesser General Public License.

All the implemented compressions extend the abstract *MyCompressor* class. The most important methods in this abstract class are the *compress(BufferedImage uncompressedImage, String imageFullPath, String destinationDir)* and the *decompress(File file)* methods which are used for compression, and decompression respectively. The compress method is abstract and therefore all of the compressor subclasses (*JpegCompressor*, *PngCompressor*, and *WsqCompressor*) implement it. On the other hand, the decompress method is not abstract but static. Each of the compressors have their own *decompress(File file)* method with *protected* access. So, by calling MyCompressor's decompress method, and depending on the file's extension, passed as a parameter, the appropriate subclass decompress method is called.

Also, the compressors are responsible for keeping track of the compression time. This is done by calling *System*'s *nanoTime()* method before and after the ImageWriter's write

[3] https://github.com/E3V3A/JMRTD/tree/master/wsq_imageio

method is invoked. Then the starting time is subtracted from the ending time and the result is converted into milliseconds, which results in the compression time.

For each compression process, certain compression parameters can be set and passed to the encoder. For JPEG compression, the compression quality can be set to a value between 0 and 100. A compression quality of 0 indicates that high compression is desired and a compression quality of 100 indicates that high image quality is desired. In lossy compressions, like JPEG, this ratio[4] should be a compromise between file size and image quality, for lossless compressions, like PNG, this number should be a compromise between file size and compression time. For PNG compression, the compression quality can be set to a value between 0 and 100 and progressive mode can be set to indicate whether the image should be encoded in a progressive mode. For WSQ compression, the encoding rate can be set. The encoding rate specifies the bitrate in bits per pixel for encoding, and can be set to a value between 0.1 and 1.0.

These compression parameters, represented by *JpegParams*, *PngParams*, and the *WsqParams* class, are subclasses of the abstract *MyParams* class.

3.2.4 Matching implementation

The final application supports the minutiae-based, and the cross-correlation fingerprint matching as well as the PSNR calculation. All matching classes are subclasses of the abstract *MyMatcher* class. The Adapter design pattern was used, as classes for the minutiae-based and the cross-correlation matching work with the 3rd-party projects.

The minutiae-based matching is done with the open-source SourceAFIS[5] algorithm for fingerprint recognition. It matches two fingerprints or searches a database for a match and the resulting score must be greater than 40 to consider fingerprints identical. For this project, only matching two fingerprints is supported. The SourceAFIS algorithm is located in the Maven Central repository, so only Maven dependency is needed to be added to download it. The SourceAFIS is under Apache Licence 2.0 and the original developer is Robert Važan[6]. This project is still being developed and so the Maven dependency may become obsolete and the methods in the *MinutiaeMatcher* class will have to be updated.

The cross-correlation matching is done with the open-source BiometricSDK project[7]. Only *CFingerprint* class was needed from this project as it also comes with GUI which was not needed. This class matches two fingerprints and the resulting score is returned as a percentage. The download site of the BiometricSDK project state that the project is under the Mozilla Public Licence version 1.0 however, the files have a header stating that they are under the GNU General Public Licence. This discrepancy has led to a decision that the CFingerprint class required for the implemented *CrossCorrelationMatcher* class will not be a part of the project to comply with the GNU GPL. The CFingerprint class can be downloaded and copied to the *BiometricSDK* directory however, the project cannot be conveyed with this class, or the class cannot be modified. Also, it should be stated that the original developer of this project is a user with a moleisking[8] username.

For PSNR matching, the *PsnrMatcher* class was implemented, which for matching, firstly calculates MSE 2.5 and then PSNR 2.6. As mentioned in [23] the resulting PSNR score needs to be higher than 40 for two images to be considered identical.

[4] https://docs.oracle.com/javase/7/docs/api/javax/imageio/ImageWriteParam.html#compressionQuality
[5] https://sourceafis.machinezoo.com/java
[6] https://robert.machinezoo.com/
[7] https://sourceforge.net/projects/biometricsdk/
[8] https://sourceforge.net/u/moleisking/profile/

3.2.5 XML implementation

The application does not need to be launched with GUI. If one already knows which fingerprint images to compress, the compression, its parameters, and the matcher to use it is possible to create an XML that will contain this information.

This XML contains root element *entry* which encloses a mandatory elements *sourceDir*, *destinationDir*, *filters*, *compressions*, *matchers*.

The sourceDir element contains an absolute path to the directory which contains the fingerprint images which will be processed. The destinationDir element contains an absolute path to the directory where the processed fingerprint images will be stored.

The filters element may contain additional *filter* elements indicating which filters to use. Each filter element contains mandatory *filterName* element and *filterParams* element as some filters also take parameters. The filterName element contains a name of a filter which will be used, namely *binarization* for image binarization, *normalization* for image normalization, *resize* for resizing an image, *noiseReduction* for reducing the noise in an image, *colorDepth* for changing the color depth. Additionally, when the resize, or the colorDepth filter is selected the filterParams element is needed. For the resize filter, the filterParams element contains *resizePercentage* element indicating the percentage to which the original image will be resized. For the colorDepth filter, the filterParams element contains *newDepth* element indicating a new depth of an image. If the filters element does not contain any additional filter element(s) then no filter is applied.

The compressions element contains supplementary *compression* elements specifying compressions that will be used. The compression element contains mandatory *compressionName* and *params* elements. The compressionName element implies the compression that will be used, particularly *jpeg*, or *jpg* for JPEG, *png* for PNG, *wsq* for the WSQ compression. The params element contains the *ratio* element for the JPEG and PNG compressions indicating the compression ratio. For the WSQ compression, the params element contains the *encodingRate* element denoting the encoding rate.

The matchers element may contain additional *matcher* elements indicating the matchers that will be used for matching the processed image with the original one. The matcher element contains the *matcherName* element denoting the matcher that will be used, specifically *crossCorrelation* for cross-correlation, *minutiaeBased* for minutiae, *psnr* for the PSNR matcher.

This XML is represented by the *XmlRepresentation* class, which holds the *execute()* method for executing XML content and the *serialize(String saveDir, String fileName)* method for serializing an XML content. Executing an XML content is described in 3.3. For generating a mass amount of XMLs for testing, class *XmlGenerator* was implemented. This class generates all the XMLs that were used for testing described in 4.1. An example XML can be seen in A.1.

3.3 XML execution

By executing XML content, loading fingerprint images from the source directory, applying the specified filter, applying the selected compression with its parameters, and matching via the selected matchers is understood.

The *XmlExecutor* class is responsible for executing XMLs. It can receive one XML or a whole directory of XMLs to execute. For XML it executes, it loads the fingerprint images in the source fingerprint image directory, then for each fingerprint image it applies

a filter or a collection of filters, next it compresses the fingerprint image with the specified compression, then it loads the compressed image, thus decompressing the image, and lastly, it compares this image to the original image with the defined matchers.

After the compression and the matching is finished, the compression time, which is measured by the compressor, and the matching score, which is measured by the matcher, is saved to an XML file representing results for the current fingerprint image.

The class *ScoreResultsExtended* is responsible for keeping track of the best and the worst matching scores, the class *TimeResultsExtended* is in charge of keeping the fastest and the slowest compression times, and the class *SizeResultsExtended* is accountable for keeping the smallest and the biggest file sizes for the currently loaded fingerprint images within the currently executed XML(s). These scores, times, and sizes are measured for each compression. So, for example, after each compression, the compression time is compared with the best and the worst compression time recorded, and if the compression time is better or worse than the currently best or the worst registered, then the compression time becomes the new best or the worst registered for the given compression and the XML path is added to the list of the best or the worst in the respective result category.

After all the XMLs are executed then the results are saved to a text file under a specified file name.

3.4 Results representation

The results from an XML execution are put into the text file. These results contain information about the best matching score, the worst matching score, the fastest compression time, the slowest compression time, the biggest file size, and the smallest file size for each compression type. In the result text file is also the name of an XML which produced that result. The following sample illustrates part of the results which may be received:

```
THE BEST JPEG MINUTIAE SCORES: 661.3097055229035
the best result xmls:
/home/marek/IdeaProjects/IBT/AdvImg2/src/test/resources/
generatedXmls/jpeg/jpeg100/generated_colorDepth_8.xml
--------------------------------
THE BIGGEST PNG FILE SIZE: 98807 B
the biggest file sizes xml:
/home/marek/IdeaProjects/IBT/AdvImg2/src/test/resources/
generatedXmls/png/png100/generated.xml
```

Besides the XML execution's results, also results for each processed fingerprint image are generated. These results are in the form of an XML with the following structure. The root element *result* encloses mandatory elements *fileNames*, *appliedFilters*, *compressionTime*, and *matcherScores*. The fileNames element contains the *currentFilename*, the *currentAbosolutePath*, the *originalFilename*, and the *originalFileAbsolutePath* elements. Each of these elements holds a name of a file or an absolute path to the file. The appliedFilter elements may enclose several *appliedFilter* elements that represent the filter that was applied. The compressionTime element holds a value in milliseconds representing the compression time. The matcherScores element consists of matcher scores that were obtained after matching. The *minutiaeBasedMatcherScore* holds the score for the minutiae-based matching and the *crossCorrelationMatcherScore* holds the score for the cross-correlation matching. An example XML can be seen in A.2.

3.5 Result parser

Section 3.7.1 talks about the performance issues encountered during the testing phase. One of the issues was insufficient heap space for source fingerprint image directory containing many fingerprint images. The temporary solution was to split the source fingerprint image directory so that it would contain fewer images. Executing the test cases for such directory results in obtaining only partial results. If the source fingerprint image directory was split into three directories then three partial results were obtained. These partial results have to be merged to obtain the real results. For this purpose, the *ResultParser* class was implemented.

This class loads all partial results and finds the real matching results, time results, and size results. The highest score value for a certain compression from all partial results is taken to obtain the real best matching score for a certain compression. For example, if partial results number one state that the best minutiae-based matching score obtained after the JPEG compression is 41 and the partial results number two state that the best score is 42 then the ResultParser puts 42 as the real best minutiae-based matching score obtained after the JPEG compression. The same analogy applies to obtaining the longest compression time and the biggest file size. On the other hand, the lowest score value for an individual compression is taken to acquire the real worst matching score for an individual compression. The same can be said for retrieving the shortest compression time, and the smallest file size.

Apart from result merging, the ResultParser also parses results to already formed LaTeXtables and plots and saves it in a separate file. For example, the best minutiae-based matching score obtained after the JPEG compression can be simplified from this:

```
THE BEST JPEG MINUTIAE SCORES: 661.3097055229035
the best result xmls:
/home/marek/IdeaProjects/IBT/AdvImg2/src/test/resources/
generatedXmls/jpeg/jpeg100/generated_colorDepth_8.xml
```

which presents the absolute path to the XML which produced this result to this:

```
% jpeg
\begin{table}[htbp!]
\centering
\begin{tabular}{|c|c|c|}
\hline
Ratio & Filters & Score \\
\hline\hline
100 & depth changed to 8b & 661.3097055229035 \\
100 & none & 661.3097055229035 \\
\hline
\end{tabular}
\caption{The ratios and filters for the best JPEG minutiae-based match}
\label{fal:bjpmm}
\end{table}
```

which states the ratio and the filter for this result. Moreover, if there were more ratios with the same filter that produced the same score they would be grouped.

3.6 Launching the application

As specified in 3.2 the only pre-requisites required for launching the application are JDK 11 or posterior, and Maven. Maven is responsible for adding the required dependencies. Also, the cross-correlation matching works a BiometricSDK project must be downloaded and the CFingerPrint class must be copied to the relevant directory. The project can be either launched from *IntelliJ* IDE or from the command line. The *README.md* file contains more information about this.

To build the application from the command line, Maven's clean, compile, package, install command must be executed:

```
mvn clean compile package
```

Application with GUI can be launched by executing the following command:

```
mvn javafx:run
```

Help can be displayed with the followng command:

```
java -jar IBT.jar -h
```

Generating test XMLs is done with the following command:

```
java -jar IBT.jar -g "fingerprints_source_dir" \
"fingerprints_destination_dir"
```

where *fingerprints_source_dir* is the absolute path to the directory which holds the fingerprint images for which the XMLs should be generated and *fingerprints_destination_dir* is the absolute path to the directory which will the fingerprint images once they are compressed.

Executing an XML is done with the following command:

```
java -jar IBT.jar -e "xml_path" "result_file_name"
```

where *xml_path* is the absolute path to an XML that will be executed and *result_file_name* is the file name that will contain the results.

Parsing and merging result files together is executed with the following command:

```
java -jar IBT.jar -p "xml_results_dir" "parsed_file_name"
```

where *xml_results_dir* is the absolute path to the directory which contains partial results that will be merged and parsed, and *parsed_file_name* is the file name that will contain the merged and parsed results.

To execute any of these commands, one must be in the directory that contains the *pom.xml* file. All of these are also mentioned in the *README.md* file.

3.7 Encountered issues

This section covers the issues that arose during test execution. The main issues concern performance while other issues may concern the usage of the open-source compression/decompression.

3.7.1 Performance issues

Executing the tests brought some serious performance issues. While some issues were fixed, some needed the workaround.

The most problematic issue was the performance during the test execution. For each fingerprint image, there were 225 test cases. Executing these test cases for a directory that contained 20 or more big size fingerprint images results in the heap memory to run out. For testing purposes, commenting out log outputs would decrease the execution time. To deal with the performance problems some temporary solutions were introduced. The first temporary solution was to split the source fingerprint directory so that it would not contain so many images. Another temporary solution to this issue may be increasing the heap size allocated to the application. However, these temporary solutions are not ideal as they do not solve the problem. The real solution that helped was to reuse the allocated BufferedImage instances and Lists during the XML execution. After this solution, all the test cases were completed successfully but still took a long time. This time, however, may only be a subjective case as it was only tested on one device.

Figure 3.6 shows the CPU usage during the test execution.

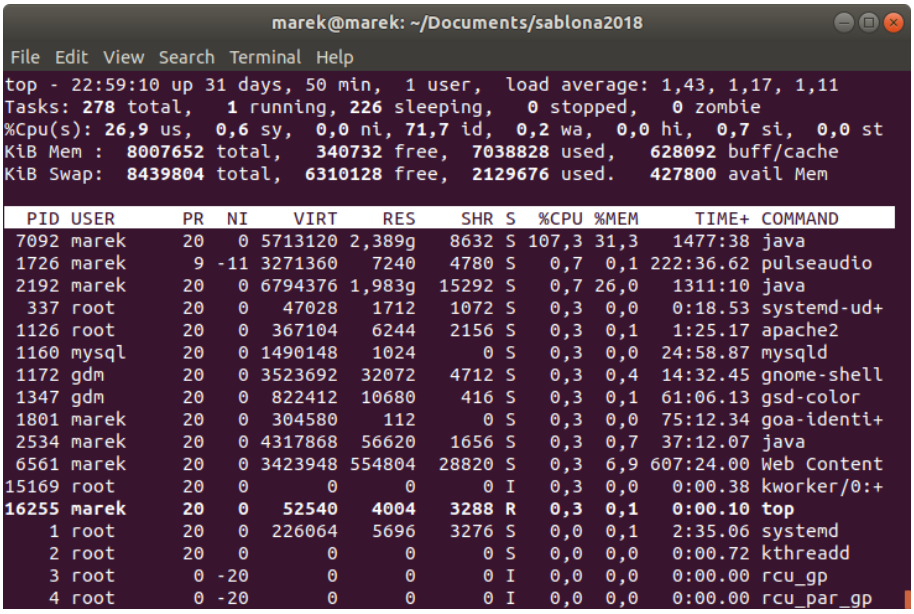


Figure 3.6: CPU usage during the test execution

3.7.2 Other issues

The issues touched in this section relate to the usage of the open-source WSQ compression/decompression, or the ImageIO's JPEG compression. The WSQ[9] issues are not mentioned in the Github repository and were encountered during the test execution.

To start with, the WSQ's *encode()* method only works for fingerprint images whose BufferedImage representation is of type *TYPE_BYTE*, *TYPE_3BYTE* or *TYPE_4BYTE*. Therefore, if the image is represented by *TYPE_INT* or *TYPE_USHORT* it firstly needs to be converted to the above-mentioned byte types.

Secondly, the WSQ's compress method seems to be having a problem in the quantization part for the encoding rate set to 0.10. Furthermore, the encoder is having trouble with array indexing as the *ArrayIndexOutOfBoundsException* exception is thrown at two different locations. These two locations are in the getLets() and build_huffcodes() methods. These

[9] https://github.com/E3V3A/JMRTD/tree/master/wsq_imageio

exceptions are thrown for the binarized images, or the images with the depth of 1 bit. Moreover, the WSQ encoded image is decoded via the WSQ's *decode()* method. This decoding leads to an image being represented by the Bitmap class. The issue occurred when trying to convert this Bitmap representation to the BufferedImage class. The conversion, as done in the test class[10]:

```
int width = bitmap.getWidth();
int height = bitmap.getHeight();
byte[] data = bitmap.getPixels();
BufferedImage image =
    new BufferedImage(width, height, BufferedImage.TYPE_BYTE_GRAY);
WritableRaster raster = image.getRaster();
Raster.setDataElements(0, 0, width,height,data);
```

can lead to pixels being indexed out of the bounds. The problem is that the newly created BufferedImage does not expect the bitmap to be of different depth than 8 bits. When the bitmap is of different depth, such as 1 bit, then also this depth has to be taken into an account when creating BufferedImage as proposed in my solution:

```
int width = bitmap.getWidth();
int height = bitmap.getHeight();
int depth = bitmap.getDepth();
BufferedImage image = new BufferedImage(width, height, depthToType(depth));
byte[] data = Arrays.copyOf(bitmap.getPixels(), width*height*depth);
image.getRaster().setDataElements(0, 0, width, height, data);
```

As mentioned in 3.2.2 the JPEG issue occurs when an image with an alpha channel is passed for the JPEG compression. Then the *Bogus input colorspace* exception will be thrown by the *JPEGImageWriter* class. To solve this issue, the image's depth must be changed.

[10] https://github.com/E3V3A/JMRTD/blob/master/wsq_imageio/src/org/jnbis/test/WSQTest.java

Chapter 4

Testing

This chapter provides the results gathered during the testing phase. For the testing purposes the implemented application was suggested to be tested on different kinds of fingerprint images. A large database containing fake, damaged, synthetic, diseased, and real fingerprint images was provided for the purpose of this project. There were 22 fake, 167 damaged, 258 sick, 950 real, and 590 synthetic fingerprint images.

4.1 Test cases

To compare the effectiveness of different compression algorithms, all the compressions included in the implemented application (JPEG, PNG, WSQ) were tested with different compression parameters. Before the compression, a filter or a collection of filters will be applied as part of the pre-processing. The application of a filter may speed up the compression process or produce a smaller file size.

Fingerprint images were exposed to 225 test cases. These test cases were composed of different combinations of filters and compressions with their parameters.

The compression and its parameters were set to the following. The JPEG compression was set to be run with the compression ratio set to 0, 0.25, 0.50, 0.75, and 1.0. The same applies to the PNG compression. The WSQ compression was set to be run with the encoding rate set to 0.10, 0.25, 0.50, 0.75, and 1.0.

It is expected that different compression parameters will produce different matching scores. The highest matching scores should be obtained by the compressed images with the compression ratio set to 1.0, while the lowest matching scores should be obtained by the compressed images with the compression ratio set to 0. Other compression ratios are anticipated to produce a compromise between the image quality and file size. The ratio set to 0.75 or 0.50 is still expected to produce an acceptable matching score. The same analogy applies to the encoding rate. The highest matching score is expected to be obtained with the encoding rate set to 1.0 and the lowest matching score is expected for the encoding rate set to 0.10. Setting the encoding rate to 0 will result in an exception being thrown. Thus, this option is excluded.

The matching score should be higher for the lossless compression. For the PNG compression, the MSE can be measured to ensure that truly no information was lost during the compression. Thus, no PSNR matching is needed for the PNG compression.

The filters were set to the following possibilities: no filter set, binarization with the adaptive threshold, changing the depth to 1 bit, changing the depth to 8 bits, reducing

the noise by using the Gabor filters, reducing the noise by using the Gabor filters then binarizing using the adaptive threshold, normalization, normalization then binarizing using the adaptive threshold, normalization then reducing the noise by using the Gabor filters, normalization then reducing the noise by using the Gabor filters then binarizing using the adaptive threshold, normalization then reducing the noise by using the Gabor filters then changing the depth to 1 bit, resizing to the 50% of the original size, resizing to the 80% of the original size, resizing to the 125% of the original size, resizing to the 150% of the original size.

There are numerous reasons behind this filter selection. As mentioned in 2.1.2 it can be harder for some matching algorithms to distinguish between the ridges and the valleys and so binarizing an image can make it easier for the matcher to produce higher scores. Therefore, the selection of the binarization with the adaptive threshold which is done via the OpenCV library. Changing the depth of an image to 1 bit is also binarization but this time it is via a creation of a new BufferedImage class with different depth and the data elements of the transformed image is redrawn into the new one. As mentioned in 2.1.2, the binarization for low-quality fingerprint images may be ineffective so the fingerprint image may stay in gray-scale. If an image is not in gray-scale then it will be converted as many matchers work with gray-scale images. Thus, changing the depth to 8 bits filter. To ensure that no false minutiae are used for image comparison, noise reduction is done using the Gabor filters. Also, to make sure that low-quality images are correctly matched, noise reduction is followed by binarization. As referred to in 2.1.2 the normalization can make the image easier to compare, therefore the normalization filter. Also, after the image is normalized, additional filters may be used to build on the normalized pixel values. To find out, if the normalization affects the binarization or the noise reduction process, binarization with the adaptive threshold, or the Gabor filters were used after the normalization. Moreover, the ultimate combination of the normalization, noise reduction, and binarization was suggested to ascertain if all three filters can be applied together. Finally, resizing an image to various proportions was suggested to see if and how it affects the compression speed, compressed file size, and the matching score.

Each test case contained the minutiae-based and the cross-correlation matcher. Test cases with lossy compressions also contained the PSNR matcher.

The testing composed of creating previously mentioned XMLs with all the possibilities of filters and compressions with their parameters mentioned above. There are 15 filter possibilities, and 5 possibilities of JPEG, PNG, and WSQ compression parameters. That results in 255 test cases. One test case is covered by one XML. Each XML was executed for the source directory containing one of the fingerprint image types.

Furthermore, for each bitmap size, the shortest and the longest compression time will be taken. This bitmap size will be measured for an image without the filter, binarized image, and an image resized to 50% of the original size. The binarized and the resized image should have roughly the same bitmap size, and this bitmap size should be much smaller than the one of the original image. These sizes are measured as the image width times the image height times the image depth in bytes. The times for these bitmap sizes are taken so it can be seen, whether the smaller bitmap truly produces a faster compression time.

4.2 Fake fingerprints results

The test results suggest that the shortest average compression time was received with the JPEG compression, the smallest average file sizes with the WSQ compression, and the best

matches with the original image after the PNG compression was used. The application of filters had almost no effect on the average compression times, for the JPEG compression, as the average compression times are almost the same. The average compression times are shown in the figure B.4. However, the choice of the filter(s) did affect the average file sizes which are shown in figure B.5. Yet the biggest difference can be seen on the average matching scores with the original image B.1, B.2, B.3 where the filter(s) reduced the score. The PNG compression produced the best minutiae-based and cross-correlation average scores, the JPEG compression produced the best PSNR average score.

With the applied filters, the shortest compression time, 0.7 milliseconds, was the same for the JPEG and the PNG compressions. The WSQ compression was slightly behind with the shortest compression time being 2.8 milliseconds. The JPEG and the WSQ compressions achieved this score when the original image was resized to 50% of the original size while the PNG compression achieved this when the image was normalized and then binarized. The longest compression time was 143.9 milliseconds for the PNG compression while the longest compression time for the WSQ compression was 59.1 milliseconds and for the JPEG compression 16.4 milliseconds. Filters that produced these times resized the image to 150% of the original image for the JPEG and the WSQ compressions and reduced the noise for the PNG compression. The best match achieved through the minutiae matcher was for the PNG compressed images with the image undergoing through the gray-scale filter. Meanwhile, the JPEG, and the PNG compressions came with the perfect correlation score, 99%, when the depth of an image was changed to either 1 or 8 bits or the image was normalized. An image with reduced noise and the WSQ compression used produced the smallest file size, 776 Bytes, while an image with reduced noise and the PNG compression produced the biggest file size, 295512 Bytes.

With no applied filters, the shortest compression time was around 2 milliseconds for the JPEG and the PNG compressions. The WSQ's shortest compression times was 8.5 milliseconds. The longest compression time was 71.3 milliseconds for the PNG compression in contrast to the 5.4 milliseconds recorded by the JPEG compression. The smallest file size, 1227 Bytes, was produced with the WSQ compression while the biggest file size, 98807 Bytes, was produced with the PNG compression. The best scores from the minutiae matcher were produced with the PNG compressions. For the correlation matcher, the JPEG, and the PNG compressions produced the best results. The best minutiae-based scores are the same when no filter was applied or the gray-scale filter applied because the original image was already in gray-scale.

The tables B.17, B.18, and B.19 demonstrate compression parameters and filters used for obtaining the shortest compression times while the tables B.20, B.21, and B.22 show compression parameters and filters used for attaining the longest compression times. The tables B.23, B.24, B.25 display compression parameters and filters used for obtaining the smallest file sizes while the tables B.26, B.27, B.28 show compression parameters and filters used for gaining the biggest file sizes. The tables B.1, B.2, B.3 present compression parameters and filters used for achieving the best minutiae scores while the tables B.4, B.5, B.6 show compression parameters and filters used for producing the worst minutiae scores. The tables B.7, B.8, B.9 present compression parameters and filters used for gaining the best correlation scores while the tables B.10, B.11, B.12 demonstrate compression parameters and filters used for receiving the worst correlation scores.

Concerning filter application, one fake fingerprint image produced an exemplary results. As shown in figure 4.1, applying Gabor filter to an image 4.1b produces satisfactory results and the minutiae score for such an image is 120 for JPEG, 101 for PNG, and 2 for the

WSQ compression respectively. However, normalizing the image before the Gabor filter is applied results in more minutiae being visible 4.1c, and thus the minutiae score increases 116 for PNG, and 90 for the WSQ compression respectively. Unfortunately, not all fake fingerprint images reacted positively to this filter combination, and their minutiae score is much smaller.

Concerning the compression times for the bitmap sizes, the JPEG compression recorded slightly higher compression times for the binarized image than for the resized image. Additionally, for the JPEG compression, the fastest compression times for the binarized image were similar to the times of the image without the filter. Thus, the smaller bitmap size did not produce a faster compression time. For the WSQ compression, compressing the binarized image was impossible as an exception was thrown

(a) The original fake fingerprint image. Image taken from the database for testing purposes containing the fake fingerprint images

(b) Image filtered with the Gabor filter

(c) Image normalized before the Gabor filter is applied

Figure 4.1: A figure shows the effect of normalization before the noise is reduced from an image

4.3 Damaged fingerprint results

The test results indicate that the shortest average compression time, with and without the filters, has been achieved with the JPEG compression while the longest average compression time, with and without the filters, has been achieved with the WSQ compression. The average compression times can be seen in the figure C.4. The application of the filters had almost no effect on the average file size for the WSQ compression while the biggest difference can be seen for the PNG compression. The average file sizes can be seen in the figure C.4. However, the application of the filters resulted in significantly lower matching scores. While all the compressions without the filters produced satisfactory average minutiae scores, only the PNG compression recorded a suitable average correlation score. Furthermore, the PNG compression with the applied filters also produced a solid average minutiae score. The

average minutiae and correlation scores are shown in the figures C.1 and C.2 respectively. The WSQ compression has recorded higher average PSNR scores, with and without the filters, than the JPEG compression. These average scores are shown in C.3.

With the applied filters, an image resized to 50% of the original image and JPEG compression recorded the shortest compression time of 0.4 milliseconds. The longest compression time, 163 milliseconds, was produced with the PNG compression. Without the applied filters the JPEG and the PNG compressions retained their best compression time close to 1 millisecond. The longest compression time, 79 milliseconds was produced by the PNG compression. The tables C.17, C.18, C.19, show the compression parameters and filters used for obtaining the shortest compression times whereas the tables C.20, C.21, C.22 present the compression parameters and filters used for gaining the longest compression times.

When the filters were applied the smallest file size, 710 Bytes, was achieved by the WSQ compression, and an image resized to 50% of the original size. The biggest file size, 519907 Bytes, was produced by the PNG compression. When the filters were not applied the smallest file size, 1065 Bytes, was also produced by the WSQ compression. The biggest file size, 173625 Bytes, was produced by the PNG compression. The compression parameters and filters used for obtaining the smallest or the biggest file sizes are shown in the tables C.23, C.24, C.25 and C.26, C.27, C.28 respectively .

Regarding the minutiae score, an image with the depth changed to 8 bites, and the PNG compression used produced the best results. For the correlation score, all the compressions with the applied filters managed to get the perfect score of 99%, or 98% match with the original image. Without the applied filters, the PNG compression achieved the best minutiae and correlation results. The original image had already had the depth of 8 bits therefore, the filter that changed the depth to 8 bits had no effect. The compression parameters and filter used for obtaining the best and the worst minutiae scores are presented in the tables C.1, C.2, C.3, and C.4, C.5, C.6, respectively. The tables C.7, C.8, and C.9 demonstrate the compression parameters and filters used for obtaining the best correlation scores whereas the table C.10, C.11, and C.12 show the compression parameters and filters used for receiving the worst correlation scores.

For the JPEG and the PNG compression, the binarized image recorded slightly higher compression times than the resized image. For the WSQ compression, compressing the binarized image was impossible as an exception was thrown. Furthermore, for the JPEG compression, the compression times for the binarized image were similar to the times of the image without the filter. Thus, the smaller bitmap size did not produce a faster compression time.

4.4 Real fingerprints results

In contrast to other fingerprint types, real fingerprints have produced a solid average score for the minutiae-based matching score with the applied filters. As seen in D.1, the highest average score with and without filters has been registered by the PNG compression. However, all compressions have produced unsatisfactory average cross-correlation scores, as shown in D.2. Concerning the PSNR scores, the WSQ compression has recorded a higher average score without the filters, but the JPEG compression has registered a higher average score with the filters. The PSNR average scores are displayed in D.3. The JPEG compression has recorded the fastest average compression times with and without the filters, shown in D.4, while the WSQ compression created the smallest file sizes on average, shown in D.5.

The PNG compression with all compression ratios has registered the best minutiae-based matching score of 1179 for an image with the depth changed to 8 bits. All compressions have the same best score either when no filter was applied or when the depth of an image was changed to 8 bits. The original image depth was 8 bits already, therefore the filter did not affect it. The best and the worst minutiae-based scores can be seen in tables D.1, D.2, D.3 and D.4, D.5, D.6 respectively.

Resizing an image to 50% of the original size resulted in compression times being less than 1 millisecond for the JPEG and the PNG compression. When no filter was applied, the PNG compression with the compression ratio set to 100 had produced the shortest compression time. The WSQ compression has had the worst shortest compression times. The compression ratios and the encoding rates that produced these times appear to be ambiguous. While for the shortest compression times for the JPEG and WSQ these compression parameter values are close to zero, the PNG compression ratio is 100. The shortest and the longest compression times are presented in tables D.17, D.18, D.19 and D.20, D.21, D.22 respectively.

Unlike other fingerprint types, the smallest file size was produced by an image with the depth changed to 1bit and the PNG compression. The compression ratios and encoding rates that produced the smallest file sizes were set to 0, 25, or 0.10 respectively, while the compression ratios and encoding rates that produced the biggest file sizes were set to 100 or 1.0 respectively. The smallest and the biggest file sizes are shown in D.23, D.24, D.25, and D.26, D.27, D.28 correspondingly.

For the JPEG and the PNG compression, the binarized image recorded slightly higher compression times than the resized image. For the WSQ compression, compressing the binarized image was impossible as an exception was thrown. Furthermore, for the JPEG compression, the fastest compression times for the binarized image were similar to the times of the image without the filter, and the longest compression times were interestingly higher for the binarized image. Hence, for the JPEG compression, the smaller bitmap size did not produce faster compression time.

4.5 Ill fingerprint results

As with the other fingerprint types, the highest average scores, for the minutiae-based and the cross-correlation matching, were produced by the PNG compression. Also, the average values for the PNG and the JPEG compressions were higher when no filters were applied. Only the WSQ compression had higher average values with the applied filters, however, its average scores were significantly lower than other compressions. The average minutiae-based and the average cross-correlation scores are shown in E.1 and E.2 respectively. For the PSNR matcher, the higher average score has been registered by the JPEG compression, shown in E.3. The shortest average compression times were recorded by the JPEG compression. The average compression times are shown in E.4. The smallest average compressed file sizes were produced by the WSQ compression. These average file sizes are shown in E.5.

The compression ratios and the encoding rates that produced the best minutiae-based scores were set to 100 or 1.0 respectively. Also, the PNG compression managed to obtain the best score with no applied filter when the compression ratio was set to 0. The compression ratios and the encoding rates for the best, and the worst minutiae-based score are presented in E.1, E.2, E.3, and E.4, E.5, E.6 respectively.

When filters were applied the best cross-correlation scores were produced by images with the depth changed to 1 bit or 8 bits. All tested ratios produced the best cross-correlation

scores for the PNG compression, and for the JPEG compression with the depth changed to 1 bit. The encodings rates were either 1.0 for the WSQ compression, and the ratios were set to 100 for the JPEG compression, and the. The ratios and rates that produced the best and the worst cross-correlation scores are presented in E.7, E.8, E.9 and E.10, E.11, E.12 correspondingly.

As predicted, the shortest compression times were registered after an image was resized to 50% of the original size. For the JPEG and the WSQ compressions, the compression ratios and the encodings rates that produced the shortest times were 0 or 1.0, while for the PNG compression the compression ratio was set to 100. In contrast, the longest compression times were produced with the ratio set to 100, and the rate set to 1.0 for the JPEG and the WSQ compression, while the PNG compression had the ratio set to 0. The compression ratios and the encodings rates for the shortest and the longest compression times are demonstrated in E.17, E.18, E.19, and E.20, E.21, E.22 respectively.

Unsurprisingly, the values of the compression ratios or the encoding rates which produced the smallest file sizes were set to 0 or 0.1 respectively. When filters were applied, an image was either resized to 50% of the original size or had the depth changed to 1 bit, as shown in E.23, E.24, E.25. Additionally, the compression ratios and the encoding rates for the biggest file sizes were set to 100 or 1.0, as shown in E.26, E.27, E.28.

The binarized image recorded slightly higher compression times than the resized image, for the JPEG compression. However, the fastest compression times for the PNG compression were higher for the resized image. For the WSQ compression, compressing the binarized image was impossible as an exception was thrown. Moreover, for the JPEG compression, the compression times for the binarized image were similar to the times of the image without the filter. So, for the JPEG compression, the smaller bitmap size did not produce faster compression time.

4.6 Synthetic fingerprint results

All compressions managed to obtain satisfactory average minutiae-based scores, whether with or without the filters. By satisfactory score, a value at least two times as high as the minimal value for images to be considered a match is meant. Once again, the highest average scores were obtained by the PNG compression. The average minutiae-based scores are shown in F.1. Concerning the average cross-correlation score values, only the PNG compression without the applied filters managed to register the acceptable value. Other compression values were all under 50%. The average cross-correlation scores are displayed in F.2. The JPEG compression recorded higher average scores for the PSNR matching, which are displayed in F.3. One more time, the shortest average compression times were produced by the JPEG compression, and the longest average compression times were obtained by the WSQ compression. These average compression times are shown in F.4. However, the smallest average file sizes were produced by the WSQ compression, while PNG has registered the largest average value. The average file sizes are shown in F.5.

There are certain vaguenesses concerning the compression times as the compression ratio set to 0 for the JPEG compression and the encoding rate set to 0.1 for the WSQ compression produced the shortest compression times. Yet, the PNG compression registered the shortest compression time when the ratio was set to 100. However, the ratio of 0 or 100 also produced the longest compression times. The shortest and the longest compression times with their ratios and rates are presented in F.17, F.18, F.19 and F.20, F.21, F.22 respectively.

If the filters were applied, the smallest file sizes were produced when an image was resized to 50% of the original size, had the depth changed to 1 bit, or was normalized and had the noise reduced. The compression ratios for these smallest sizes were set to 0, the encoding rates to 0.1. For the biggest file sizes, the compression ratios were set to 100, and the encoding rates were set to 1.0. The ratios and the encodings that produced the smallest or the biggest file sizes are presented in F.23, F.24, F.25 or F.26, F.27, F.28 respectively.

The best minutiae-based scores are the same whether the gray-scale filter was applied, or not. Meaning that the original image had its depth at 8bits. For the JPEG compression, the ratio of 100 and for the WSQ the encoding of 0.75 produced the best score whether the filters were or were not applied. While for the PNG, all tested ratios produced the best score.

The binarized image recorded slightly higher compression times than the resized image, for the JPEG compression. However, the fastest compression times for the PNG compression were higher for the resized image. Moreover, for the JPEG compression, the fastest compression times for the binarized image were similar to the times of the image without the filter. So, for the JPEG compression, the smaller bitmap size did not produce faster compression time. Additionally, it is worth mentioning that there are 3 different bitmap sizes of the binarized image while there are 4 different sizes of resized and non-filtered image. This is because there were images with a depth of 24 bits, and their bitmap size was 698880 bytes. For them to be binarized, they first need to be converted to gray-scale, thus their bitmap size becomes 232960 bytes. And since there already are gray-scale images with the bitmap size of 232960 bytes, they both produce the same binarized bitmap size. For the WSQ compression, compressing the binarized image was impossible as an exception was thrown.

4.7 Result summary and discussion

For real and synthetic fingerprints, with and without the pre-processing phase, matched by the minutiae-based technique, almost all compressions recorded the average score at least twice the minimum required score for them to be considered a match. However, the average scores from the cross-correlation matching technique suggest that only the PNG compression without the pre-processing phase produced acceptable results. The average minutiae-based scores for other fingerprint types suggest that the JPEG and the PNG compressions, without the pre-processing, can be used to obtain adequate results. The average cross-correlation results imply that only the PNG compression, without the pre-processing, produces satisfactory results. As mentioned in [15], the minutiae-based technique is the most used one, so results from this category could have a higher value. But the only compression, that produced satisfactory results with both matching techniques on all fingerprint types is the PNG compression without the pre-processing phase.

Unlike in [11], the fingerprint enhancement did not improve the matching scores. The average matching scores suggest that pre-processing an image may lower the matching score. Furthermore, results in [21] indicate that compressions based on the Discrete wavelet transform keep better image quality. This study, on the other hand, suggests that the WSQ compression, based on the Discrete wavelet transform, may not provide such image quality as lossless compressions like PNG. Although, it is worth noting the used WSQ compression was error-prone, thus, the full capabilities of the WSQ compression could not be tested.

Without the applied filters, the JPEG compression had higher average PSNR scores for 3 out of 5 fingerprint types. Also, these average PSNR values produced by the JEPG

compression for the fake fingerprints, and the WSQ compression for the real fingerprints were similar to those in [21]. In that study, the PSNR value for the JPEG compression was 24.42, and 19.86 for the compression based on the Discrete wavelet transform. The synthetic fingerprints produced slightly lower average PSNR values but still close to the ones produced in [21]. For the PNG compression, the PSNR values were not measured. PNG is a lossless compression, and therefore there is no difference between the original and the reconstructed image. To test this, the MSE 2.5 was measured for 100 randomly selected images. Every one of these images produced MSE of 0 meaning, no difference between the original and the reconstructed image.

Nevertheless, the PNG compression does not provide the reduction capabilities or the compression times as the other compression methods in this study. The results of this work indicate that the ideal compression, concerning the compression times, is the JPEG compression, which produced the best average compression times. Yet the best compression, concerning the compressed file sizes, is the WSQ compression that produced the smallest average file sizes. For the JPEG compression, the compression ratio set to 0 produced the shortest compressions and the smallest file sizes, while the compression ratio of 100 produced the biggest file sizes and the longest compression times. For the WSQ compression, the encoding rate set to 0.1 produced the smallest file sizes and the shortest compression times, while the encoding rate of 1.0 produced the biggest file sizes and the longest compression times. However, for the PNG compression, the compression ratio set to 0 produced the longest compression times and the smallest file sizes, while the compression ratio of 100 produced the shortest compression times and the biggest file sizes. As stated in JavaDoc[1], this ratio is a compromise between the file size and the compression time, for the lossless compression. Meaning that higher ratio values produce a shorter time and a bigger file size. For the lossy compression, this ratio is a compromise between the file size and the image quality. Meaning that lower ratio values produce a smaller file size and a worse quality image.

The results also suggest that pre-processing an image, like changing its size to 50% of the original size, or changing its depth to 1 bit may reduce the compression time or the file size. Still, it may not be ideal, as such images did not register acceptable matching scores. Only the ill, synthetic, and fake fingerprint images with the depth changed to 1 bit compressed by the JPEG or the PNG compression recorded satisfactory scores by the cross-correlation matching technique.

Interesting results were provided by the compression times for respective bitmap sizes. Almost in all cases, the binarized image had higher compression times than the image resized to 50% of the original size, albeit the binarized image had the smaller bitmap size. Also, for the JPEG compression, the compression time for the binarized image was similar to the compression time of an image with no filter applied. It is worth noting that different compression times were obtained during various test runs, indicating that the compression times are not deterministic, which may explain the roughness in the graphs. The binarized images were not compressible with the WSQ compression as the compression threw ArrayIndexOutOfBoundsException exception.

As mentioned in [1], the pre-processing should eliminate the false minutiae and improve the matching. However, their pre-processing also included thinning the fingerprint image, which this implemented application does not support. In addition to that, the implemented Gabor filter does not always produce relevant images, as displayed in 3.5. This Gabor filter

[1] https://docs.oracle.com/javase/7/docs/api/javax/imageio/ImageWriteParam.html#setCompressionQuality

works with OpenCV methods, which require an image to be represented by OpenCV's *Mat* class. However, in the implemented application an image is represented by Java's BufferedImage class. The conversion between these classes may be the reason why the resulting filtered image is sometimes unusable. Apart from that, the other issue causing this problem may be the partial images being sometimes wrongly added together.

As seen in the tables for the worst matching scores, any combination of compression, compression parameter, and applied filter can produce a matching score of 0. There are a few reasons for this. According to [11] the nature of the ridges differs with the input fingerprint image. Therefore, the matchers will have trouble with less quality images. Less error rate is accomplished with the quality images, which is stated in [26]. Also, some types of fingerprint images are just not ideal for matching. According to [27], the obfuscated, or damaged fingerprints are harder to match. Additionally, the matchers that were used are open-source fingerprint matchers. They may not possess the matching quality of the commercial matchers. It is not stated on their respective project websites how they handle low-quality fingerprint images.

Chapter 5

Conclusion

The purpose of this work was to determine how different compression methods affect different types of fingerprint images and how different fingerprint matching techniques cope with these compressed images. This study was conducted on the database consisting of fake, damaged, ill, synthetic, and real fingerprint images. The compression methods included JPEG, PNG, and the WSQ compression, and matching included the minutiae-based, the cross-correlation, and the PSNR matcher. The results showed that the PNG compression registered the most satisfying and acceptable average scores. Meanwhile, the JPEG compression recorded the best average compression times, and the WSQ compression produced the best average file sizes. It was suggested that applying various fingerprint enhancement techniques could improve the matching scores. However, applying these pre-processing techniques to an image produced significantly lower average scores.

This work could be further expanded by including the JPEG2000 compression to the used compressions. This compression provides lossy and lossless compressions, and therefore could match the average values achieved by the PNG compression. Also, the preprocessing techniques could be broadened by the thinning filter. Applying such a filter to fingerprint images may increase the performance of the matching algorithms. Likewise, the performance of the implemented noise reduction filter could be improved, so more relevant results are produced. Furthermore, a better implementation of the WSQ compression should be used since one that was used was prone to errors.

Bibliography

[1] Akram, M., Tariq, A., Khan, S. and Nasir, S. Fingerprint image: pre- and post-processing. *International Journal of Biometrics*. january 2008, vol. 1.

[2] Alonso Fernandez, F., Fierrez, J. and Ortega Garcia, J. An enhanced Gabor filter-based segmentation algorithm for fingerprint recognition systems. In: *Conference: Image and Signal Processing and Analysis*. October 2005, p. 239 – 244. DOI: 10.1109/ISPA.2005.195416. ISBN 953-184-089-X.

[3] Bansal, R., Sehgal, P. and Bedi, P. Minutiae Extraction from Fingerprint Images - a Review. *International Journal of Computer Science Issues*. november 2011, vol. 8.

[4] Cabeen, K. and Gent, P. Image Compression and the Discrete Cosine Transform. In: *Math 45* [online]. College of Redwoods. Available at: https://www.math.cuhk.edu.hk/~lmlui/dct.pdf.

[5] Cadd, S., Islam, M., Manson, P. and Bleay, S. Fingerprint composition and ageing: A literature review. *Science and Justice*. Jul 2015, vol. 55, p. 219–238.

[6] Drahanský, M. and Orság, F. *Biometrie*. 1st ed. Brno: Computer Press, 2011. ISBN 978-80-254-8979-6.

[7] Erdmann, C. *Finally understanding PNG* [online]. Compress-or-Die, 2017 [cit. 2020-05-14]. Available at: https://compress-or-die.com/Understanding-PNG.

[8] Gao, Q. A Preliminary Study of Fake Fingerprints. *International Journal of Computer Network and Information Security*. november 2014, vol. 6, p. 1–8. DOI: 10.5815/ijcnis.2014.12.01.

[9] Goljan, M., Chen, M., Comesana, P. and Fridrich, J. Effect of Compression on Sensor-Fingerprint Based Camera Identification. *Electronic Imaging*. february 2016, vol. 2016, p. 1–10. DOI: 10.2352/ISSN.2470-1173.2016.8.MWSF-086.

[10] Harmon, K. *Can You Lose Your Fingerprints?* [online]. Scientific American, 2009 [cit. 2020-04-16]. Available at: https://www.scientificamerican.com/article/lose-your-fingerprints/.

[11] Hong, L., Wan, Y. and AK, J. Fingerprint Image Enhancement: Algorithm and Performance Evaluation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. september 1998, vol. 20, p. 777 – 789. DOI: 10.1109/34.709565.

[12] Hussain, A., Al Fayadh, A. and Radi, N. Image compression techniques: A survey in lossless and lossy algorithms. *Neurocomputing*. Jul 2018, vol. 300, p. 44–69.

[13] Jaiswal, R. and Saxena, G. Biometric Recognition System (Algorithm). In: *Conference: EMIE 2016*. June 2017. ISBN 978-93-5260-435-7.

[14] Joshi, P. *Understanding Gabor filters* [online]. 2014 [cit. 2020-03-28]. Available at: https://prateekvjoshi.com/2014/04/26/understanding-gabor-filters/.

[15] Kanjan, N., Patil, K., Ranaware, S. and Sarokte, P. A Comparative Stude of Fingerprint Matching Algorithms. *Internation Research Journal of Engineering and Technology*. november 2017, vol. 4, p. 1892–1896.

[16] Libert, J. M., Orandi, S. and Grathm, J. D. *Comparison of the WSQ and JPEG2000 Image Compression Algorithms On 500 ppi Fingerprint Imagery*. NIST Interagency Report 7781, Apr 2012.

[17] Maltoni, D., Maio, D., Jain, A. and Prabhakar, S. Fingerprint Analysis and Representation. In: *Handbook of Fingerprint Recognition*. New York, NY: Springer New York, 2003, p. 83–130. DOI: 10.1007/0-387-21587-5_3. ISBN 978-0-387-21587-7. Available at: https://doi.org/10.1007/0-387-21587-5_3.

[18] Maltoni, D., Maio, D., Jain, A. and Prabhakar, S. Synthetic Fingerprint Generation. In: *Handbook of Fingerprint Recognition*. April 2006, p. 203–231. DOI: 10.1007/0-387-21587-5_6.

[19] McAnlis, C. *How PNG Works* [online]. Medium, 2016 [cit. 2020-05-14]. Available at: https://medium.com/@duhroach/how-png-works-f1174e3cc7b7.

[20] Mehmandoust, S. and Shahbahrami, A. A Comparison between Different Fingerprint Matching Techniques. In: Cherifi, H., Zain, J. M. and El Qawasmeh, E., ed. *Digital Information and Communication Technology and Its Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, p. 242–253. ISBN 978-3-642-21984-9.

[21] Mozammel, M., Chowdhury, H. and Khatun, A. Image Compression Using Discrete Wavelet Transform. *International Journal of Computer Science Issues*. july 2012, vol. 9.

[22] Murthy, K. *GABOR FILTERS : A PRACTICAL OVERVIEW* [online]. Wordpress, 2014 [cit. 2020-03-05]. Available at: https://cvtuts.wordpress.com/2014/04/27/gabor-filters-a-practical-overview/.

[23] Pandey, V. Analysis of Image Compression using Wavelets. *Internation Journal of Computer Applications*. Oct 2014, vol. 103, p. 1–8.

[24] Pathak, P. Image Compression Algorithms for Fingerprint System. *Internation Journal of Computer Science Issues*. May 2010, vol. 7, p. 45–50.

[25] Seow, B., Yeoh, S., Lai, S. and Abu, N. Image based fingerprint verification. In: *Student Conference on Research and Development*. IEEE, February 2002, p. 58 – 61. DOI: 10.1109/SCORED.2002.1033054. ISBN 0-7803-7565-3.

[26] Simon, D., Ortega Garcia, J., Fierrez, J. and Gonzalez Rodriguez, J. Image quality and position variability assessment in minutiae-based fingerprint verification. *Vision, Image and Signal Processing, IEE Proceedings* -. january 2004, vol. 150, p. 402 – 408. DOI: 10.1049/ip-vis:20031037.

[27] Yoon, S., Feng, J. and Jain, A. Altered Fingerprints: Analysis and Detection. *IEEE transactions on pattern analysis and machine intelligence*. july 2011, vol. 34, p. 451–64. DOI: 10.1109/TPAMI.2011.161.

Appendix A

Example XMLs

A.1 Testing XML

This section includes an example XML used during the testing. This example XML applies the noise reduction filter to each fingerprint image in the directory in the sourceDir element. Then the PNG compression with the compression ratio set to 50 is applied. Each compressed fingerprint image is then saved to the directory in the destinationDir element. Finally the original image is then compared to the compressed image by the cross-correlation and the minutiae matcher.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<entry>
    <sourceDir>
      /home/marek/Documents/KomplexniPrurez/nemocne_tmp/sourceDir
    </sourceDir>
    <destinationDir>
      /home/marek/Documents/KomplexniPrurez/nemocne_tmp/destinationDir
    </destinationDir>
    <filters>
        <filter>
            <filterName>noiseReduction</filterName>
        </filter>
    </filters>
    <compressions>
        <compression>
            <compressionName>png</compressionName>
            <params>
                <ratio>50</ratio>
            </params>
        </compression>
    </compressions>
    <matchers>
        <matcher>
            <matcherName>crossCorrelation</matcherName>
        </matcher>
        <matcher>
```

```
            <matcherName>minutiaeBased</matcherName>
        </matcher>
    </matchers>
</entry>
```

A.2 Image result XML

This section includes an example XML holds the results for a certain fingerprint image. This example fingerprint image was compressed with the PNG compression and had the noise reduction and the binarization filters applied. The compression time took only 2 milliseconds, the minutiae score was 1.17 and the cross-correlation score was 51.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<result>
  <fileNames>
    <currentFilename>173-p2-sec2-denoised-binarized.png</currentFilename>
    <currentAbsolutePath>
      /home/marek/Documents/KomplexniPrurez/FalesneResults/Falesne2/
      compressed/173-p2-sec2-denoised-binarized.png
    </currentAbsolutePath>
    <originalFilename>173-p2-sec2.png</originalFilename>
    <originalFileAbsolutePath>
      /home/marek/Documents/KomplexniPrurez/Falesne2/173-p2-sec2.png
    </originalFileAbsolutePath>
  </fileNames>
  <appliedFilters>
    <appliedFilter>denoised</appliedFilter>
    <appliedFilter>binarized</appliedFilter>
  </appliedFilters>
  <compressionTime>2</compressionTime>
  <matcherScores>
    <minutiaeBasedMatcherScore>1.1743483788938347</minutiaeBasedMatcherScore>
    <crossCorrelationMatcherScore>51.0</crossCorrelationMatcherScore>
  </matcherScores>
</result>
```

Appendix B

Fake fingerprint results

B.1 Matching scores

B.1.1 The best minutiae-based scores

Ratio	Filters	Score
100	depth changed to 8b	661.3097055229035
100	none	661.3097055229035

Table B.1: The ratios and filters for the best JPEG minutiae-based match

Ratio	Filters	Score
all tested	depth changed to 8b	829.8393484687903
0	none	829.8393484687903

Table B.2: The ratios and filters for the best PNG minutiae-based match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score
0.5	depth changed to 8b	169.56989170131155
0.5	none	169.56989170131155

Table B.3: The encodings and filters for the best WSQ minutiae-based match

B.1.2 The worst minutiae-based score

Ratio	Filters	Score
all tested	all tested	0.0
100	none	0.0

Table B.4: The ratios and filters for the worst JPEG minutiae-based match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Ratio	Filters	Score
all tested	all tested	0.0
0	none	0.0

Table B.5: The ratios and filters for the worst PNG minutiae-based match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score
all tested	resized to 50%	0.0
all tested	resized to 125%	0.0
all tested	normalized	0.0
all tested	resized to 80%	0.0
all tested	denoised	0.0
all tested	normalized, denoised	0.0
all tested	resized to 150%	0.0
all tested	depth changed to 8b	0.0
0.75	none	0.0

Table B.6: The encodings and filters for the worst WSQ minutiae-based match. Encoding rates of 0.10, 0.25, 0.50, 0.75, 1.00 are understood by all tested

B.1.3 The best cross-correlation scores

Ratio	Filters	Score [%]
75, 100	normalized	99.0
50, 75, 100	depth changed to 1b	99.0
100	depth changed to 8b	99.0
100	none	99.0

Table B.7: The ratios and filters for the best JPEG cross-correlation match

Ratio	Filters	Score [%]
all tested	depth changed to 1b	99.0
all tested	normalized	99.0
all tested	depth changed to 8b	99.0
0	none	99.0

Table B.8: The ratios and filters for the best PNG cross-correlation match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score [%]
0.75	normalized	86.0
1.0	none	83.0

Table B.9: The encodings and filters for the best WSQ cross-correlation match

B.1.4 The worst cross-correlation scores

Ratio	Filters	Score [%]
all tested	resized to 50%	0.0
25, 75	depth changed to 8b	0.0
0, 25	normalized, binarized	0.0
0, 50	resized to 80%	0.0
25	normalized, denoised, depth changed to 1b	0.0
0, 25	normalized	0.0
25, 100	resized to 125%	0.0
0	binarized	0.0
0	resized to 150%	0.0
0	normalized, denoised	0.0
0	depth changed to 1b	0.0
75	none	0.0

Table B.10: The ratios and filters for the worst JPEG cross-correlation match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Ratio	Filters	Score [%]
all tested	resized to 50%	0.0
all tested	resized to 80%	0.0
all tested	depth changed to 8b	0.0
0	none	0.0

Table B.11: The ratios and filters for the worst PNG cross-correlation match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score [%]
all tested	resized to 150%	0.0
0.25, 0.5, 0.75	depth changed to 8b	0.0
all tested	resized to 125%	0.0
all tested	resized to 50%	0.0
all tested	resized to 80%	0.0
all tested	normalized, denoised	0.0
all tested	denoised	0.0
0.1, 0.25, 0.5, 0.75	normalized	0.0
0.75	none	0.0

Table B.12: The encodings and filters for the worst WSQ cross-correlation match. Encoding rates of 0.10, 0.25, 0.50, 0.75, 1.00 are understood by all tested

B.1.5 The best PSNR scores

Ratio	Filters	Score [dB]
100	depth changed to 8b	63.702294883802786
100	none	63.702294883802786

Table B.13: The ratios and filters for the best JPEG PSNR match

Encoding	Filters	Score [dB]
0.1	normalized, denoised	17.576663707024306
1.0	none	16.56645179279506

Table B.14: The encodings and filters for the best WSQ PSNR match

B.1.6 The worst PSNR scores

Ratio	Filters	Score [dB]
0	normalized, denoised	7.927548033270092
0	none	11.580130244274024

Table B.15: The ratios and filters for the worst JPEG PSNR match

Encoding	Filters	Score [dB]
0.5	denoised	6.607181826989057
0.1	none	9.549908806047236

Table B.16: The encodings and filters for the worst WSQ PSNR match

B.2 Compression times

B.2.1 The shortest compression times

Ratio	Filters	Time [ms]
0	resized to 50%	0.787652
0	none	2.093098

Table B.17: The ratios and filters for the shortest JPEG compression time

Ratio	Filters	Time [ms]
100	normalized, binarized	0.706703
100	none	1.947856

Table B.18: The ratios and filters for the shortest PNG compression time

Encoding	Filters	Time [ms]
0.1	resized to 50%	2.861224
0.1	none	8.561105

Table B.19: The encodings and filters for the shortest WSQ compression time

B.2.2 The longest compression times

Ratio	Filters	Time [ms]
100	resized to 150%	16.483067
100	none	5.433836

Table B.20: The ratios and filters for the longest JPEG compression time

Ratio	Filters	Time [ms]
0	denoised	143.961452
0	none	71.361985

Table B.21: The ratios and filters for the longest PNG compression time

Encoding	Filters	Time [ms]
1.0	resized to 150%	59.137226
0.75	none	19.503243

Table B.22: The encodings and filters for the longest WSQ compression time

B.3 Compressed file sizes

B.3.1 The smallest file sizes

Ratio	Filters	Size [B]
0	resized to 50%	1199
0	none	2849

Table B.23: The ratios and filters for the smallest JPEG file size

Ratio	Filters	Size [B]
0	depth changed to 1b	1289
0	none	10427

Table B.24: The ratios and filters for the smallest PNG file size

Encoding	Filters	Size [B]
0.1	denoised	776
0.1	none	1227

Table B.25: The encodings and filters for the smallest WSQ file size

B.3.2 The biggest file sizes

Ratio	Filters	Size [B]
100	resized to 150%	287048
100	none	110046

Table B.26: The ratios and filters for the biggest JPEG file size

Ratio	Filters	Size [B]
100	normalized, denoised	295512
100	denoised	295512
100	none	98807

Table B.27: The ratios and filters for the biggest PNG file size

Encoding	Filters	Size [B]
1.0	resized to 150%	23698
1.0	none	7020

Table B.28: The encodings and filters for the biggest WSQ file size

B.4 The average values

B.4.1 The average minutiae-based scores

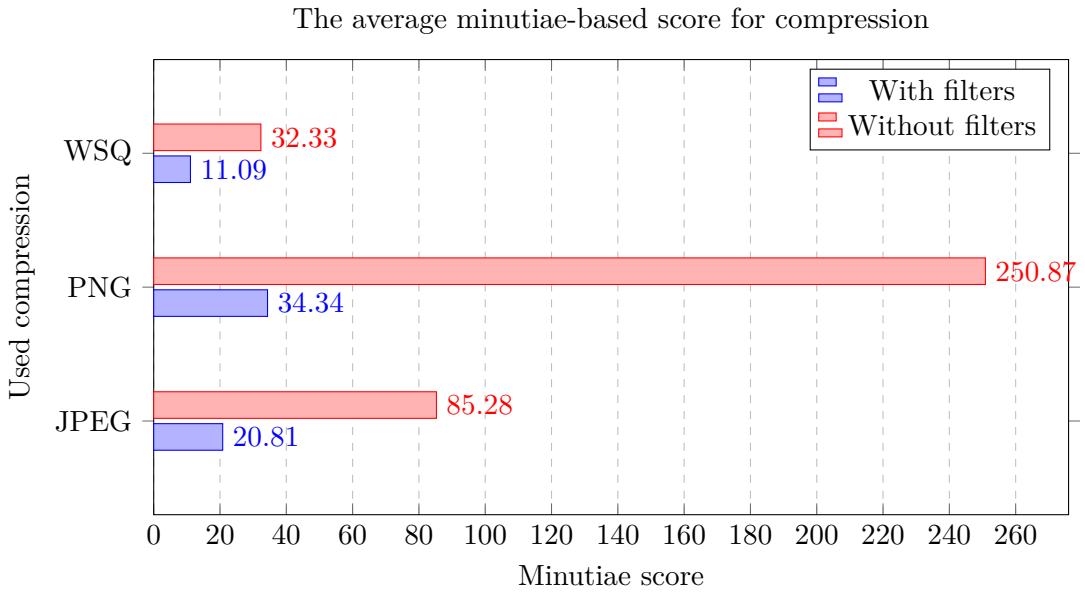


Figure B.1: The average minutiae-based score for compression. Score more than 40 means the fingerprints matched.

B.4.2 The average cross-correlation scores

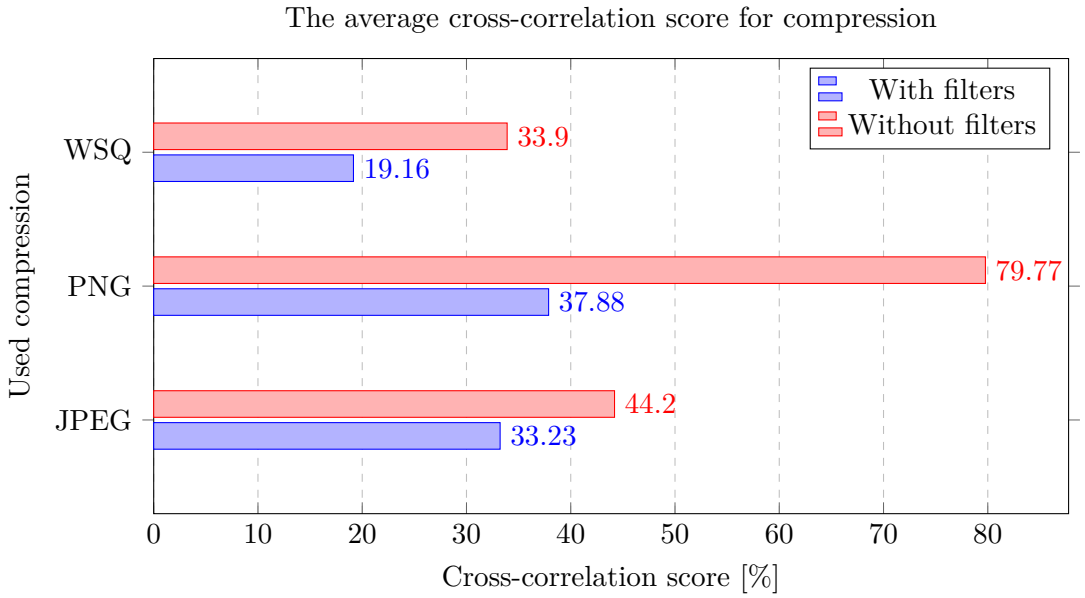


Figure B.2: The average cross-correlation score for compression. Score expresses percentage match between fingerprints.

B.4.3 The average PSNR scores

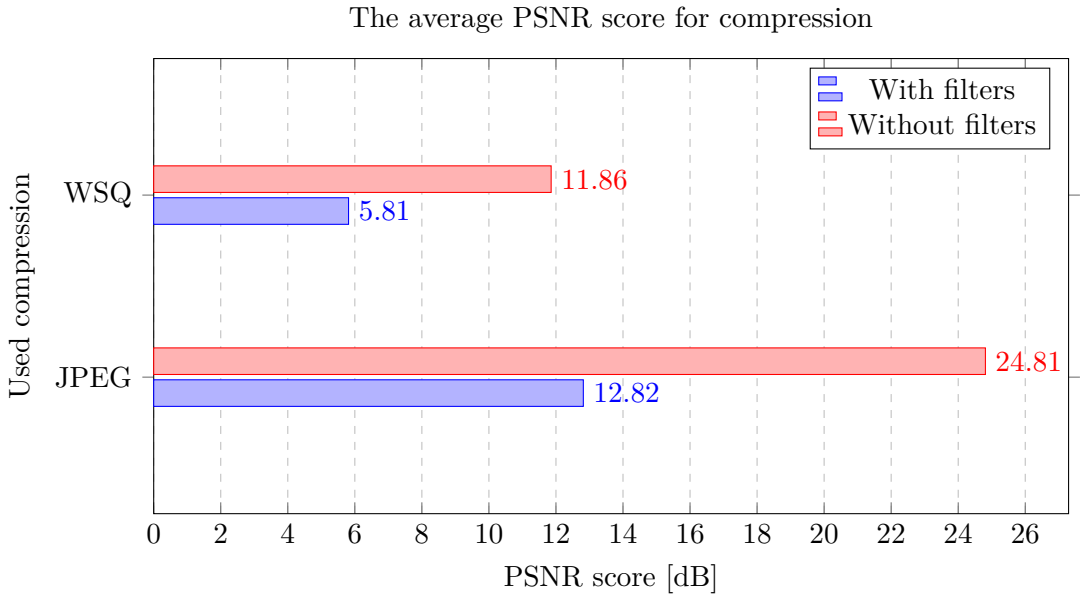


Figure B.3: The average PSNR score for compression. Score more than 40 means the fingerprints matched.

B.4.4 The average compression times

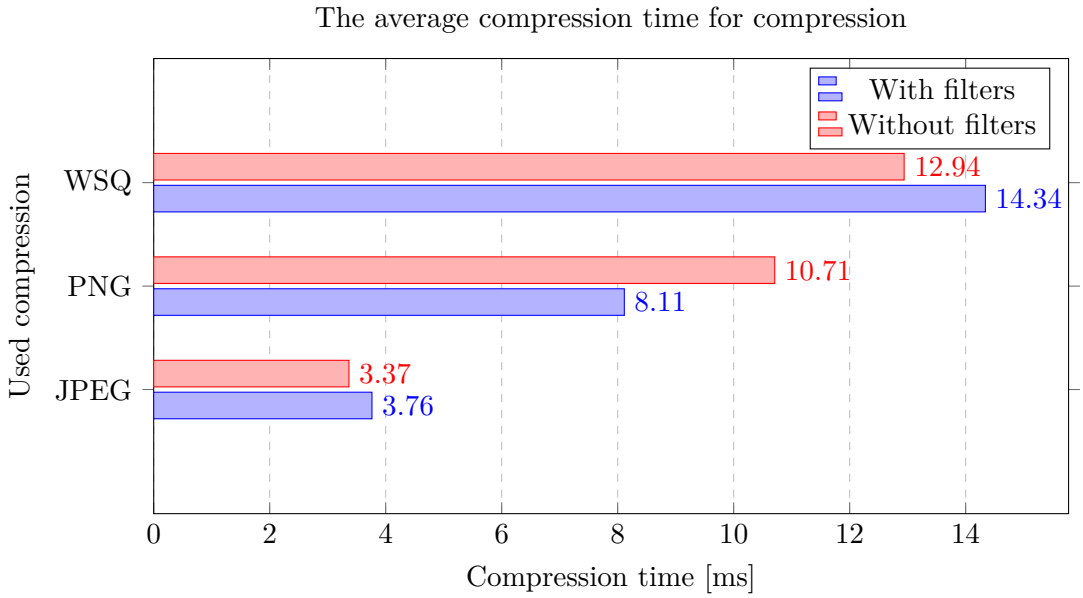


Figure B.4: The average compression time for compression in milliseconds.

B.4.5 The average compressed file sizes

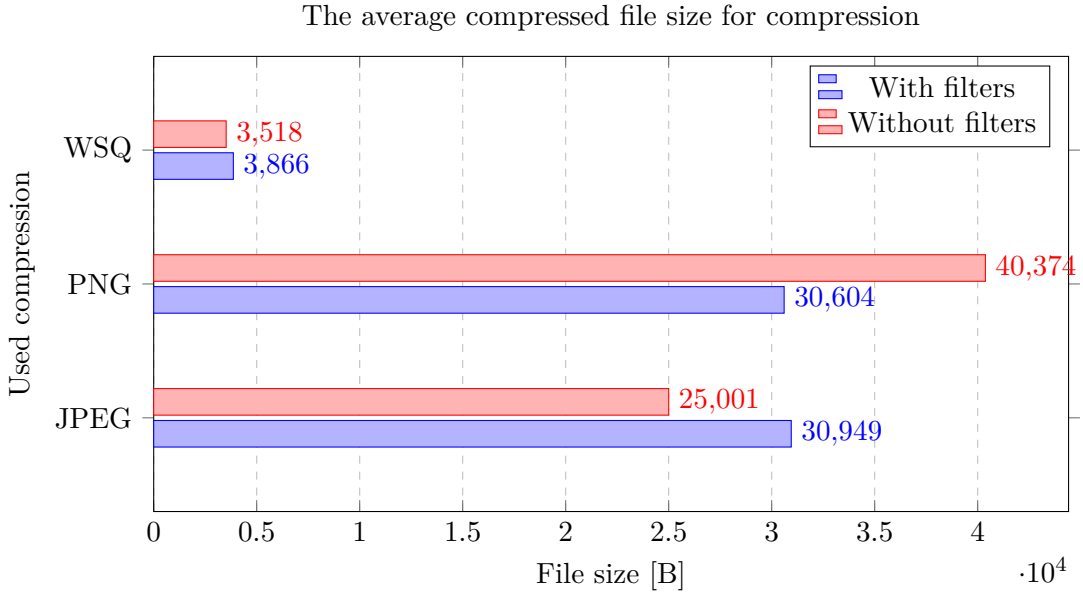


Figure B.5: The average compressed file size for compression in Bytes

B.5 Compression time dependency on the bitmap size

B.5.1 The fastest JPEG compression times for bitmap size

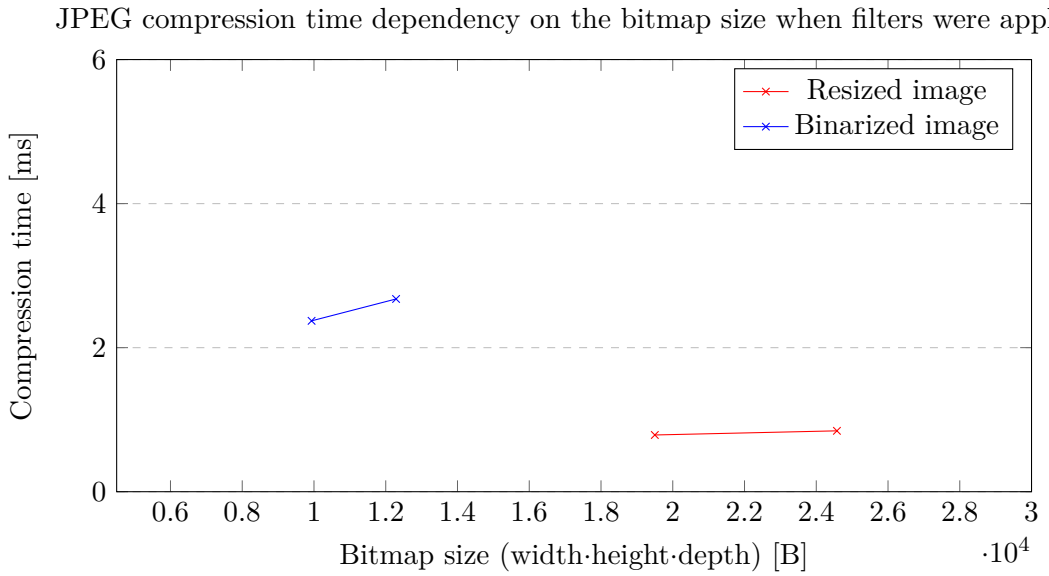


Figure B.6: Plot shows the fastest JPEG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

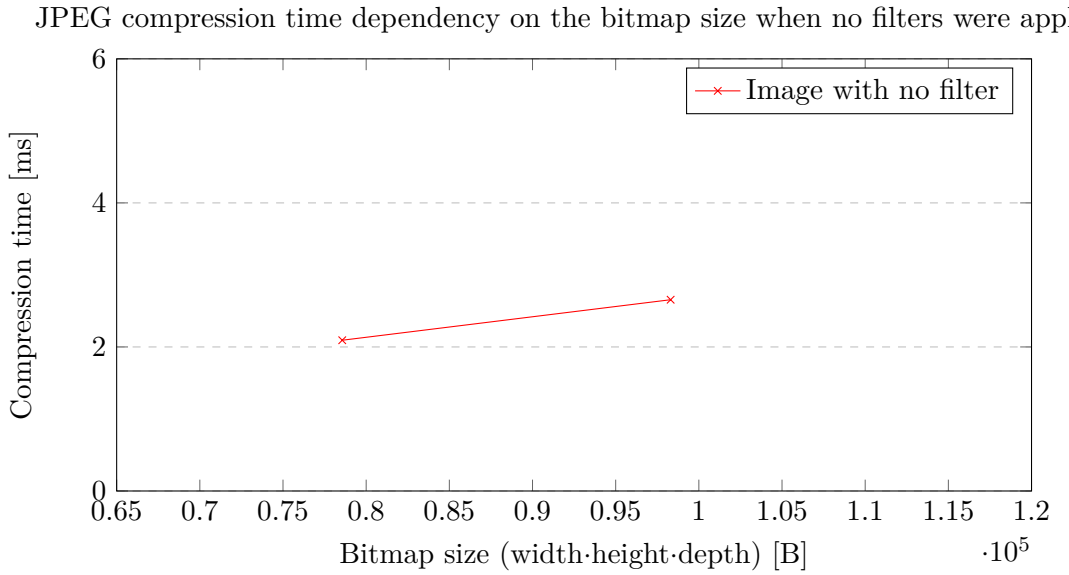


Figure B.7: Plot shows the shortest JPEG compression time for each bitmap size when image was not pre-processed (filters were not applied)

B.5.2 The longest JPEG compression times for bitmap size

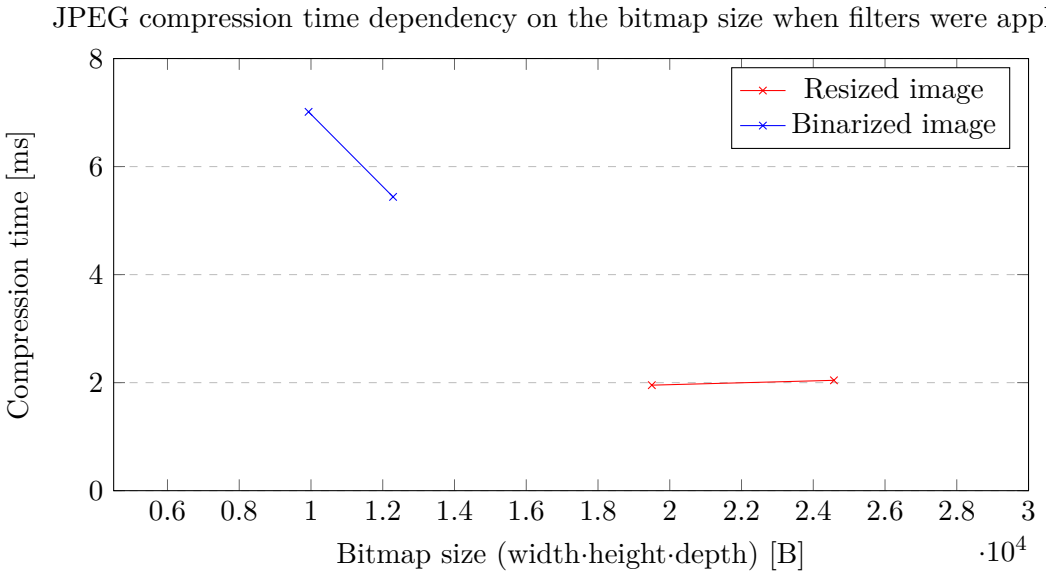


Figure B.8: Plot shows the longest JPEG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

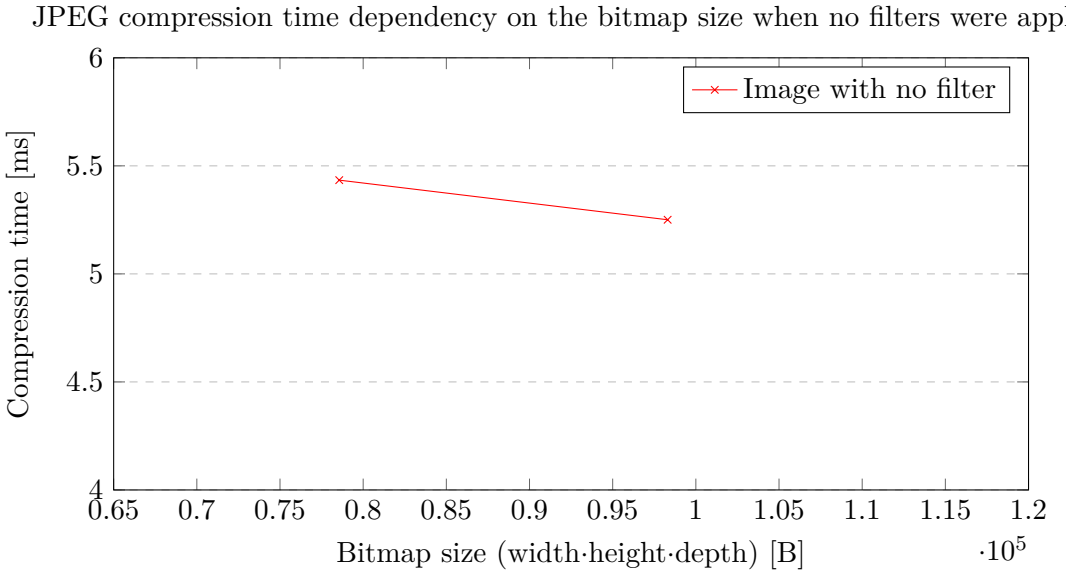


Figure B.9: Plot shows the longest JPEG compression time for each bitmap size when image was not pre-processed (filters were not applied)

B.5.3 The fastest PNG compression times for bitmap size

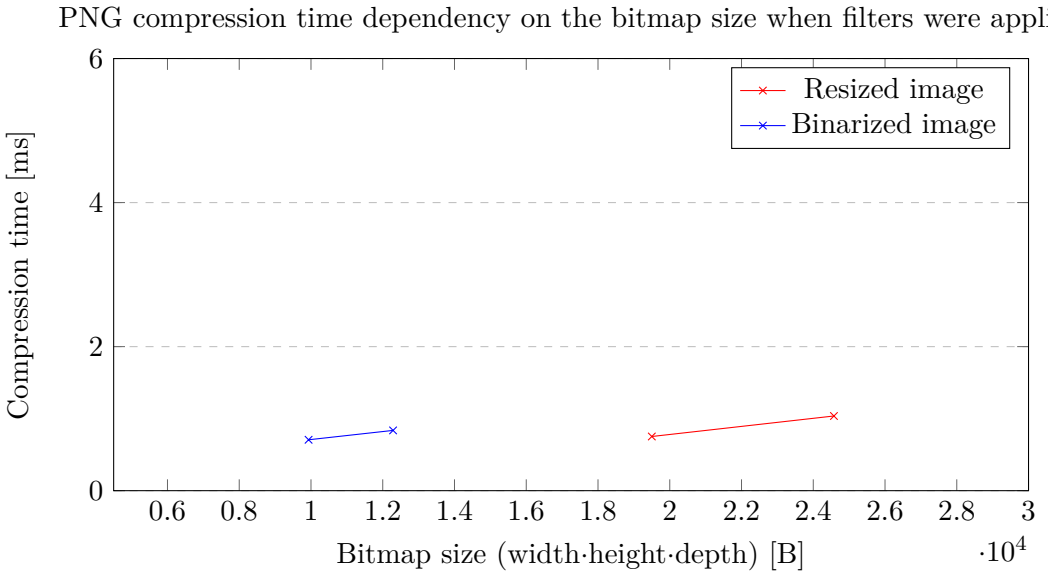


Figure B.10: Plot shows the shortest PNG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

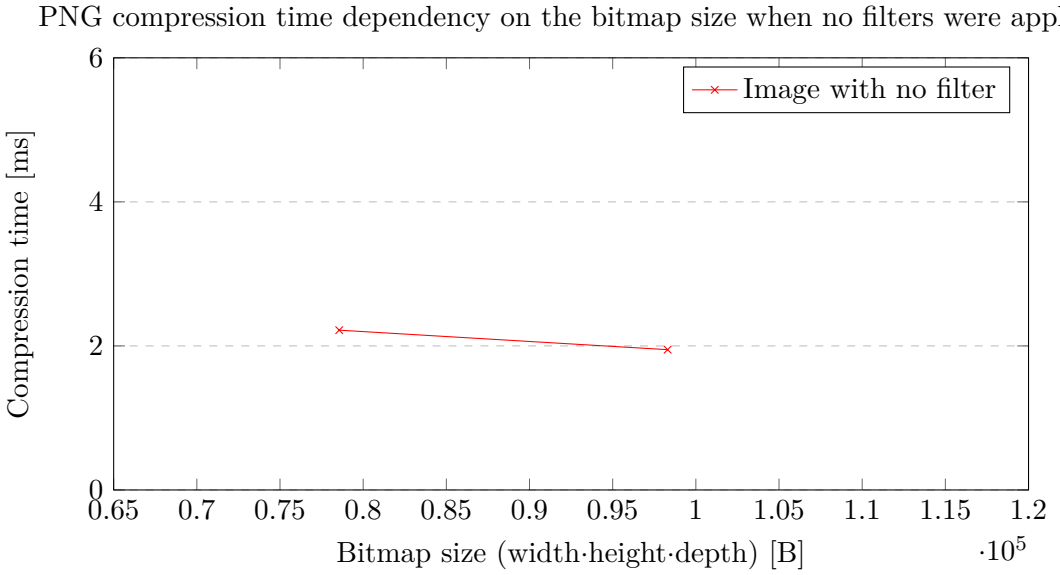


Figure B.11: Plot shows the shortest PNG compression time for each bitmap size when image was not pre-processed (filters were not applied)

B.5.4 The longest PNG compression times for bitmap size

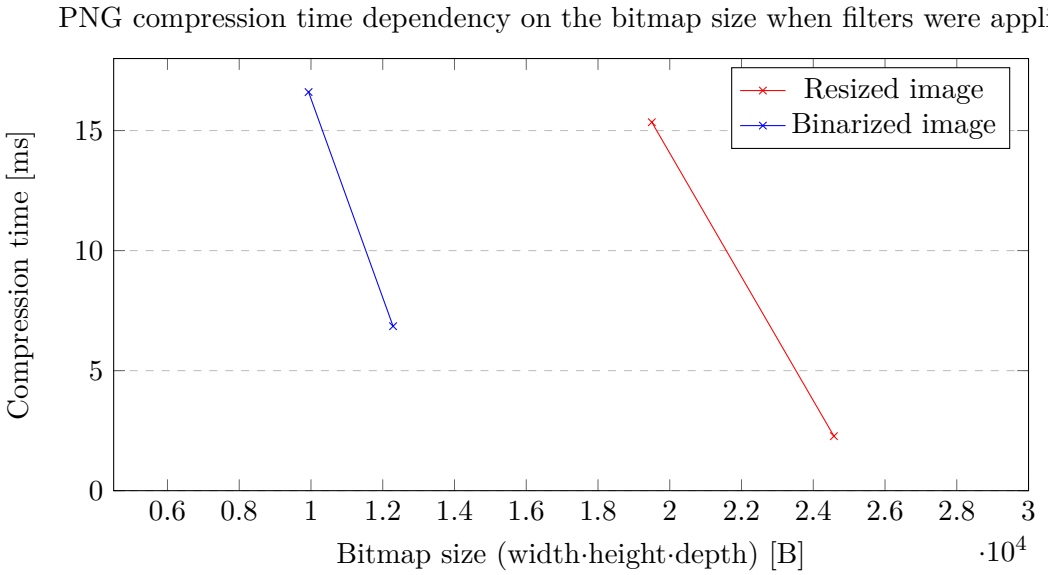


Figure B.12: Plot shows the longest PNG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

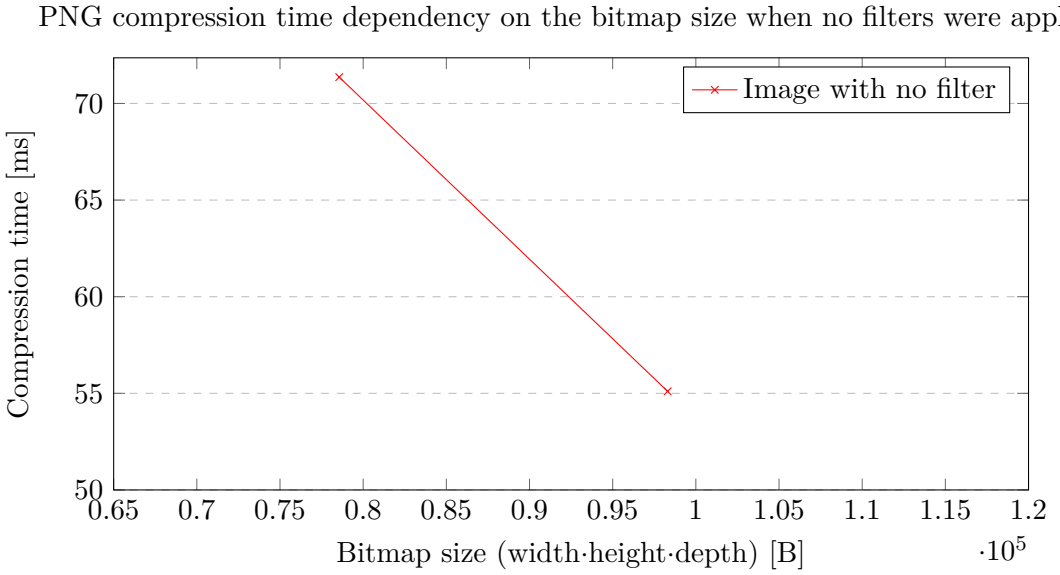


Figure B.13: Plot shows the longest PNG compression time for each bitmap size when image was not pre-processed (filters were not applied)

B.5.5 The fastest WSQ compression times for bitmap size

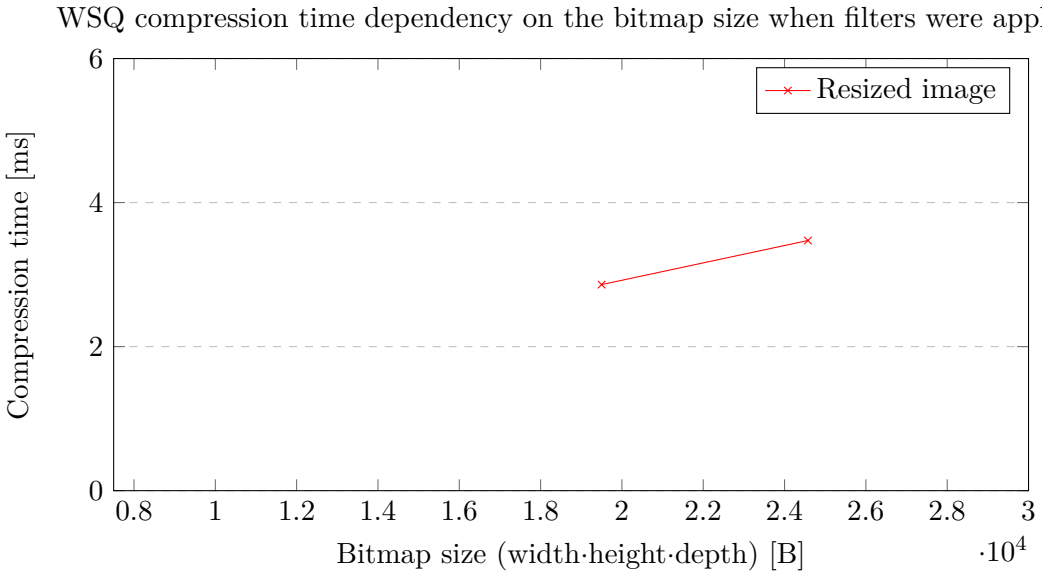


Figure B.14: Plot shows the shortest WSQ compression time for each bitmap size when image was resized to 50% of the original size

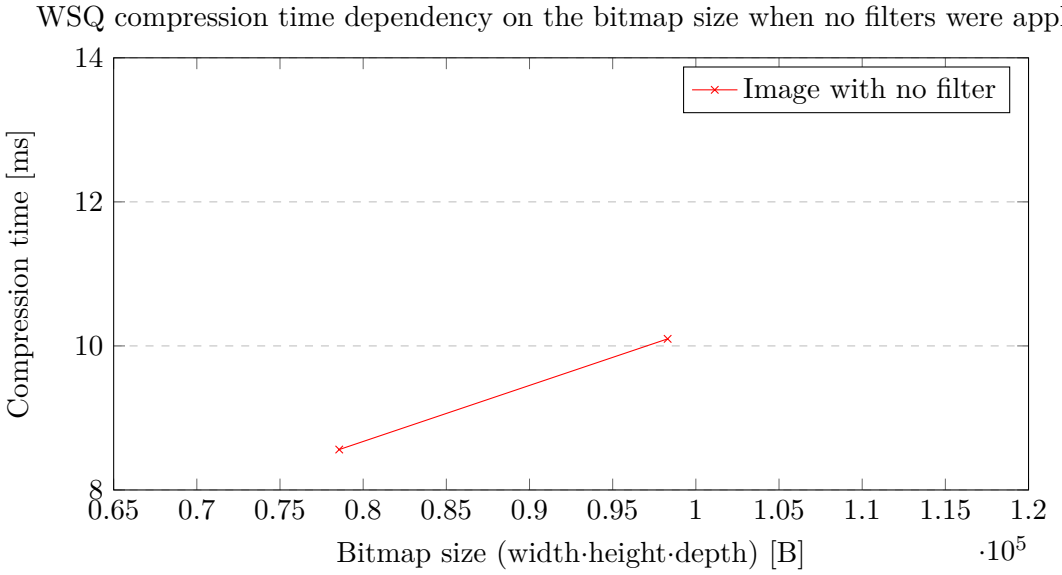


Figure B.15: Plot shows the shortest WSQ compression time for each bitmap size when image was not pre-processed (filters were not applied)

B.5.6 The longest WSQ compression times for bitmap size

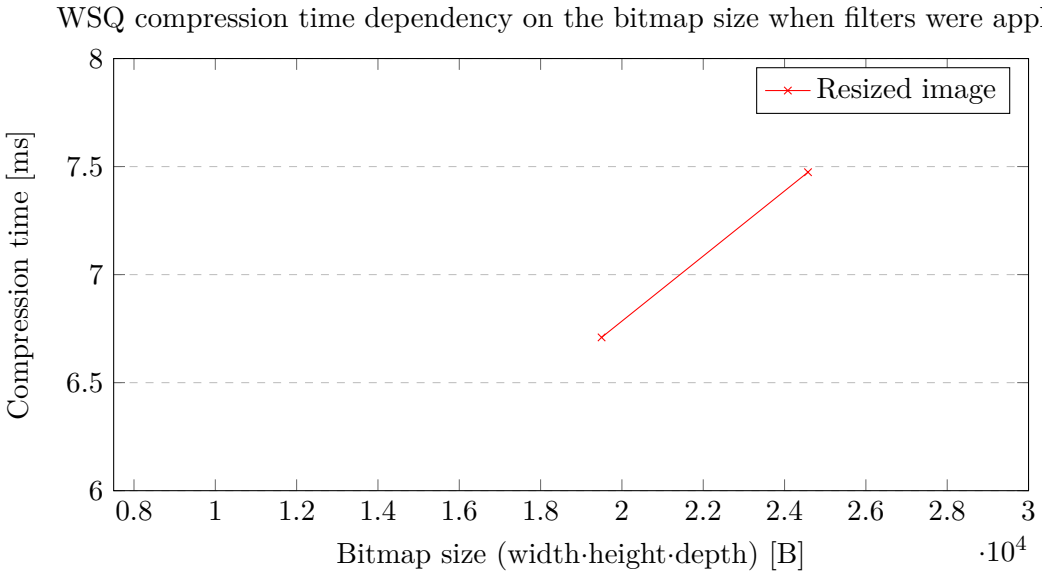


Figure B.16: Plot shows the longest WSQ compression time for each bitmap size when image was resized to 50% of the original size

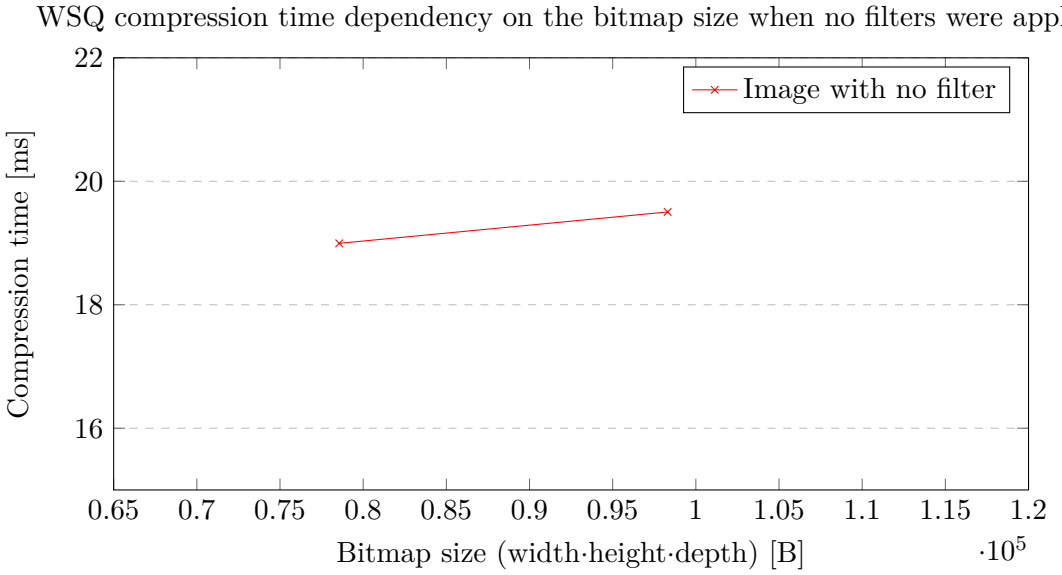


Figure B.17: Plot shows the longest WSQ compression time for each bitmap size when image was not pre-processed (filters were not applied)

Appendix C

Damaged fingerprint results

C.1 Matching scores

C.1.1 The best minutiae-based scores

Ratio	Filters	Score
100	depth changed to 8b	424.75142757322834
100	none	424.75142757322834

Table C.1: The ratios and filters for the best JPEG minutiae-based match

Ratio	Filters	Score
all tested	depth changed to 8b	1179.5927929938084
0	none	1179.5927929938084

Table C.2: The ratios and filters for the best PNG minutiae-based match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score
0.75	depth changed to 8b	310.89555323260913
0.75	none	310.89555323260913

Table C.3: The encodings and filters for the best WSQ minutiae-based match

C.1.2 The worst minutiae-based score

Ratio	Filters	Score
all tested	all tested	0.0
100	none	0.0

Table C.4: The ratios and filters for the worst JPEG minutiae-based match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Ratio	Filters	Score
all tested	all tested	0.0
0	none	0.0

Table C.5: The ratios and filters for the worst PNG minutiae-based match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score
all tested	resized to 50%	0.0
all tested	resized to 125%	0.0
all tested	normalized	0.0
all tested	resized to 80%	0.0
all tested	denoised	0.0
all tested	normalized, denoised	0.0
all tested	resized to 150%	0.0
all tested	depth changed to 8b	0.0
0.75	none	0.0

Table C.6: The encodings and filters for the worst WSQ minutiae-based match. Encoding rates of 0.10, 0.25, 0.50, 0.75, 1.00 are understood by all tested

C.1.3 The best cross-correlation scores

Ratio	Filters	Score [%]
50, 75, 100	normalized	99.0
50, 75, 100	normalized, binarized	99.0
75	none	93.0

Table C.7: The ratios and filters for the best JPEG cross-correlation match

Ratio	Filters	Score [%]
all tested	normalized, binarized	99.0
all tested	normalized	99.0
all tested	depth changed to 8b	99.0
0	none	99.0

Table C.8: The ratios and filters for the best PNG cross-correlation match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score [%]
0.5	depth changed to 8b	98.0
0.5	normalized	98.0
0.5	none	98.0

Table C.9: The encodings and filters for the best WSQ cross-correlation match

C.1.4 The worst cross-correlation scores

Ratio	Filters	Score [%]
all tested	all tested	0.0
100	none	0.0

Table C.10: The ratios and filters for the worst JPEG cross-correlation match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Ratio	Filters	Score [%]
all tested	resized to 50%	0.0
all tested	resized to 125%	0.0
all tested	normalized	0.0
all tested	binarized	0.0
all tested	resized to 80%	0.0
all tested	denoised	0.0
all tested	normalized, denoised, depth changed to 1b	0.0
all tested	normalized, denoised, binarized	0.0
all tested	normalized, binarized	0.0
all tested	denoised, binarized	0.0
all tested	normalized, denoised	0.0
all tested	depth changed to 1b	0.0
all tested	resized to 150%	0.0
0	none	99.0

Table C.11: The ratios and filters for the worst PNG cross-correlation match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score [%]
all tested	resized to 50%	0.0
all tested	resized to 125%	0.0
all tested	normalized	0.0
all tested	resized to 80%	0.0
all tested	denoised	0.0
all tested	normalized, denoised	0.0
all tested	resized to 150%	0.0
all tested	depth changed to 8b	0.0
0.75	none	0.0

Table C.12: The encodings and filters for the worst WSQ cross-correlation match. Encoding rates of 0.10, 0.25, 0.50, 0.75, 1.00 are understood by all tested

C.1.5 The best PSNR scores

Ratio	Filters	Score [dB]
100	depth changed to 1b	21.768398233078692
0	none	19.57734247693712

Table C.13: The ratios and filters for the best JPEG PSNR match

Encoding	Filters	Score [dB]
1.0	depth changed to 8b	25.114499414989005
1.0	none	25.114499414989005

Table C.14: The encodings and filters for the best WSQ PSNR match

C.1.6 The worst PSNR scores

Ratio	Filters	Score [dB]
50	depth changed to 8b	6.29325268254798
50	none	6.29325268254798

Table C.15: The ratios and filters for the worst JPEG PSNR match

Encoding	Filters	Score [dB]
0.1	normalized, denoised	6.9089682612922605
0.1	none	8.874867746514719

Table C.16: The encodings and filters for the worst WSQ PSNR match

C.2 Compression times

C.2.1 The shortest compression times

Ratio	Filters	Time [ms]
0	resized to 50%	0.416831
0	none	1.159169

Table C.17: The ratios and filters for the shortest JPEG compression time

Ratio	Filters	Time [ms]
100	resized to 50%	0.294821
100	none	0.750107

Table C.18: The ratios and filters for the shortest PNG compression time

Encoding	Filters	Time [ms]
0.1	resized to 50%	1.679653
0.1	none	4.369841

Table C.19: The encodings and filters for the shortest WSQ compression time

C.2.2 The longest compression times

Ratio	Filters	Time [ms]
100	resized to 150%	21.988825
100	none	10.019492

Table C.20: The ratios and filters for the longest JPEG compression time

Ratio	Filters	Time [ms]
0	resized to 150%	163.286202
0	none	79.225599

Table C.21: The ratios and filters for the longest PNG compression time

Encoding	Filters	Time [ms]
1.0	resized to 150%	89.209265
1.0	none	36.218393

Table C.22: The encodings and filters for the longest WSQ compression time

C.3 Compressed file sizes

C.3.1 The smallest file sizes

Ratio	Filters	Size [B]
0	resized to 50%	1078
0	none	2116

Table C.23: The ratios and filters for the smallest JPEG file size

Ratio	Filters	Size [B]
0	depth changed to 1b	725
0	none	7697

Table C.24: The ratios and filters for the smallest PNG file size

Encoding	Filters	Size [B]
0.1	resized to 50%	710
0.1	none	1065

Table C.25: The encodings and filters for the smallest WSQ file size

C.3.2 The biggest file sizes

Ratio	Filters	Size [B]
100	resized to 150%	306716
100	none	150920

Table C.26: The ratios and filters for the biggest JPEG file size

Ratio	Filters	Size [B]
100	normalized, denoised	519907
100	denoised	519907
100	none	173625

Table C.27: The ratios and filters for the biggest PNG file size

Encoding	Filters	Size [B]
1.0	resized to 150%	27395
1.0	none	12460

Table C.28: The encodings and filters for the biggest WSQ file size

C.4 The average values

C.4.1 The average minutiae-based scores

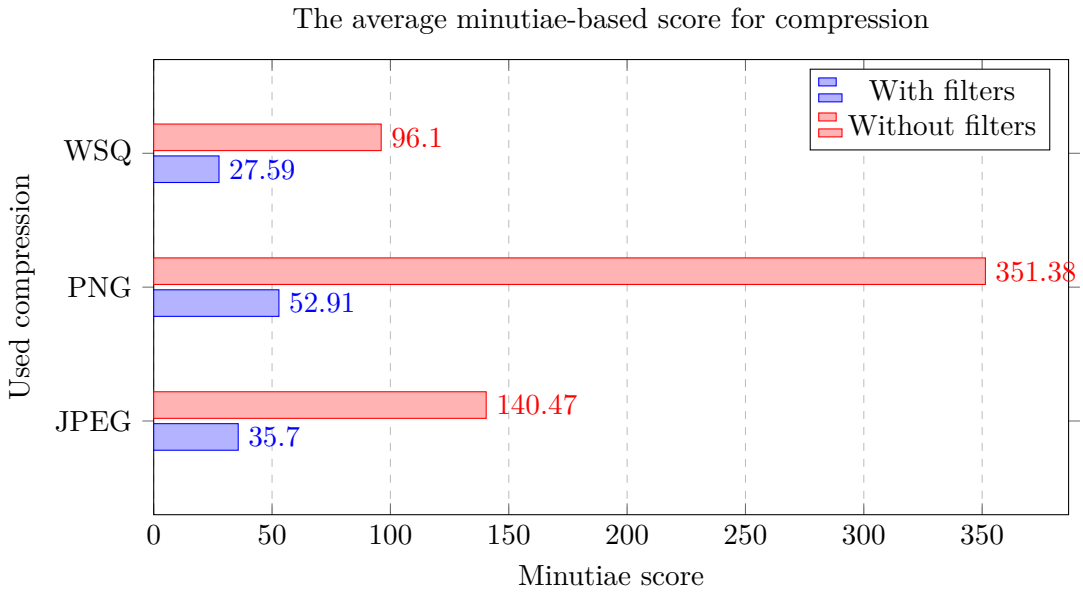


Figure C.1: The average minutiae-based score for compression. Score more than 40 means the fingerprints matched.

C.4.2 The average cross-correlation scores

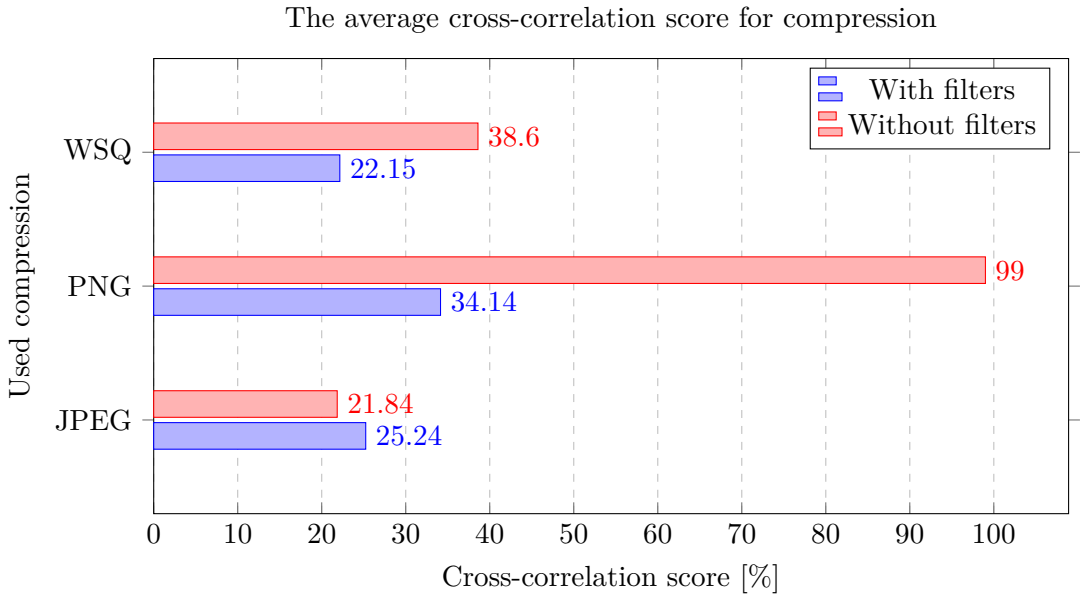


Figure C.2: The average cross-correlation score for compression. Score expresses percentage match between fingerprints.

C.4.3 The average PSNR scores

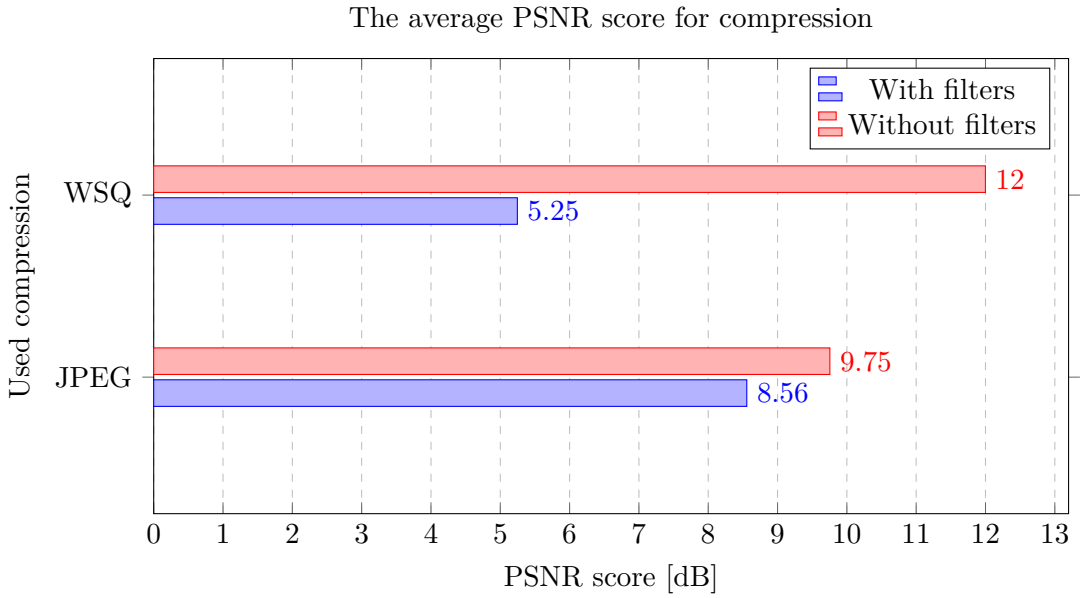


Figure C.3: The average PSNR score for compression. Score more than 40 means the fingerprints matched.

C.4.4 The average compression times

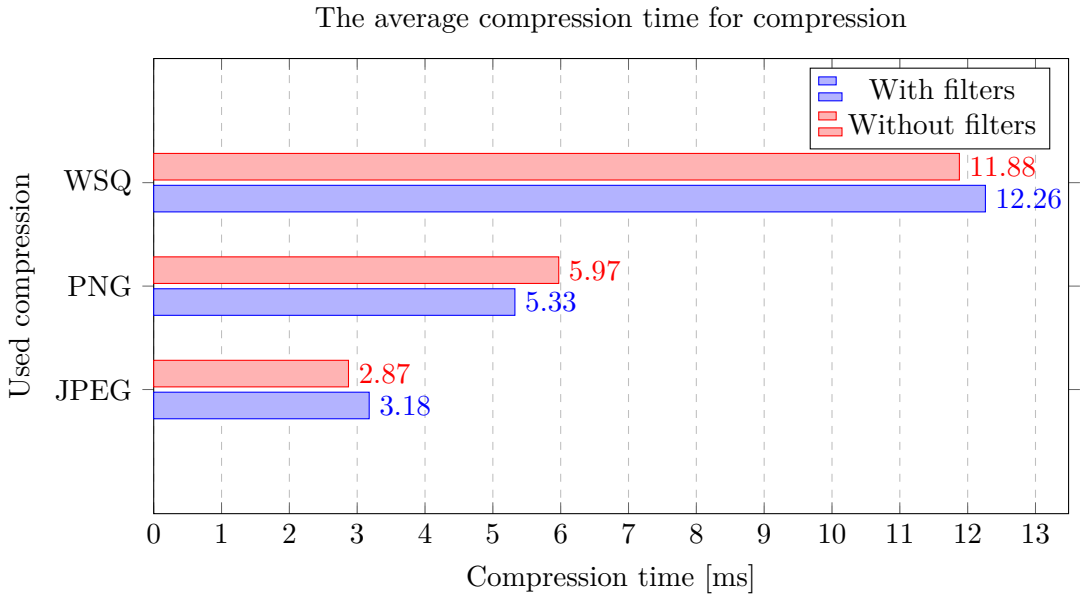


Figure C.4: The average compression time for compression in milliseconds.

C.4.5 The average compressed file sizes

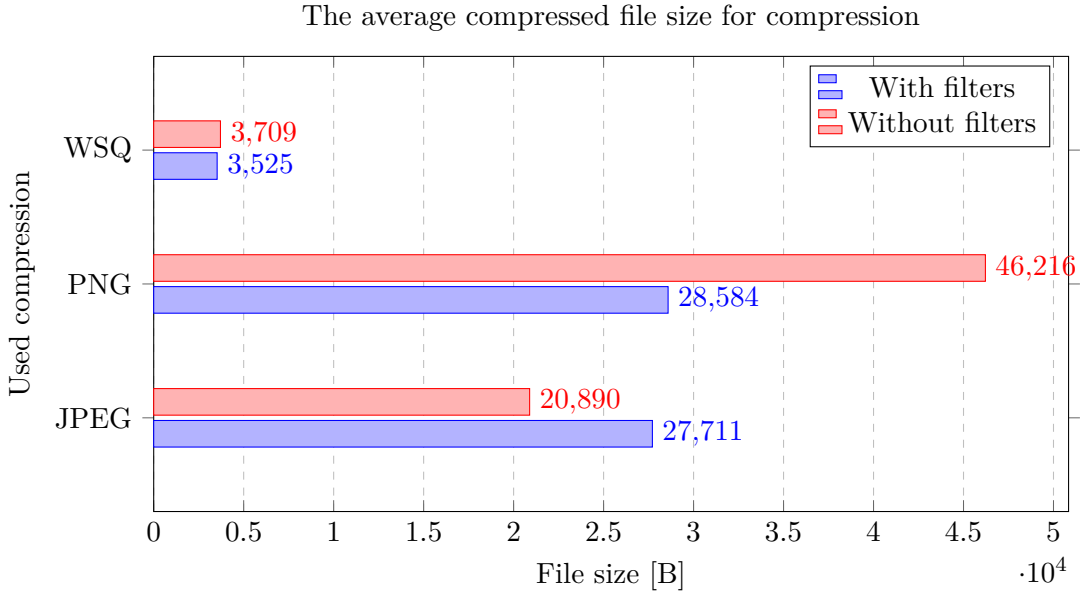


Figure C.5: The average compressed file size for compression in Bytes

C.5 Compression time dependency on the bitmap size

C.5.1 The fastest JPEG compression times for bitmap size

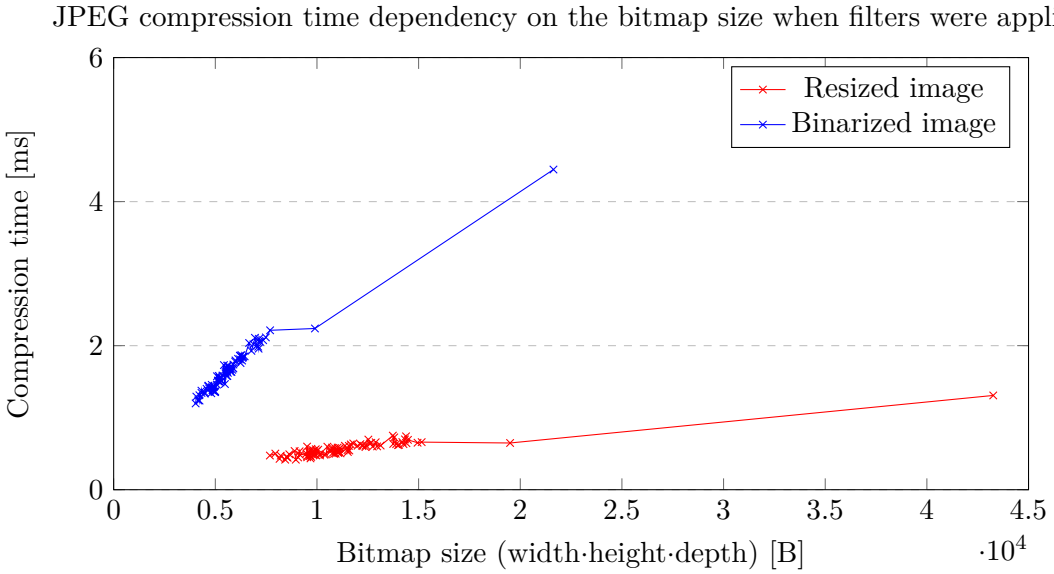


Figure C.6: Plot shows the fastest JPEG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

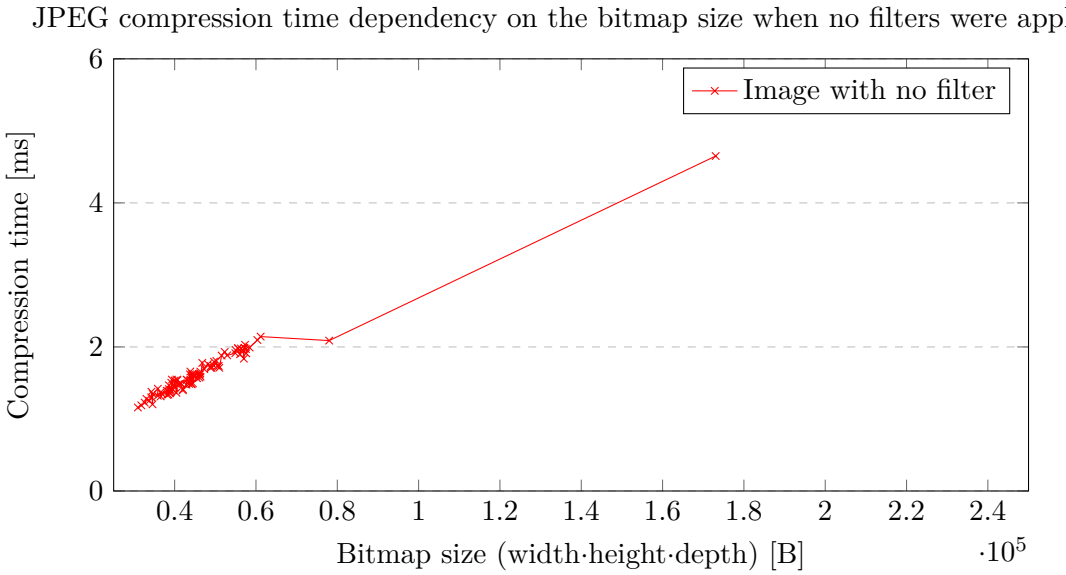


Figure C.7: Plot shows the shortest JPEG compression time for each bitmap size when image was not pre-processed (filters were not applied)

C.5.2 The longest JPEG compression times for bitmap size

JPEG compression time dependency on the bitmap size when filters were applied

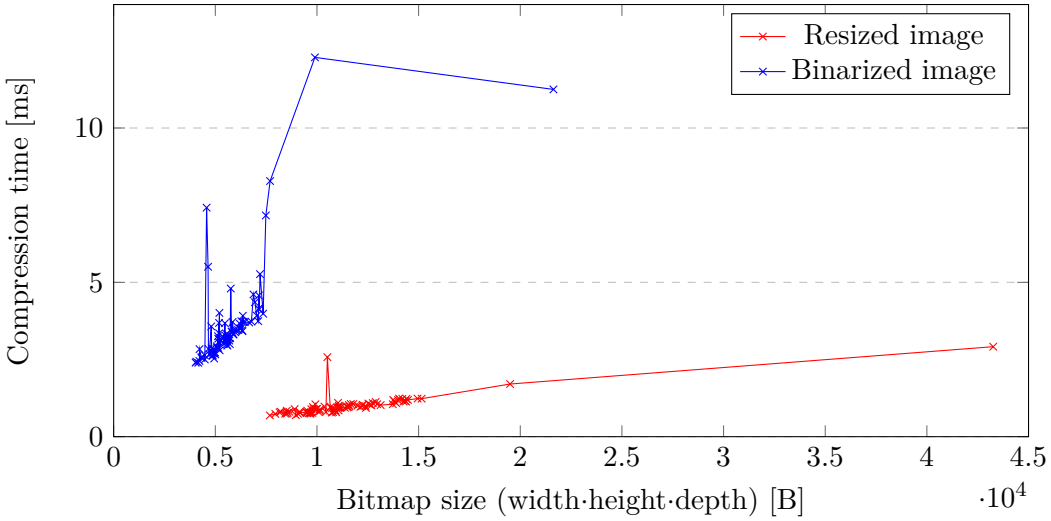


Figure C.8: Plot shows the longest JPEG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

JPEG compression time dependency on the bitmap size when no filters were applied

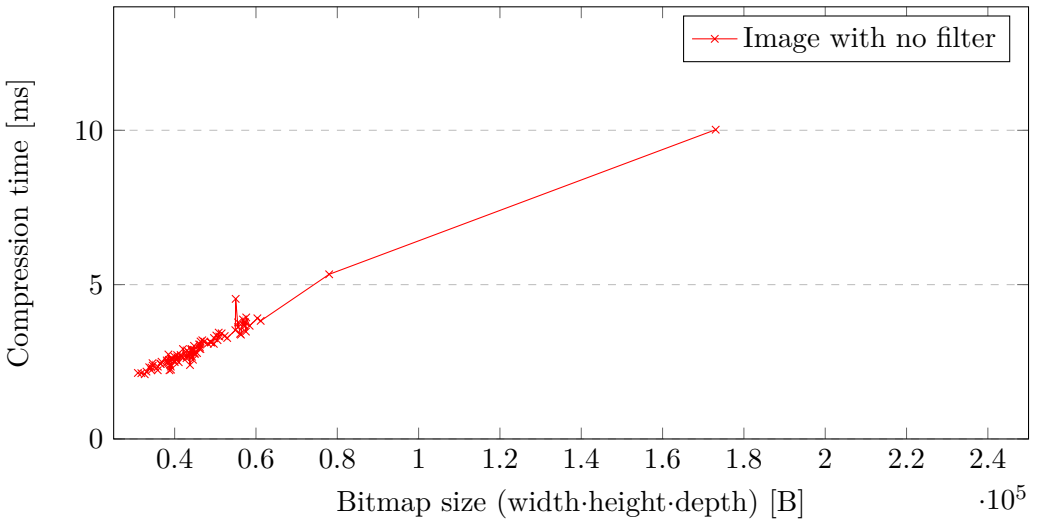


Figure C.9: Plot shows the longest JPEG compression time for each bitmap size when image was not pre-processed (filters were not applied)

C.5.3 The fastest PNG compression times for bitmap size

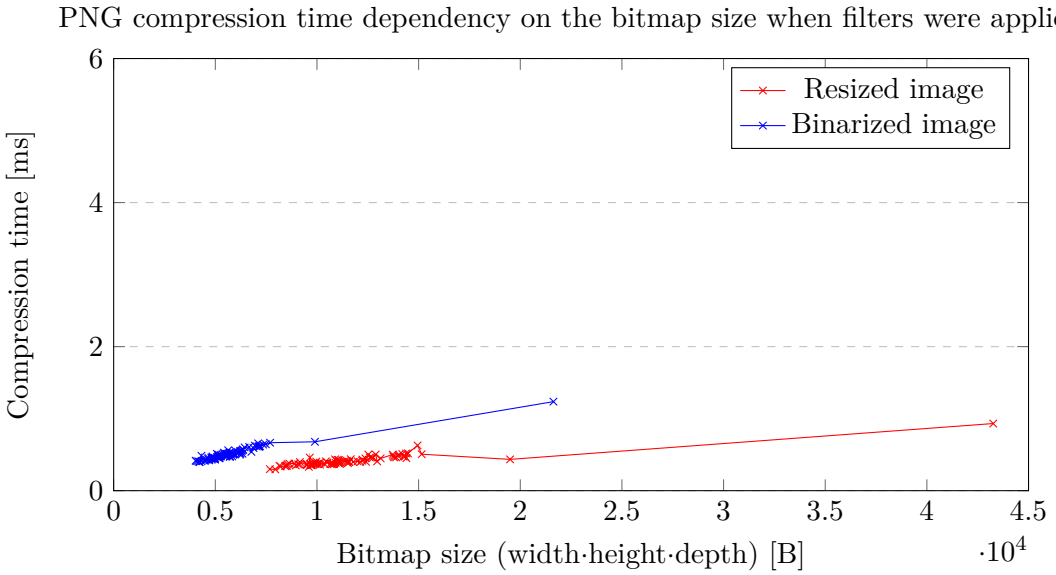


Figure C.10: Plot shows the shortest PNG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

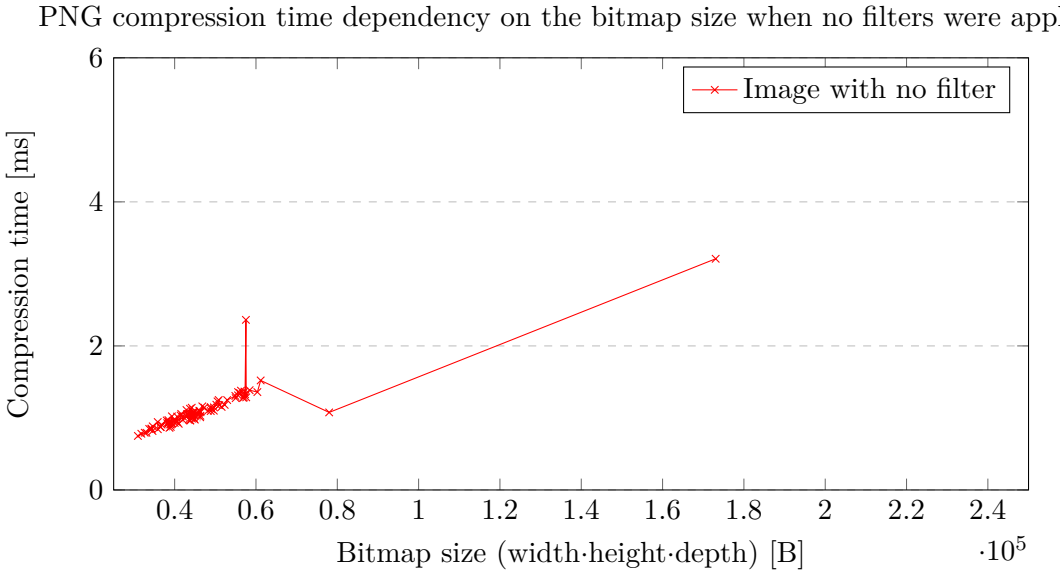


Figure C.11: Plot shows the shortest PNG compression time for each bitmap size when image was not pre-processed (filters were not applied)

C.5.4 The longest PNG compression times for bitmap size

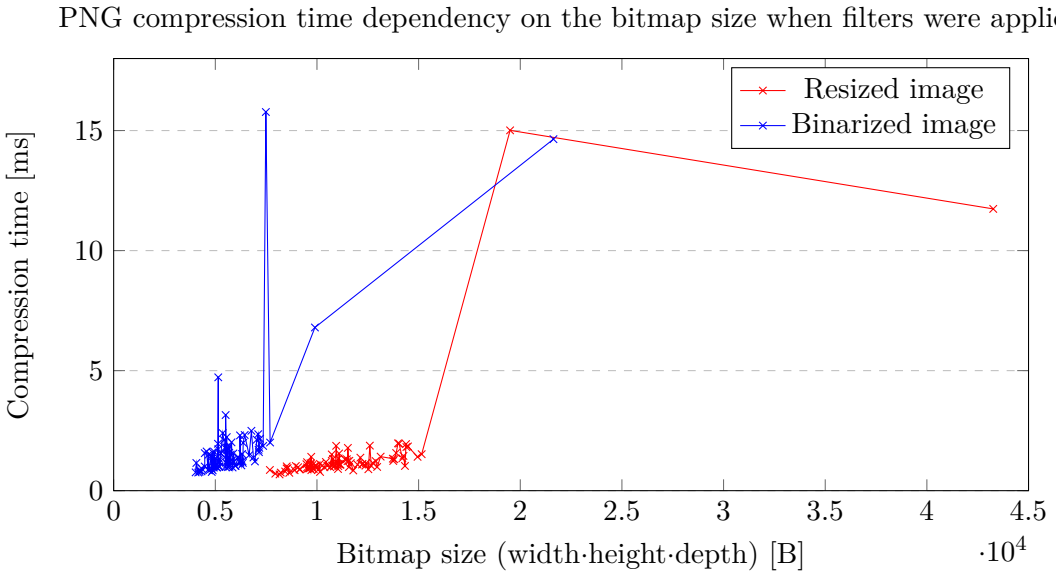


Figure C.12: Plot shows the longest PNG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

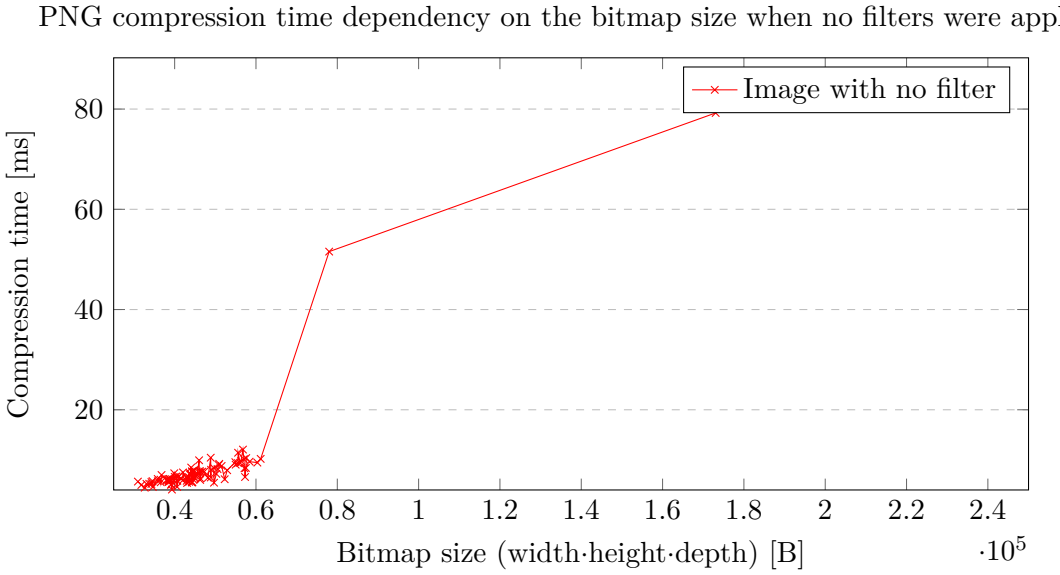


Figure C.13: Plot shows the longest PNG compression time for each bitmap size when image was not pre-processed (filters were not applied)

C.5.5 The fastest WSQ compression times for bitmap size

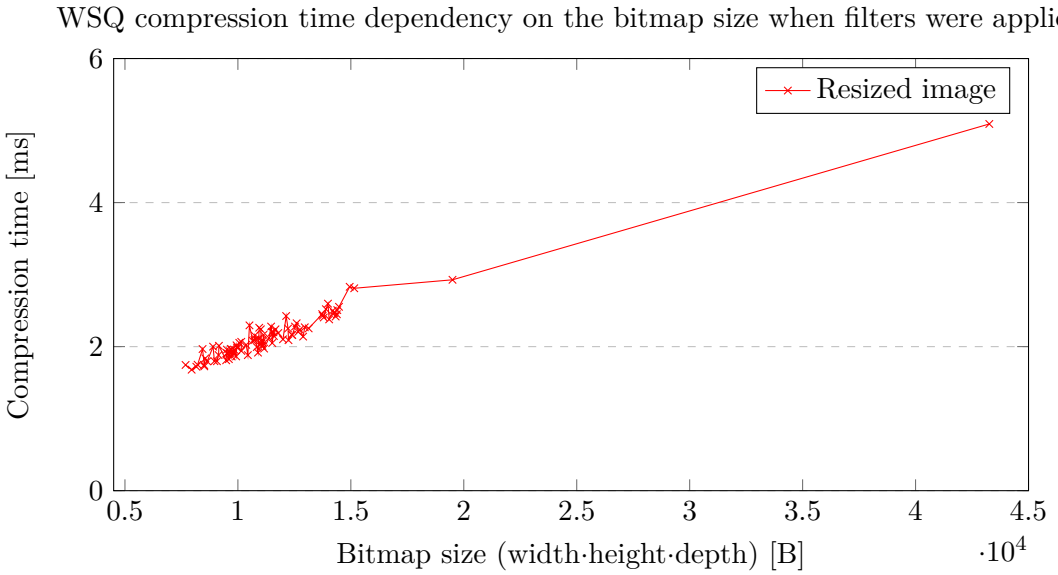


Figure C.14: Plot shows the shortest WSQ compression time for each bitmap size when image was resized to 50% of the original size

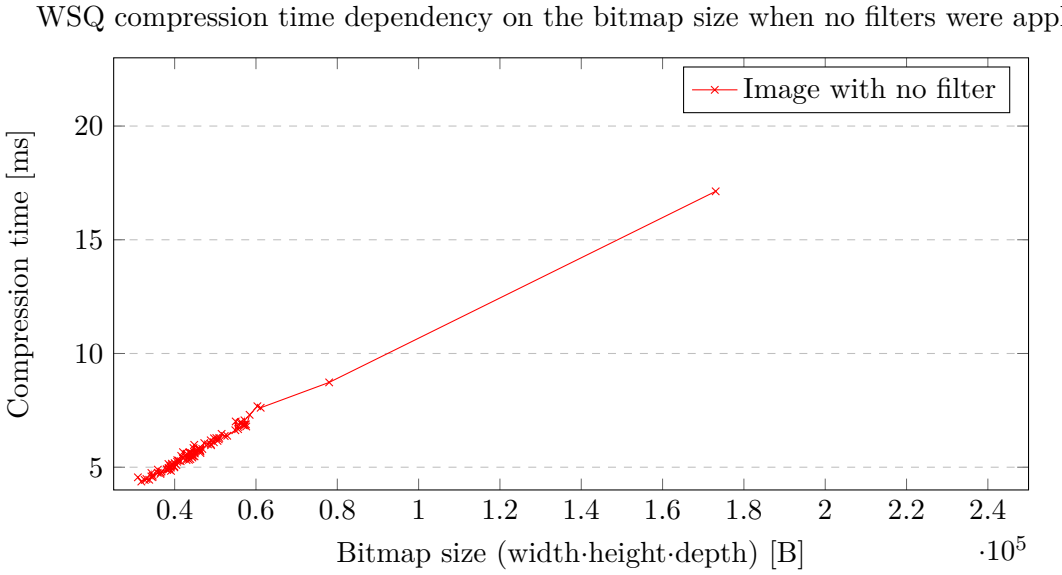


Figure C.15: Plot shows the shortest WSQ compression time for each bitmap size when image was not pre-processed (filters were not applied)

C.5.6 The longest WSQ compression times for bitmap size

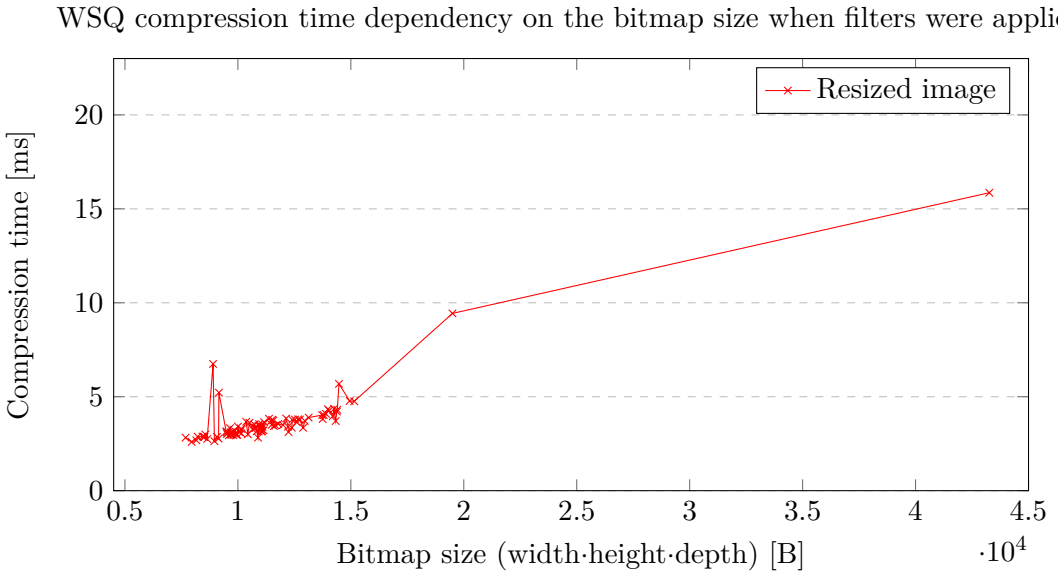


Figure C.16: Plot shows the longest WSQ compression time for each bitmap size when image was resized to 50% of the original size

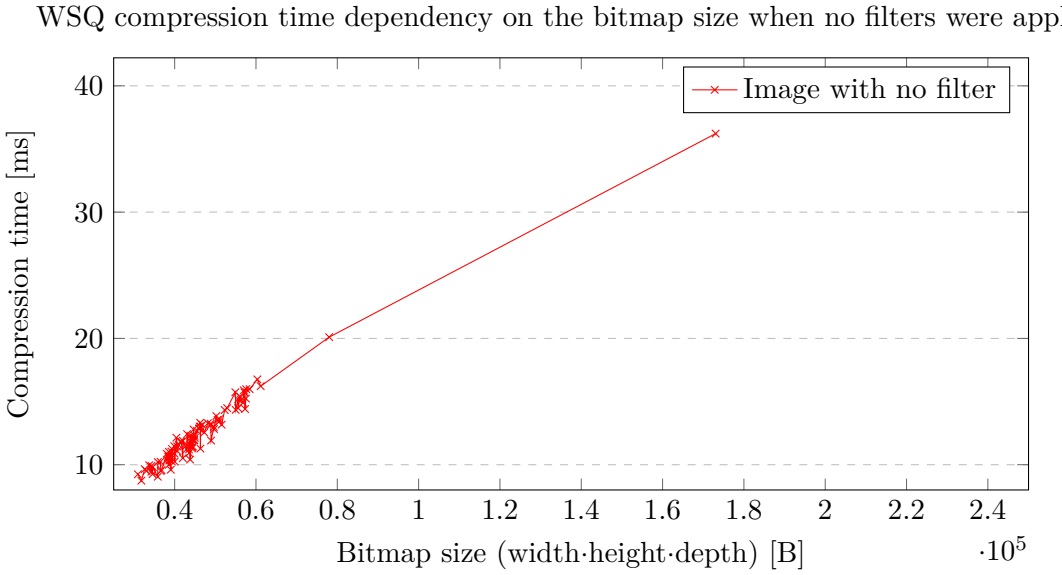


Figure C.17: Plot shows the longest WSQ compression time for each bitmap size when image was not pre-processed (filters were not applied)

Appendix D

Real fingerprint results

D.1 Matching scores

D.1.1 The best minutiae-based scores

Ratio	Filters	Score
100	depth changed to 8b	724.8772397674417
100	none	724.8772397674417

Table D.1: The ratios and filters for the best JPEG minutiae-based match

Ratio	Filters	Score
all tested	depth changed to 8b	1179.5927929938084
0	none	1179.5927929938084

Table D.2: The ratios and filters for the best PNG minutiae-based match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score
0.75	depth changed to 8b	649.7977894701282
0.75	none	649.7977894701282

Table D.3: The encodings and filters for the best WSQ minutiae-based match

D.1.2 The worst minutiae-based score

Ratio	Filters	Score
all tested	normalized, denoised, depth changed to 1b	0.0
all tested	resized to 150%	0.0
all tested	denoised, binarized	0.0
0, 25, 50, 75	normalized, binarized	0.0
all tested	denoised	0.0
0, 75, 100	resized to 125%	0.0
all tested	resized to 50%	0.0
0, 75	normalized	0.0
0	resized to 80%	0.0
all tested	depth changed to 1b	0.0
all tested	normalized, denoised	0.0
all tested	normalized, denoised, binarized	0.0
0	depth changed to 8b	0.0
0	none	0.0

Table D.4: The ratios and filters for the worst JPEG minutiae-based match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Ratio	Filters	Score
all tested	normalized, denoised, binarized	0.0
all tested	resized to 50%	0.0
all tested	normalized	0.0
all tested	denoised, binarized	0.0
all tested	binarized	0.0
all tested	resized to 80%	0.0
all tested	denoised	0.0
all tested	normalized, denoised	0.0
all tested	depth changed to 1b	0.0
all tested	resized to 150%	0.0
all tested	normalized, denoised, depth changed to 1b	0.0
0	none	141.9909075695878

Table D.5: The ratios and filters for the worst PNG minutiae-based match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score
all tested	resized to 150%	0.0
all tested	resized to 50%	0.0
0.1, 0.25, 0.75, 1.0	resized to 80%	0.0
all tested	normalized, denoised	0.0
all tested	denoised	0.0
0.1, 0.5	resized to 125%	0.0
0.1, 0.75	normalized	0.0
0.1	none	1.7512533241406483

Table D.6: The encodings and filters for the worst WSQ minutiae-based match. Encoding rates of 0.10, 0.25, 0.50, 0.75, 1.00 are understood by all tested

D.1.3 The best cross-correlation scores

Ratio	Filters	Score [%]
0	denoised	99.0
50, 75, 100	denoised, binarized	99.0
25	normalized, denoised	99.0
75	resized to 80%	99.0
25	none	96.0

Table D.7: The ratios and filters for the best JPEG cross-correlation match

Ratio	Filters	Score [%]
all tested	depth changed to 8b	99.0
all tested	resized to 150%	99.0
all tested	resized to 125%	99.0
0, 25, 75	denoised, binarized	99.0
0	normalized, denoised, depth changed to 1b	99.0
0	none	99.0

Table D.8: The ratios and filters for the best PNG cross-correlation match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score [%]
0.25	depth changed to 8b	99.0
0.5, 1.0	resized to 150%	99.0
0.25, 0.75	resized to 125%	99.0
0.1	normalized	99.0
0.25	none	99.0

Table D.9: The encodings and filters for the best WSQ cross-correlation match

D.1.4 The worst cross-correlation scores

Ratio	Filters	Score [%]
all tested	all tested	0.0
100	none	0.0

Table D.10: The ratios and filters for the worst JPEG cross-correlation match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Ratio	Filters	Score [%]
all tested	all tested	0.0
0	none	0.0

Table D.11: The ratios and filters for the worst PNG cross-correlation match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score [%]
all tested	resized to 50%	0.0
all tested	resized to 125%	0.0
all tested	normalized	0.0
all tested	resized to 80%	0.0
all tested	denoised	0.0
all tested	normalized, denoised	0.0
all tested	resized to 150%	0.0
all tested	depth changed to 8b	0.0
0.75	none	0.0

Table D.12: The encodings and filters for the worst WSQ cross-correlation match. Encoding rates of 0.10, 0.25, 0.50, 0.75, 1.00 are understood by all tested

D.1.5 The best PSNR scores

Ratio	Filters	Score [dB]
75	depth changed to 1b	24.406654820985278
100	none	21.303054075369026

Table D.13: The ratios and filters for the best JPEG PSNR match

Encoding	Filters	Score [dB]
1.0	depth changed to 8b	35.243883201233686
1.0	none	35.243883201233686

Table D.14: The encodings and filters for the best WSQ PSNR match

D.1.6 The worst PSNR scores

Ratio	Filters	Score [dB]
100	depth changed to 8b	5.3960317606377455
100	none	5.3960317606377455

Table D.15: The ratios and filters for the worst JPEG PSNR match

Encoding	Filters	Score [dB]
1.0	normalized, denoised	8.068990618862403
0.1	none	9.126309368455237

Table D.16: The encodings and filters for the worst WSQ PSNR match

D.2 Compression times

D.2.1 The shortest compression times

Ratio	Filters	Time [ms]
0	resized to 50%	0.752461
0	none	2.477509

Table D.17: The ratios and filters for the shortest JPEG compression time

Ratio	Filters	Time [ms]
100	resized to 50%	0.54065
100	none	1.733561

Table D.18: The ratios and filters for the shortest PNG compression time

Encoding	Filters	Time [ms]
0.1	resized to 50%	2.782313
0.1	none	8.778466

Table D.19: The encodings and filters for the shortest WSQ compression time

D.2.2 The longest compression times

Ratio	Filters	Time [ms]
100	resized to 150%	37.500664
0	none	13.584782

Table D.20: The ratios and filters for the longest JPEG compression time

Ratio	Filters	Time [ms]
0	normalized, denoised	208.612401
0	none	80.927067

Table D.21: The ratios and filters for the longest PNG compression time

Encoding	Filters	Time [ms]
1.0	resized to 150%	135.700093
1.0	none	99.564061

Table D.22: The encodings and filters for the longest WSQ compression time

D.3 Compressed file sizes

D.3.1 The smallest file sizes

Ratio	Filters	Size [B]
0	resized to 50%	1042
0	none	2145

Table D.23: The ratios and filters for the smallest JPEG file size

Ratio	Filters	Size [B]
0, 25	depth changed to 1b	119
0	none	41783

Table D.24: The ratios and filters for the smallest PNG file size

Encoding	Filters	Size [B]
0.1	resized to 50%	764
0.1	none	1285

Table D.25: The encodings and filters for the smallest WSQ file size

D.3.2 The biggest file sizes

Ratio	Filters	Size [B]
100	resized to 150%	343392
100	none	166582

Table D.26: The ratios and filters for the biggest JPEG file size

Ratio	Filters	Size [B]
100	normalized, denoised	519907
100	denoised	519907
100	none	173625

Table D.27: The ratios and filters for the biggest PNG file size

Encoding	Filters	Size [B]
1.0	resized to 150%	28779
1.0	none	13874

Table D.28: The encodings and filters for the biggest WSQ file size

D.4 The average values

D.4.1 The average minutiae-based scores

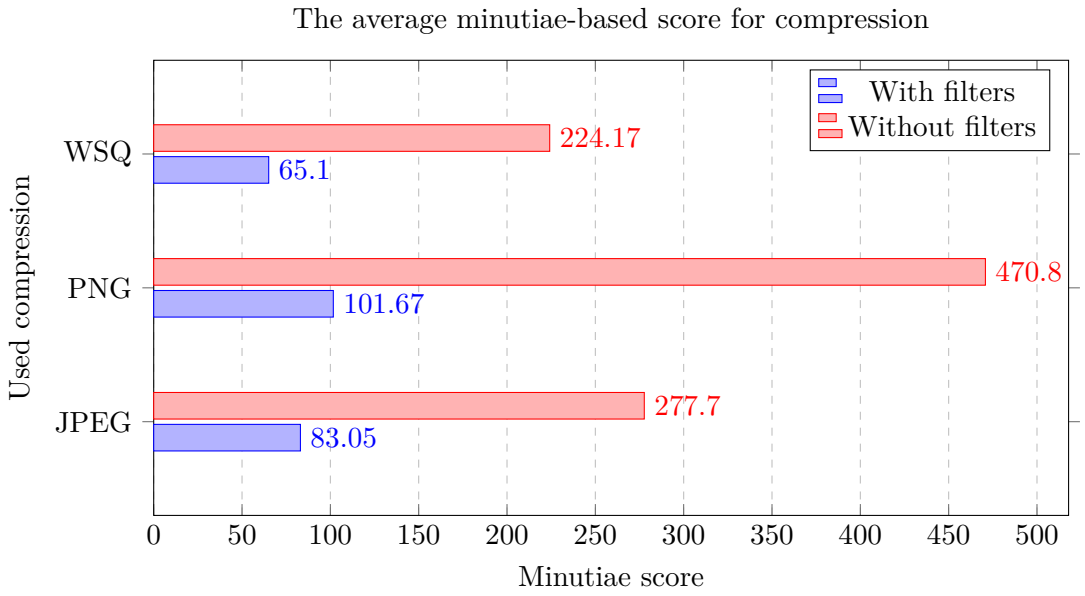


Figure D.1: The average minutiae-based score for compression. Score more than 40 means the fingerprints matched.

D.4.2 The average cross-correlation scores

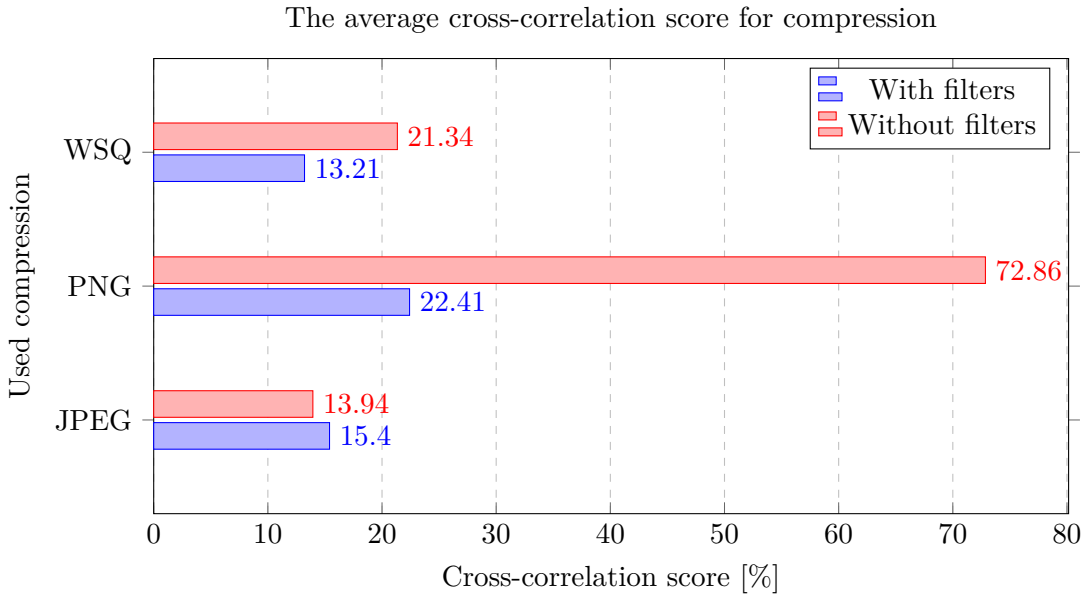


Figure D.2: The average cross-correlation score for compression. Score expresses percentage match between fingerprints.

D.4.3 The average PSNR scores

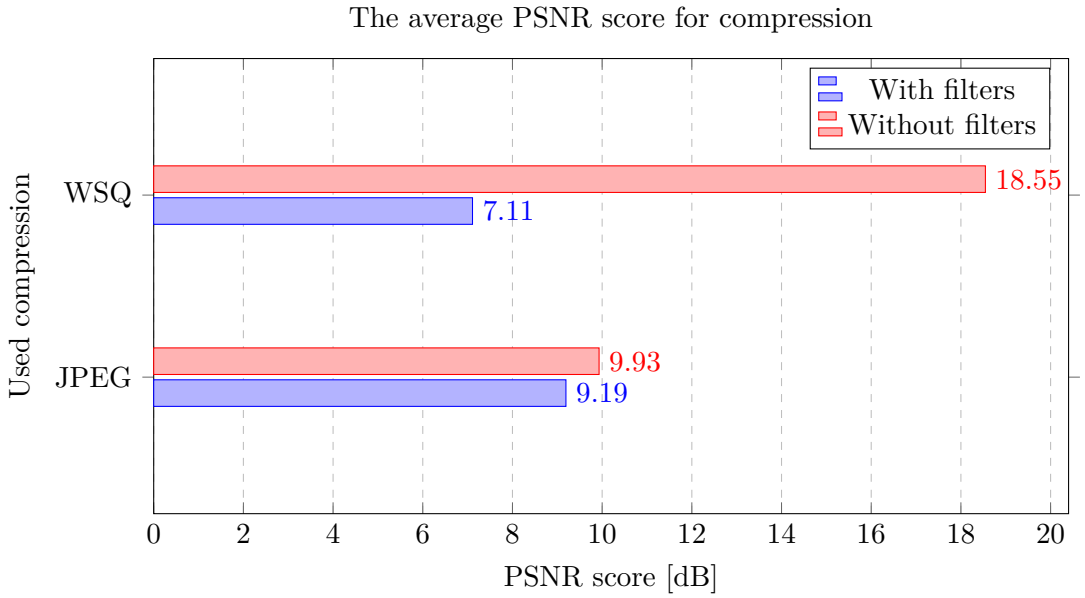


Figure D.3: The average PSNR score for compression. Score more than 40 means the fingerprints matched.

D.4.4 The average compression times

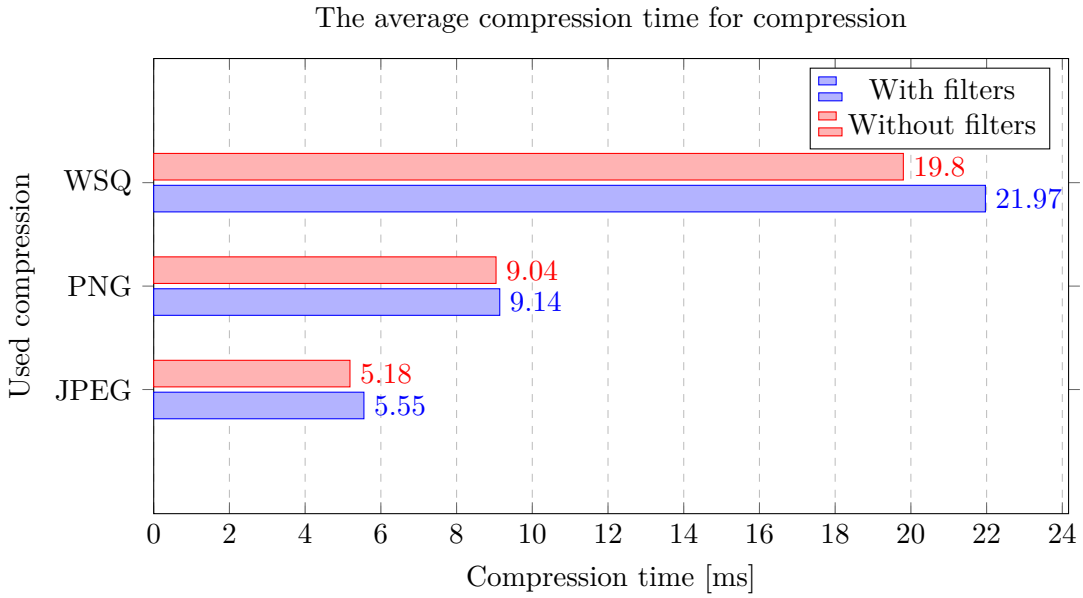


Figure D.4: The average compression time for compression in milliseconds.

D.4.5 The average compressed file sizes

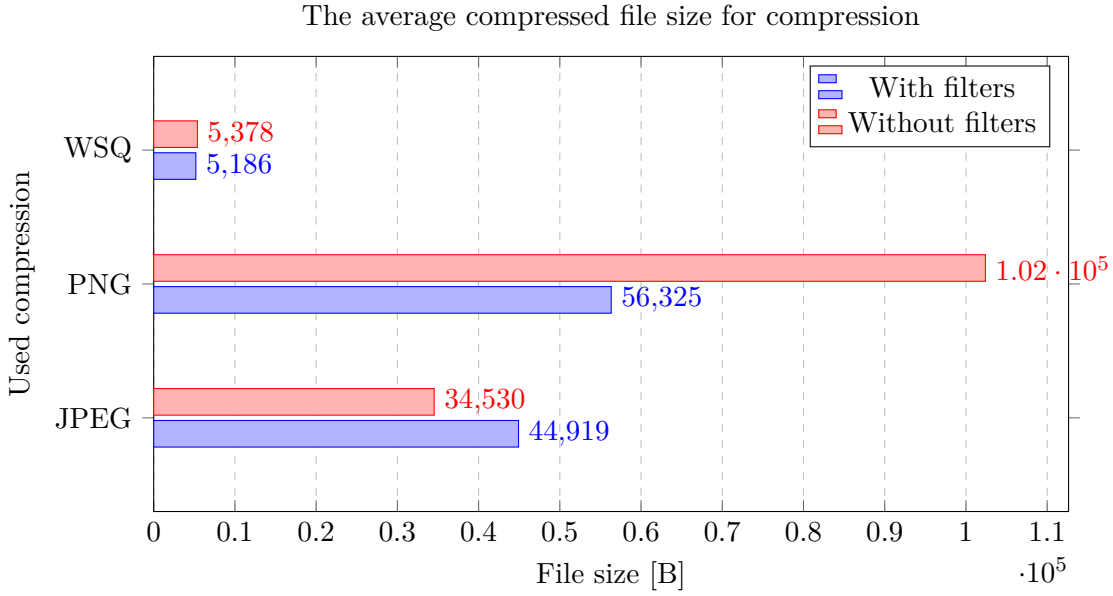


Figure D.5: The average compressed file size for compression in Bytes

D.5 Compression time dependency on the bitmap size

D.5.1 The fastest JPEG compression times for bitmap size

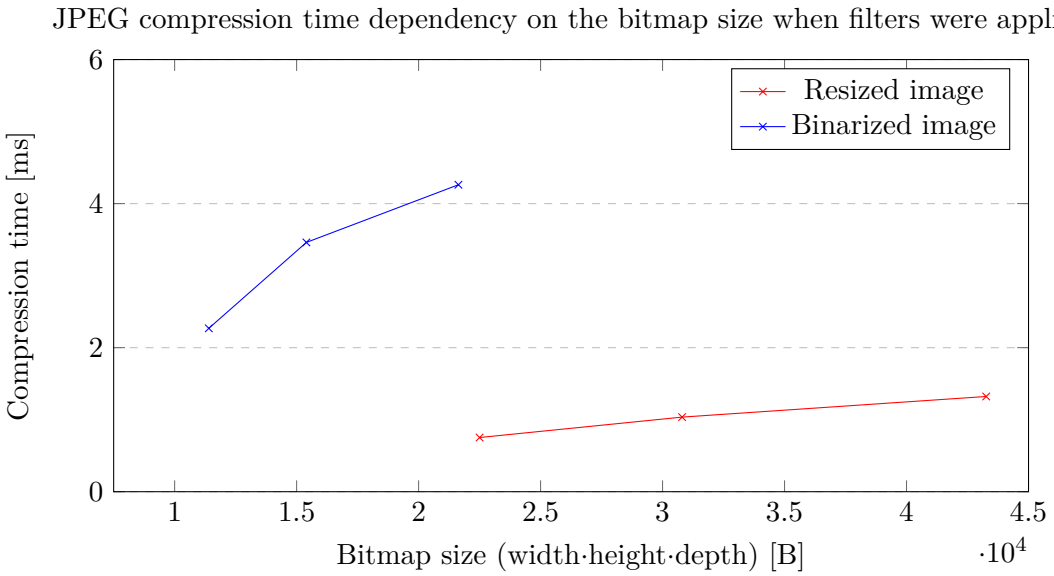


Figure D.6: Plot shows the fastest JPEG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

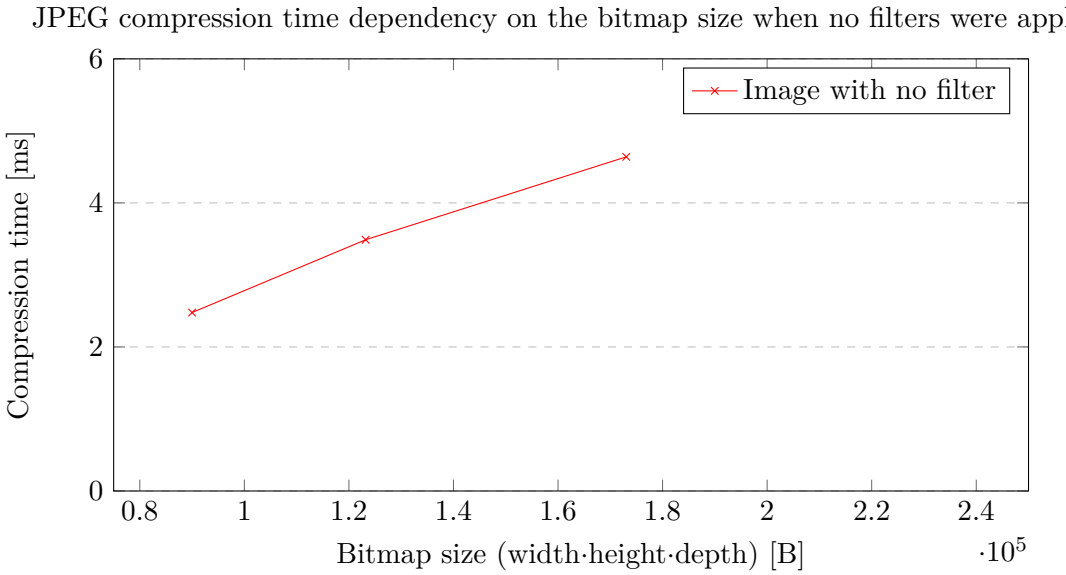


Figure D.7: Plot shows the shortest JPEG compression time for each bitmap size when image was not pre-processed (filters were not applied)

D.5.2 The longest JPEG compression times for bitmap size

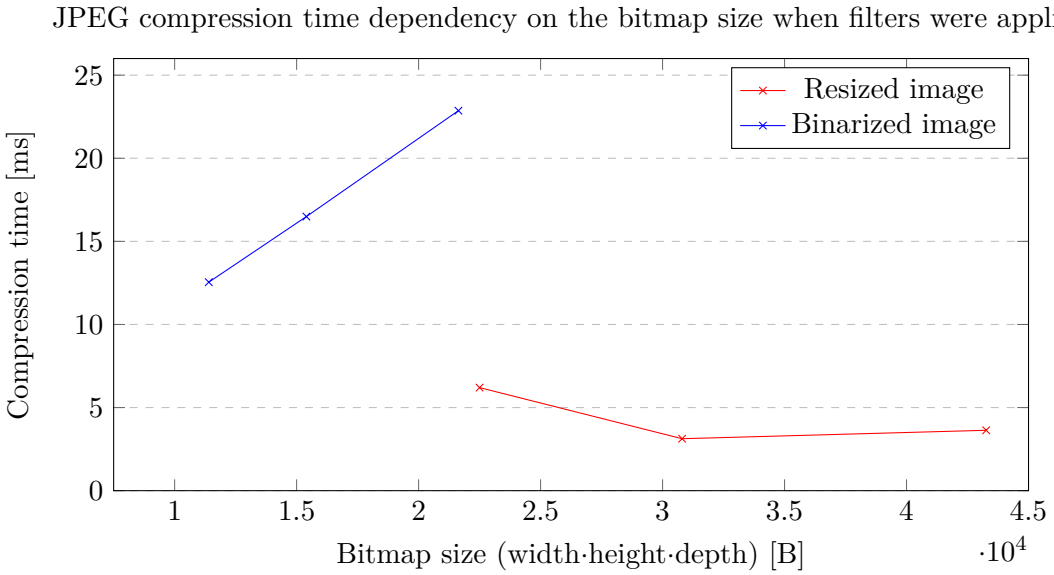


Figure D.8: Plot shows the longest JPEG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

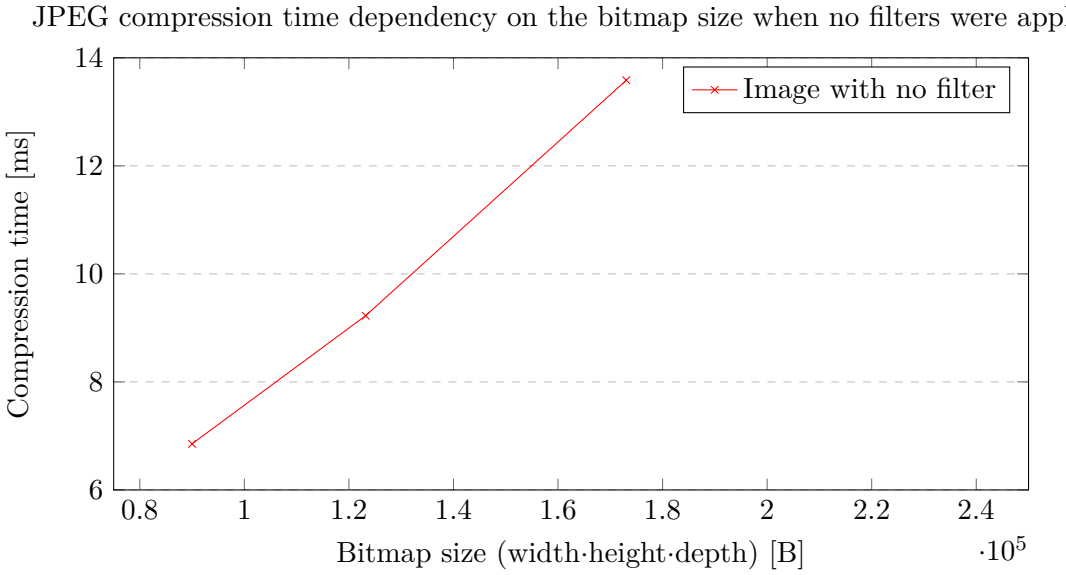


Figure D.9: Plot shows the longest JPEG compression time for each bitmap size when image was not pre-processed (filters were not applied)

D.5.3 The fastest PNG compression times for bitmap size

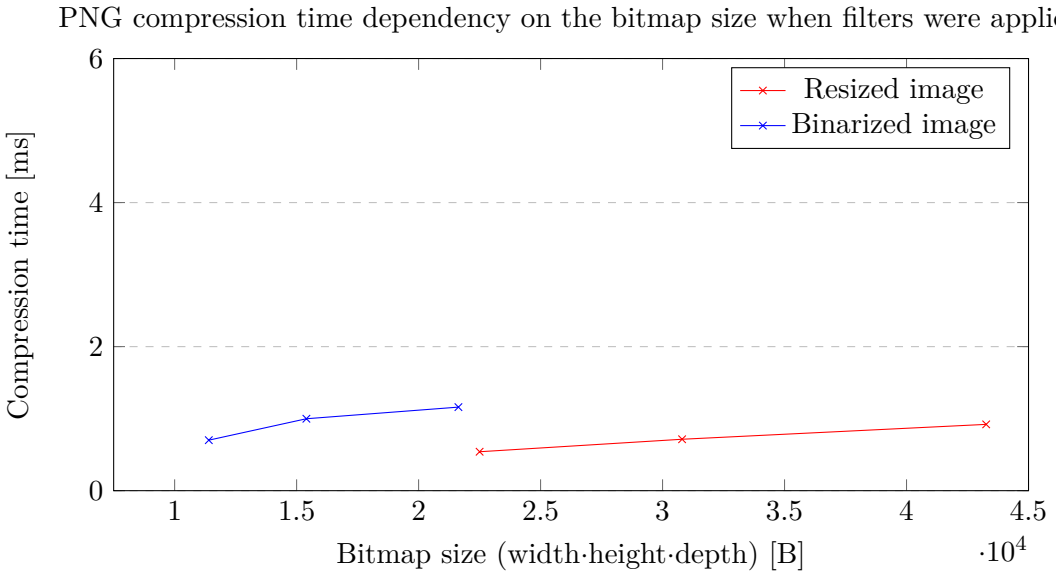


Figure D.10: Plot shows the shortest PNG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

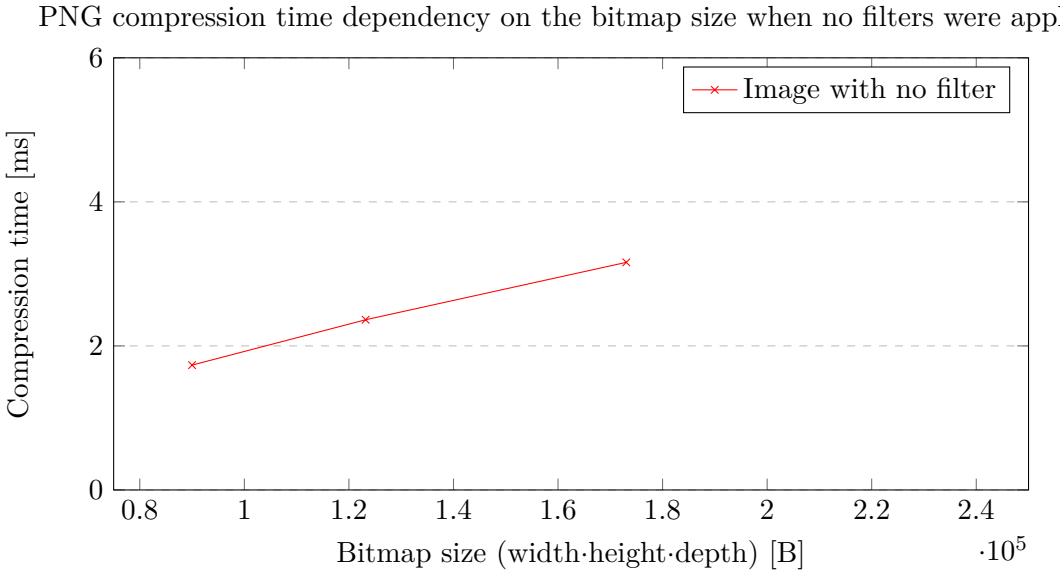


Figure D.11: Plot shows the shortest PNG compression time for each bitmap size when image was not pre-processed (filters were not applied)

D.5.4 The longest PNG compression times for bitmap size

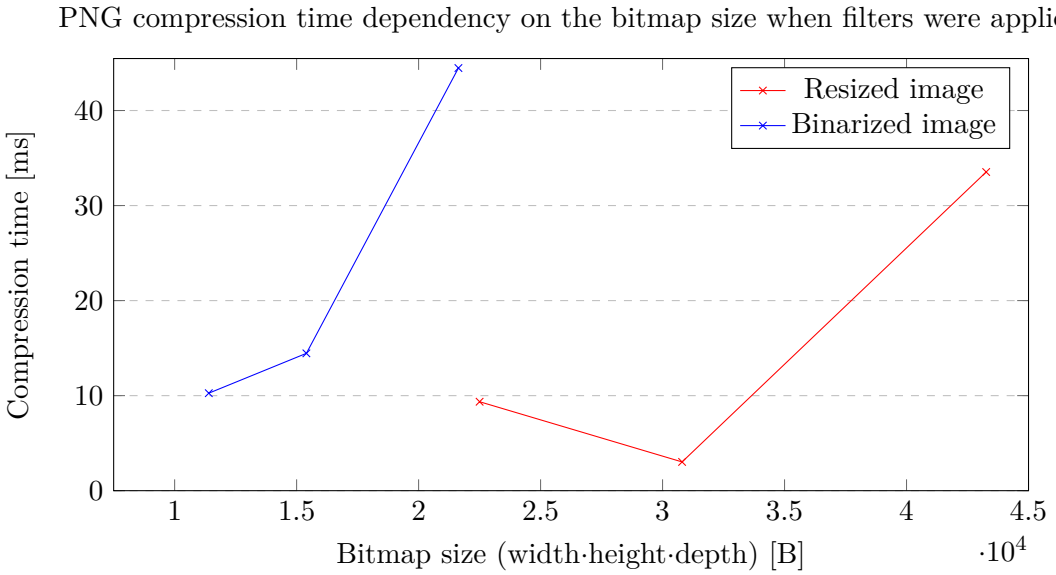


Figure D.12: Plot shows the longest PNG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

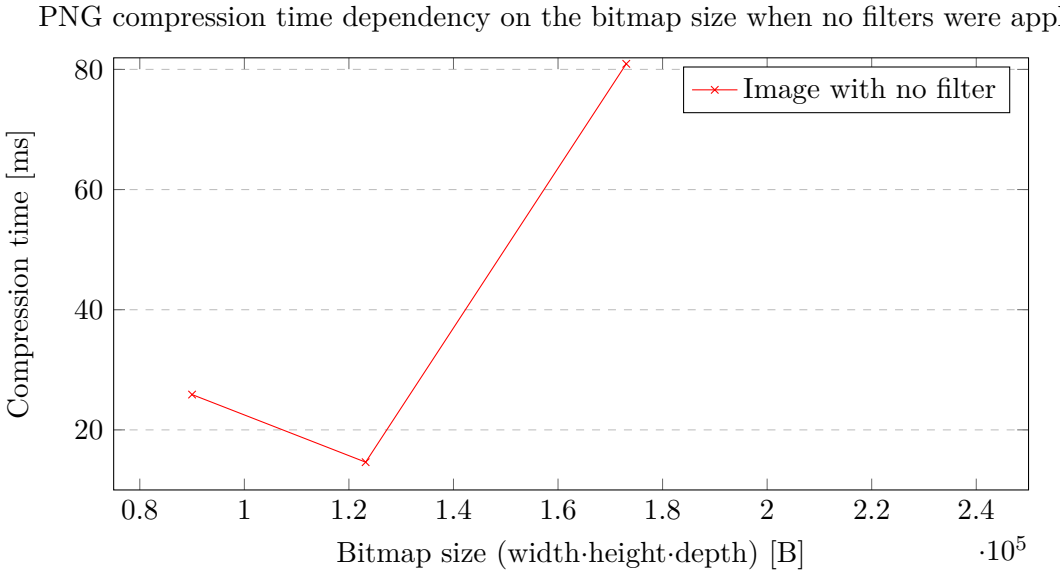


Figure D.13: Plot shows the longest PNG compression time for each bitmap size when image was not pre-processed (filters were not applied)

D.5.5 The fastest WSQ compression times for bitmap size

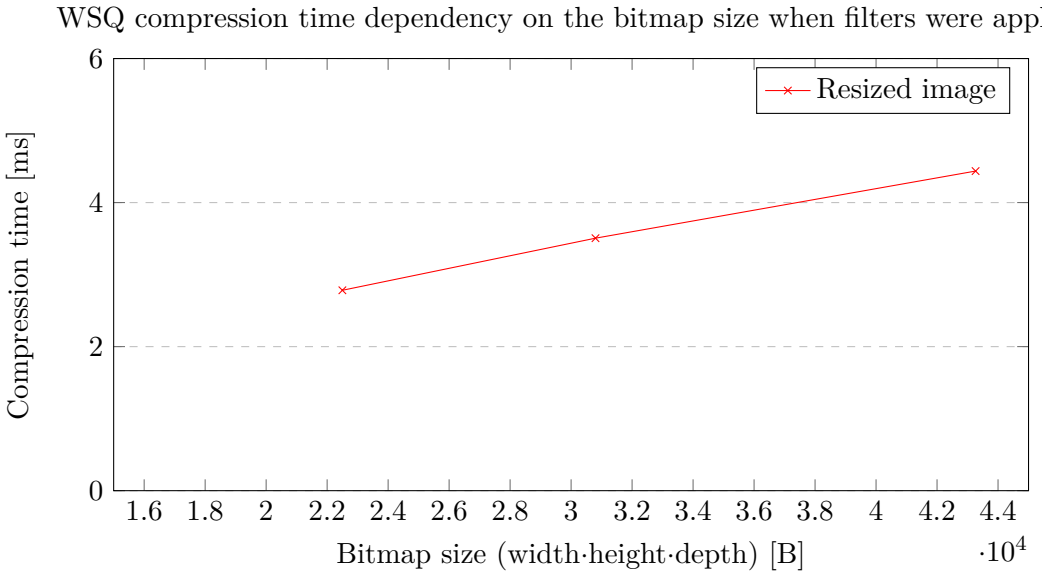


Figure D.14: Plot shows the shortest WSQ compression time for each bitmap size when image was resized to 50% of the original size

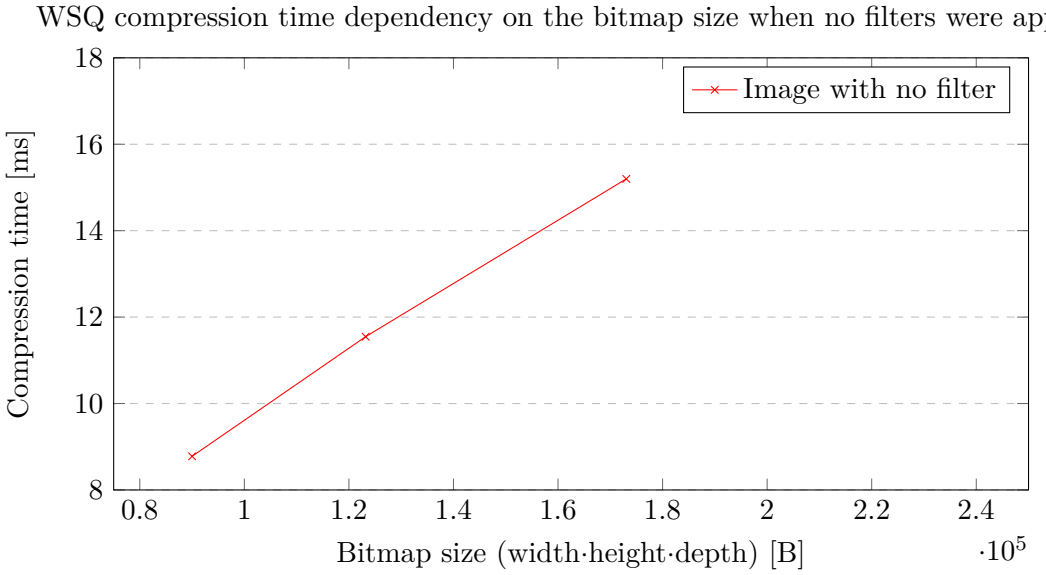


Figure D.15: Plot shows the shortest WSQ compression time for each bitmap size when image was not pre-processed (filters were not applied)

D.5.6 The longest WSQ compression times for bitmap size

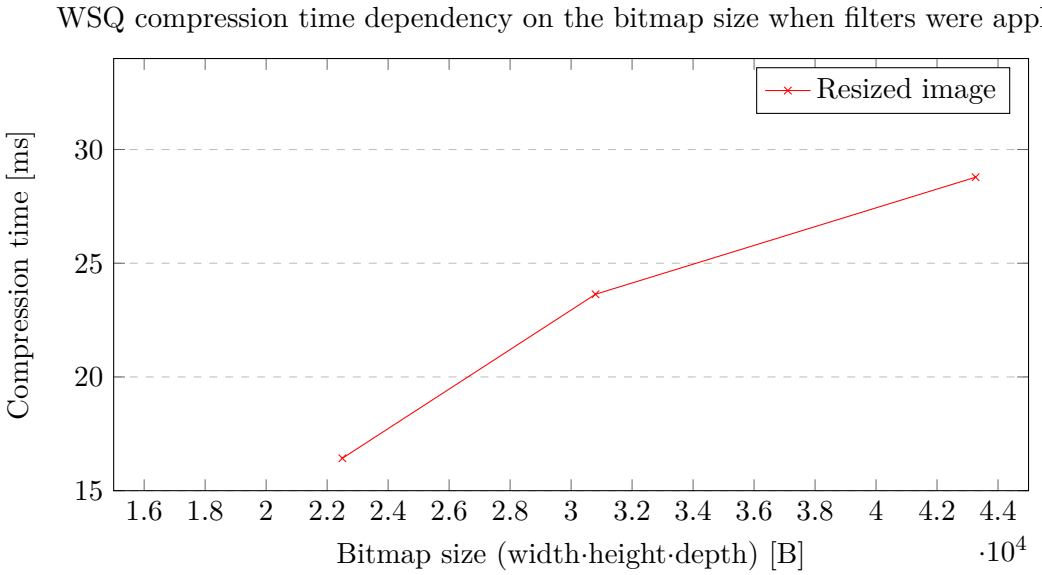


Figure D.16: Plot shows the longest WSQ compression time for each bitmap size when image was resized to 50% of the original size

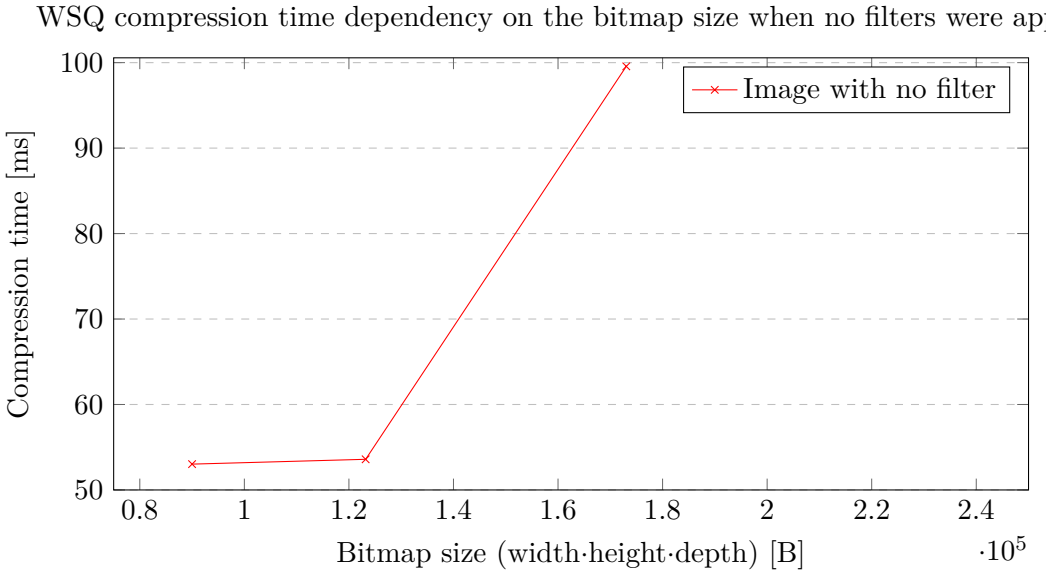


Figure D.17: Plot shows the longest WSQ compression time for each bitmap size when image was not pre-processed (filters were not applied)

Appendix E

Ill fingerprint results

E.1 Matching scores

E.1.1 The best minutiae-based scores

Ratio	Filters	Score
100	depth changed to 8b	1050.3138952418142
100	none	1050.3138952418142

Table E.1: The ratios and filters for the best JPEG minutiae-based match

Ratio	Filters	Score
all tested	depth changed to 8b	1179.5927929938084
0	none	1179.5927929938084

Table E.2: The ratios and filters for the best PNG minutiae-based match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score
1.0	depth changed to 8b	801.9407278725625
0.5	none	113.93537859516215

Table E.3: The encodings and filters for the best WSQ minutiae-based match

E.1.2 The worst minutiae-based score

Ratio	Filters	Score
75	resized to 80%	0.0
75	denoised	0.0
75	resized to 150%	0.0
75	depth changed to 1b	0.0
75	normalized, denoised	0.0
75	resized to 125%	0.0
75	normalized, binarized	0.0
75	denoised, binarized	0.0
0, 25, 50, 100	all tested	0.0
75	normalized, denoised, binarized	0.0
75	normalized	0.0
75	resized to 50%	0.0
75	binarized	0.0
75	normalized, denoised, depth changed to 1b	0.0
75	none	0.0

Table E.4: The ratios and filters for the worst JPEG minutiae-based match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Ratio	Filters	Score
all tested	all tested	0.0
0	none	86.39675721909222

Table E.5: The ratios and filters for the worst PNG minutiae-based match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score
all tested	resized to 50%	0.0
all tested	resized to 125%	0.0
all tested	normalized	0.0
all tested	resized to 80%	0.0
all tested	denoised	0.0
all tested	normalized, denoised	0.0
all tested	resized to 150%	0.0
all tested	depth changed to 8b	0.0
0.75	none	0.0

Table E.6: The encodings and filters for the worst WSQ minutiae-based match. Encoding rates of 0.10, 0.25, 0.50, 0.75, 1.00 are understood by all tested

E.1.3 The best cross-correlation scores

Ratio	Filters	Score [%]
all tested	depth changed to 1b	99.0
100	depth changed to 8b	99.0
100	none	99.0

Table E.7: The ratios and filters for the best JPEG cross-correlation match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Ratio	Filters	Score [%]
all tested	depth changed to 1b	99.0
all tested	depth changed to 8b	99.0
0	none	99.0

Table E.8: The ratios and filters for the best PNG cross-correlation match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score [%]
1.0	depth changed to 8b	95.0
1.0	none	95.0

Table E.9: The encodings and filters for the best WSQ cross-correlation match

E.1.4 The worst cross-correlation scores

Ratio	Filters	Score [%]
50, 75, 100	resized to 80%	0.0
50, 75, 100	normalized, denoised, depth changed to 1b	0.0
50, 75, 100	normalized, binarized	0.0
50, 75, 100	resized to 150%	0.0
50, 75, 100	resized to 50%	0.0
50, 75, 100	binarized	0.0
50, 75, 100	normalized, denoised, binarized	0.0
50, 75, 100	depth changed to 8b	0.0
50, 75, 100	normalized, denoised	0.0
50, 75, 100	denoised	0.0
50, 75, 100	resized to 125%	0.0
0, 25	all tested	0.0
50, 75, 100	denoised, binarized	0.0
50, 75, 100	normalized	0.0
100	none	0.0

Table E.10: The ratios and filters for the worst JPEG cross-correlation match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Ratio	Filters	Score [%]
all tested	resized to 50%	0.0
all tested	resized to 125%	0.0
all tested	normalized	0.0
all tested	binarized	0.0
all tested	resized to 80%	0.0
all tested	denoised	0.0
all tested	depth changed to 8b	0.0
all tested	normalized, denoised, depth changed to 1b	0.0
all tested	normalized, denoised, binarized	0.0
all tested	normalized, binarized	0.0
all tested	denoised, binarized	0.0
all tested	normalized, denoised	0.0
all tested	resized to 150%	0.0
0	none	99.0

Table E.11: The ratios and filters for the worst PNG cross-correlation match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score [%]
all tested	resized to 50%	0.0
all tested	resized to 125%	0.0
all tested	normalized	0.0
all tested	resized to 80%	0.0
all tested	denoised	0.0
all tested	normalized, denoised	0.0
all tested	resized to 150%	0.0
all tested	depth changed to 8b	0.0
0.75	none	0.0

Table E.12: The encodings and filters for the worst WSQ cross-correlation match. Encoding rates of 0.10, 0.25, 0.50, 0.75, 1.00 are understood by all tested

E.1.5 The best PSNR scores

Ratio	Filters	Score [dB]
100	depth changed to 8b	46.16888504837653
100	none	46.16888504837653

Table E.13: The ratios and filters for the best JPEG PSNR match

Encoding	Filters	Score [dB]
1.0	depth changed to 8b	24.29071532212435
1.0	none	24.29071532212435

Table E.14: The encodings and filters for the best WSQ PSNR match

E.1.6 The worst PSNR scores

Ratio	Filters	Score [dB]
0	depth changed to 8b	2.900830129464076
0	none	5.881392951683909

Table E.15: The ratios and filters for the worst JPEG PSNR match

Encoding	Filters	Score [dB]
0.1	normalized, denoised	5.296364236116347
0.1	none	5.125237162034955

Table E.16: The encodings and filters for the worst WSQ PSNR match

E.2 Compression times

E.2.1 The shortest compression times

Ratio	Filters	Time [ms]
0	resized to 50%	0.566321
0	none	1.588024

Table E.17: The ratios and filters for the shortest JPEG compression time

Ratio	Filters	Time [ms]
100	resized to 50%	0.39697
100	none	1.109979

Table E.18: The ratios and filters for the shortest PNG compression time

Encoding	Filters	Time [ms]
0.1	resized to 50%	2.429921
0.1	none	5.843844

Table E.19: The encodings and filters for the shortest WSQ compression time

E.2.2 The longest compression times

Ratio	Filters	Time [ms]
100	resized to 150%	198.874218
100	none	97.324384

Table E.20: The ratios and filters for the longest JPEG compression time

Ratio	Filters	Time [ms]
0	resized to 150%	787.195029
0	none	425.592341

Table E.21: The ratios and filters for the longest PNG compression time

Encoding	Filters	Time [ms]
1.0	resized to 150%	655.620694
1.0	none	297.571566

Table E.22: The encodings and filters for the longest WSQ compression time

E.3 Compressed file sizes

E.3.1 The smallest file sizes

Ratio	Filters	Size [B]
0	resized to 50%	1442
0	none	3049

Table E.23: The ratios and filters for the smallest JPEG file size

Ratio	Filters	Size [B]
0	denoised, binarized	368
0	none	31345

Table E.24: The ratios and filters for the smallest PNG file size

Encoding	Filters	Size [B]
0.1	resized to 50%	729
0.1	none	860

Table E.25: The encodings and filters for the smallest WSQ file size

E.3.2 The biggest file sizes

Ratio	Filters	Size [B]
100	resized to 150%	2195339
100	none	1109765

Table E.26: The ratios and filters for the biggest JPEG file size

Ratio	Filters	Size [B]
100	resized to 150%	8853549
100	none	3935287

Table E.27: The ratios and filters for the biggest PNG file size

Encoding	Filters	Size [B]
1.0	resized to 150%	195215
1.0	none	77862

Table E.28: The encodings and filters for the biggest WSQ file size

E.4 The average values

E.4.1 The average minutiae-based scores

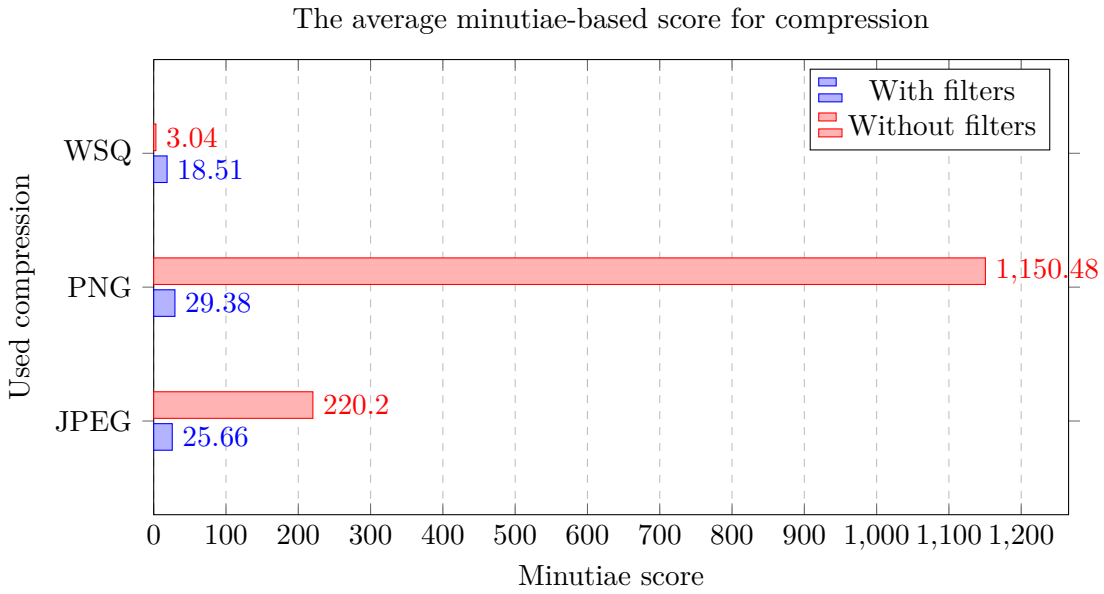


Figure E.1: The average minutiae-based score for compression. Score more than 40 means the fingerprints matched.

E.4.2 The average cross-correlation scores

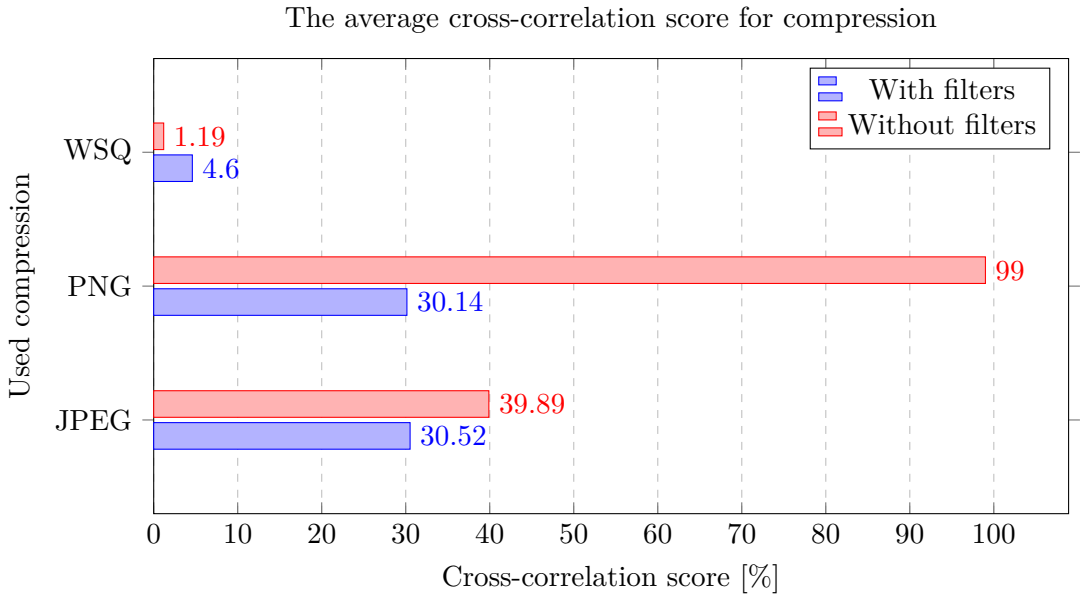


Figure E.2: The average cross-correlation score for compression. Score expresses percentage match between fingerprints.

E.4.3 The average PSNR scores

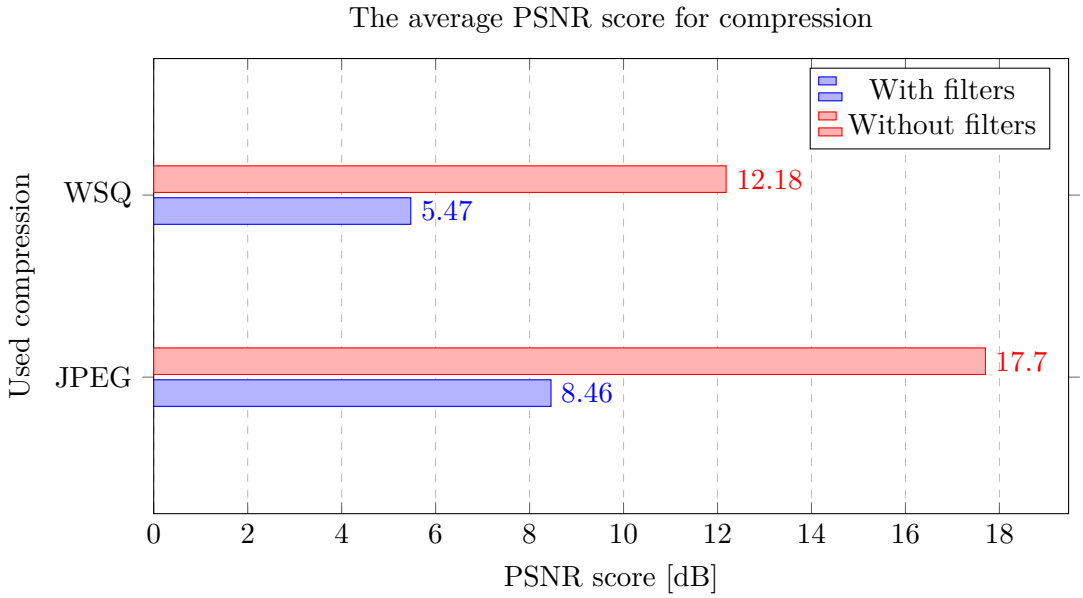


Figure E.3: The average PSNR score for compression. Score more than 40 means the fingerprints matched.

E.4.4 The average compression times

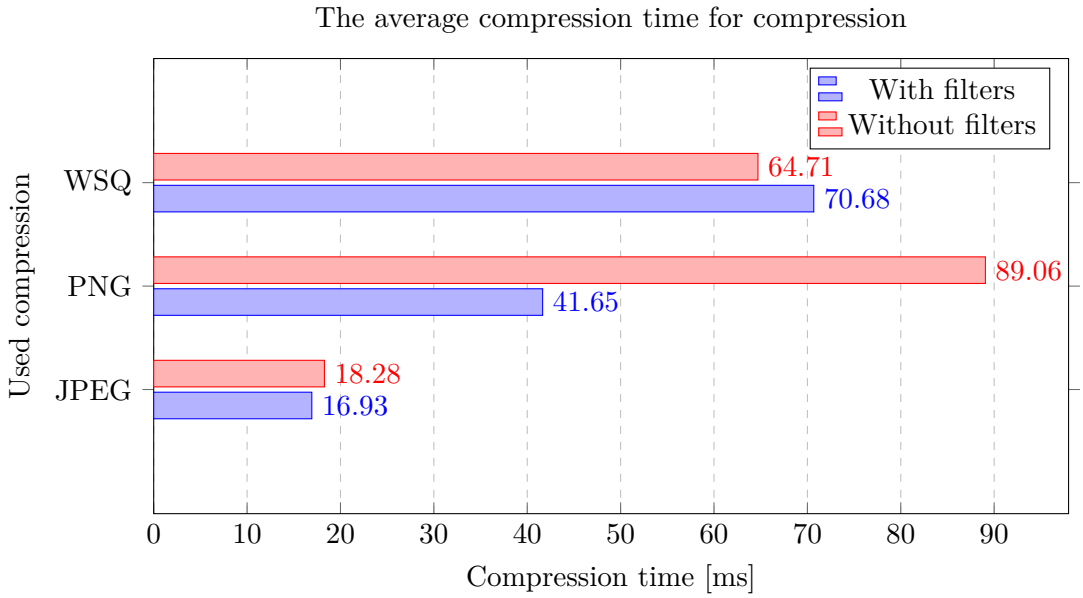


Figure E.4: The average compression time for compression in milliseconds.

E.4.5 The average compressed file sizes

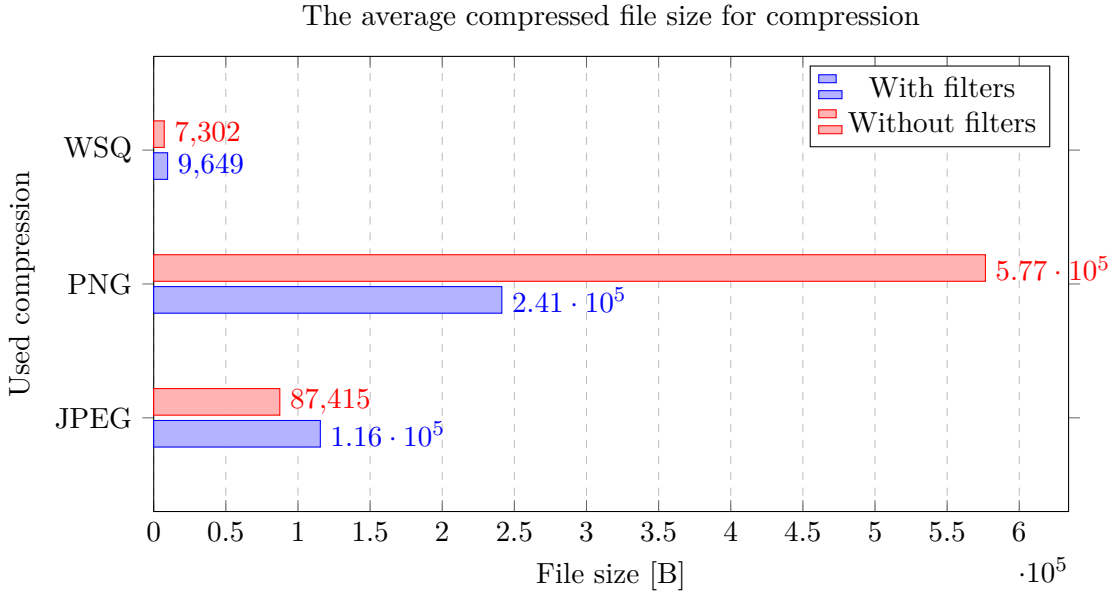


Figure E.5: The average compressed file size for compression in Bytes

E.5 Compression time dependency on the bitmap size

E.5.1 The fastest JPEG compression times for bitmap size

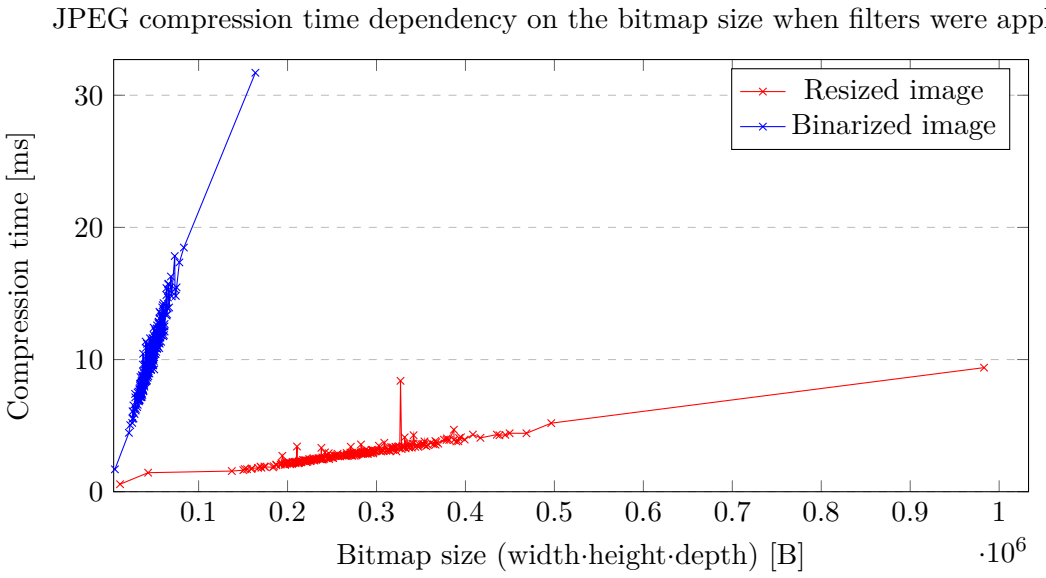


Figure E.6: Plot shows the fastest JPEG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

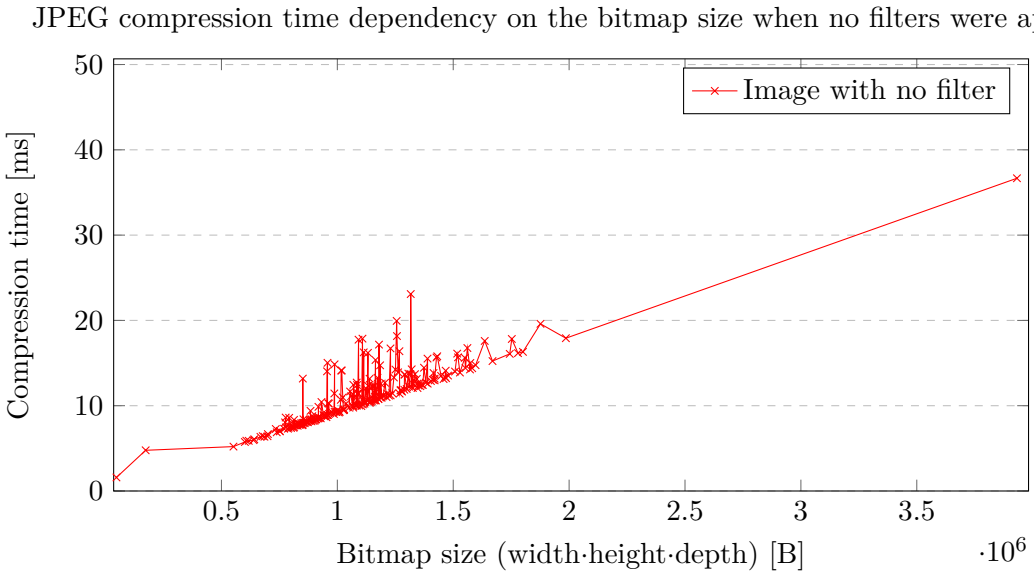


Figure E.7: Plot shows the shortest JPEG compression time for each bitmap size when image was not pre-processed (filters were not applied)

E.5.2 The longest JPEG compression times for bitmap size

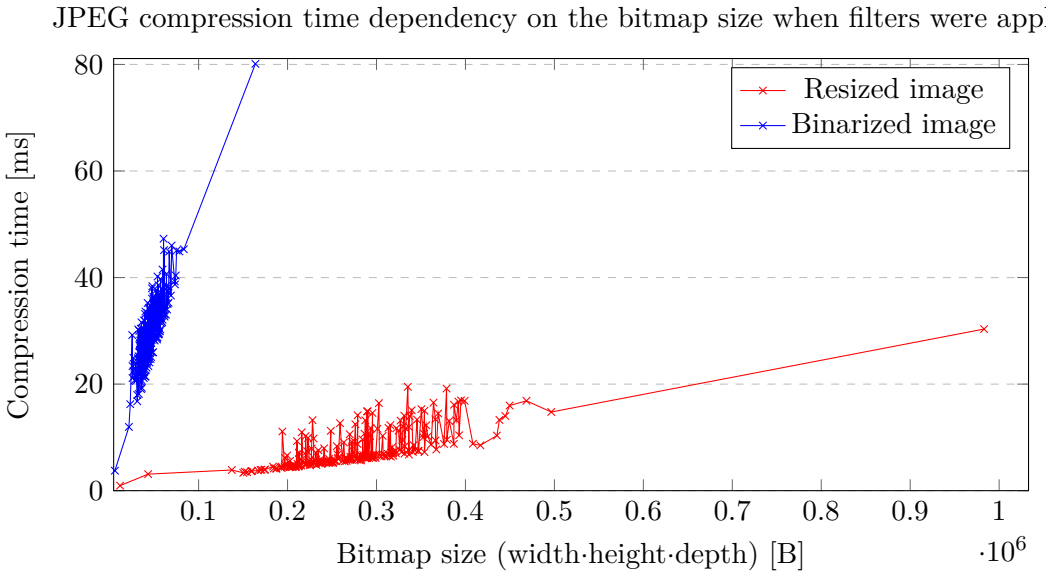


Figure E.8: Plot shows the longest JPEG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

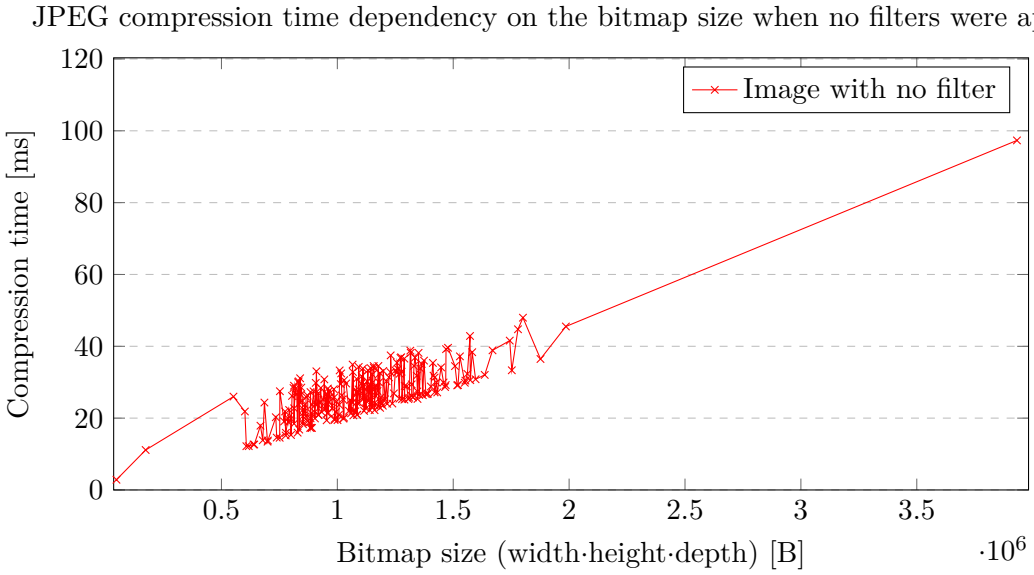


Figure E.9: Plot shows the longest JPEG compression time for each bitmap size when image was not pre-processed (filters were not applied)

E.5.3 The fastest PNG compression times for bitmap size

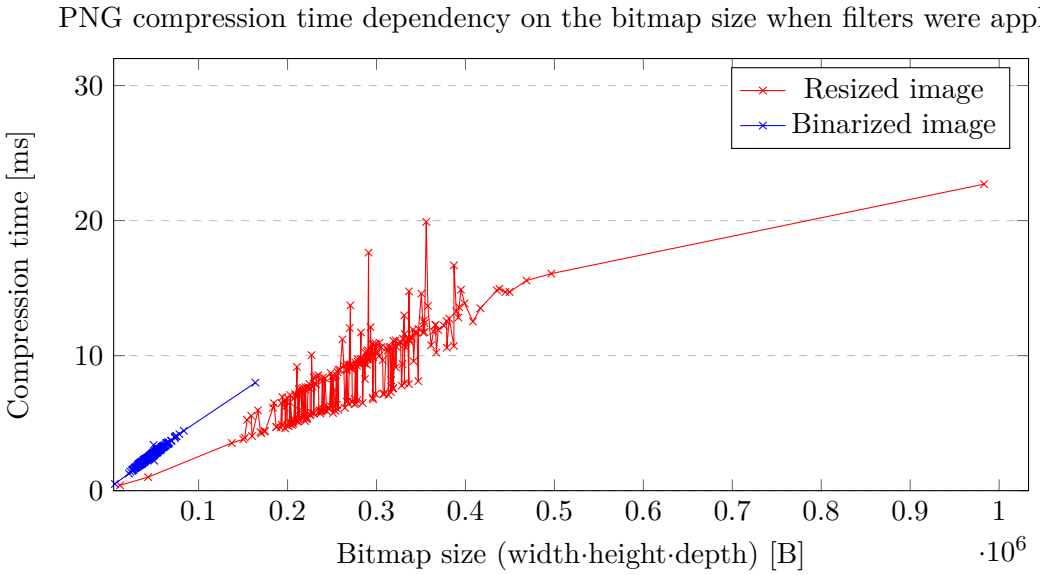


Figure E.10: Plot shows the shortest PNG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

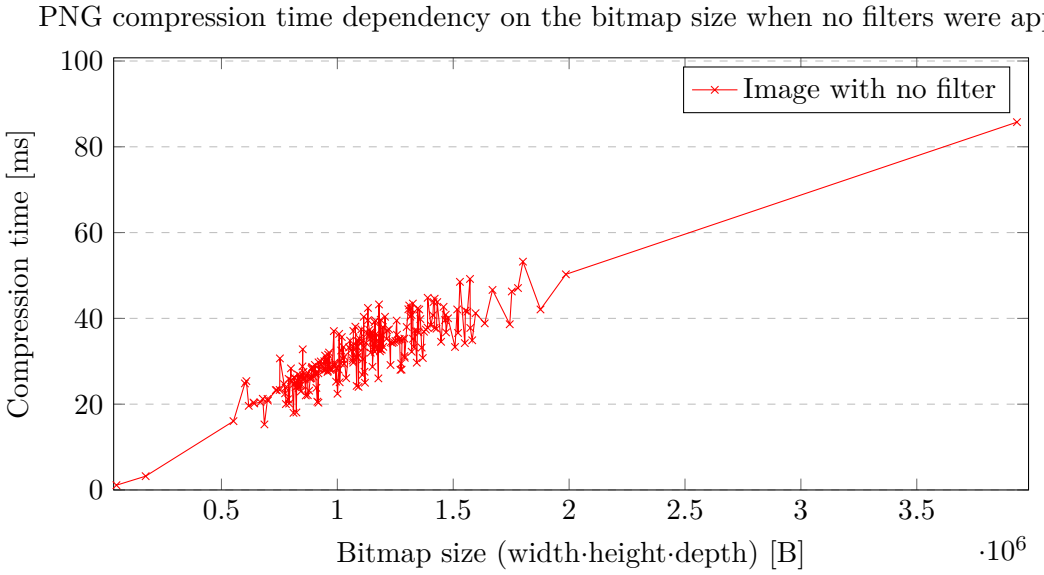


Figure E.11: Plot shows the shortest PNG compression time for each bitmap size when image was not pre-processed (filters were not applied)

E.5.4 The longest PNG compression times for bitmap size

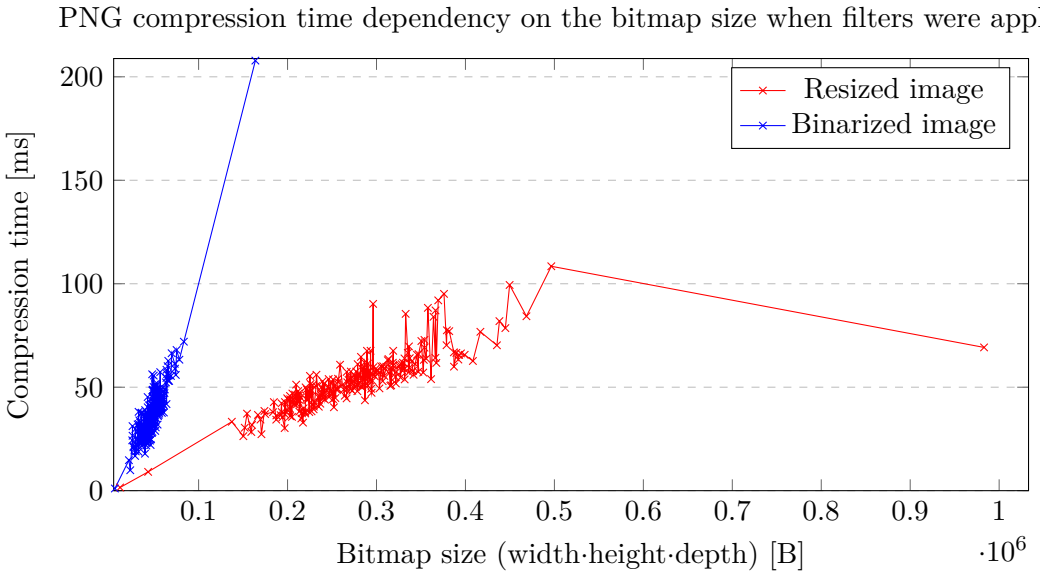


Figure E.12: Plot shows the longest PNG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

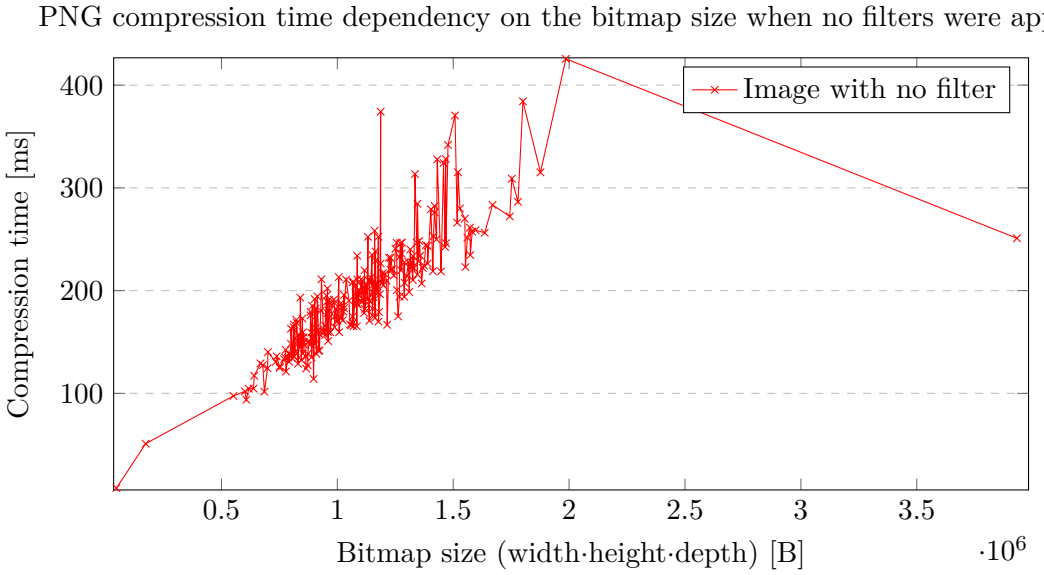


Figure E.13: Plot shows the longest PNG compression time for each bitmap size when image was not pre-processed (filters were not applied)

E.5.5 The fastest WSQ compression times for bitmap size

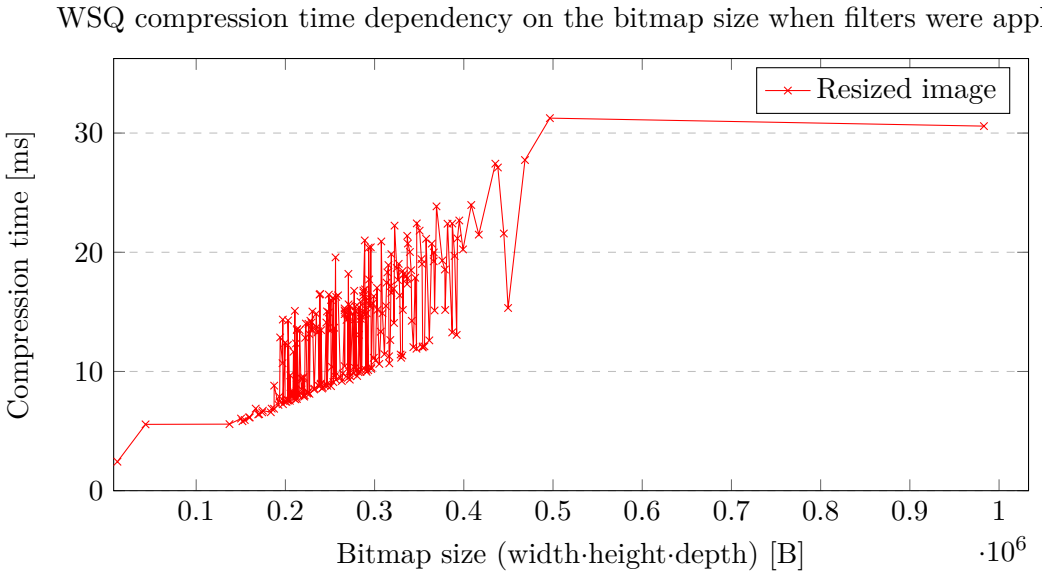


Figure E.14: Plot shows the shortest WSQ compression time for each bitmap size when image was resized to 50% of the original size

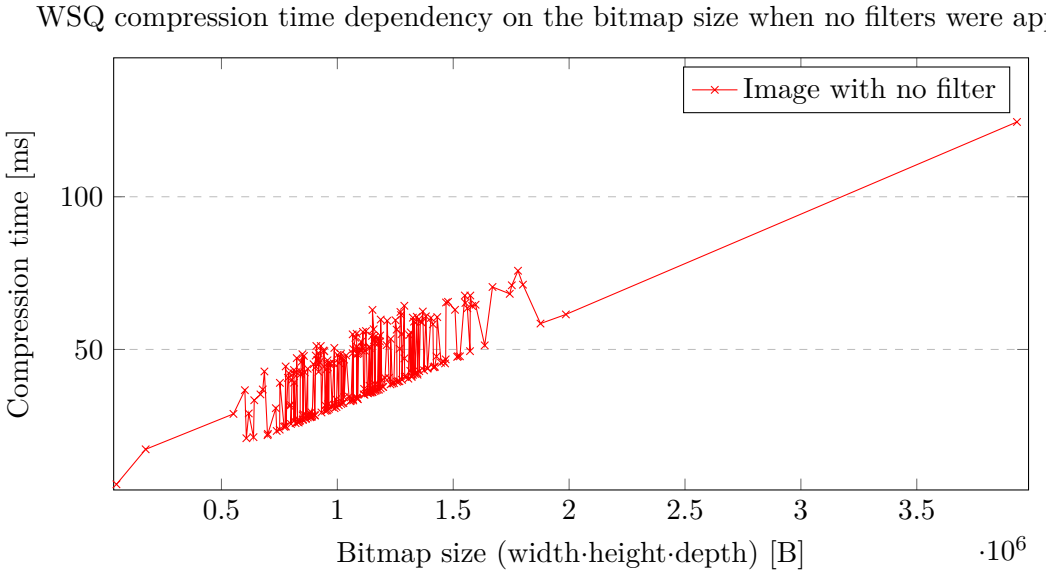


Figure E.15: Plot shows the shortest WSQ compression time for each bitmap size when image was not pre-processed (filters were not applied)

E.5.6 The longest WSQ compression times for bitmap size

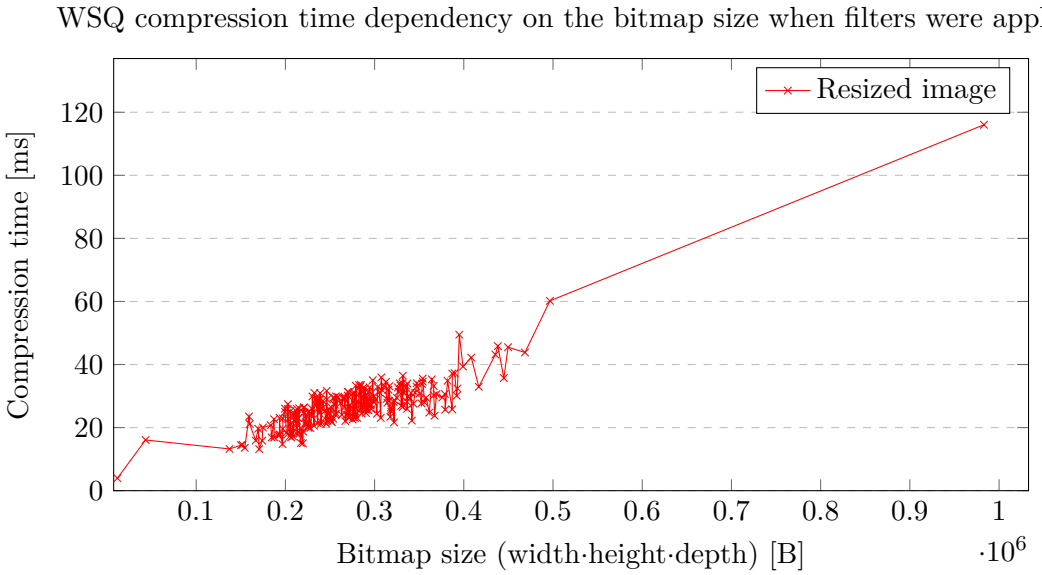


Figure E.16: Plot shows the longest WSQ compression time for each bitmap size when image was resized to 50% of the original size

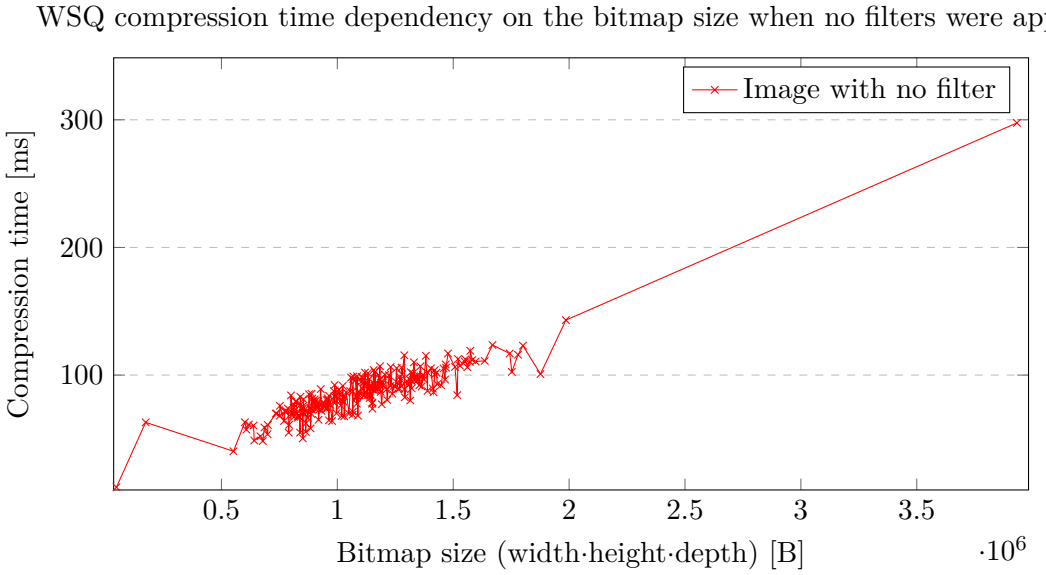


Figure E.17: Plot shows the longest WSQ compression time for each bitmap size when image was not pre-processed (filters were not applied)

Appendix F

Synthetic fingerprint results

F.1 Matching scores

F.1.1 The best minutiae-based scores

Ratio	Filters	Score
100	depth changed to 8b	765.7814791598335
100	none	765.7814791598335

Table F.1: The ratios and filters for the best JPEG minutiae-based match

Ratio	Filters	Score
all tested	depth changed to 8b	864.8146929212922
0	none	864.8146929212922

Table F.2: The ratios and filters for the best PNG minutiae-based match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score
0.75	depth changed to 8b	532.0896747922129
0.75	none	532.0896747922129

Table F.3: The encodings and filters for the best WSQ minutiae-based match

F.1.2 The worst minutiae-based score

Ratio	Filters	Score
all tested	resized to 50%	0.0
75	resized to 80%	0.0
all tested	normalized, denoised, depth changed to 1b	0.0
all tested	normalized, denoised	0.0
all tested	denoised, binarized	0.0
all tested	normalized, denoised, binarized	0.0
all tested	denoised	0.0
0, 25	resized to 150%	0.0
all tested	depth changed to 1b	0.0
25, 50, 75, 100	normalized	0.0
75	normalized, binarized	0.0
0	none	1.1924114947552458

Table F.4: The ratios and filters for the worst JPEG minutiae-based match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Ratio	Filters	Score
all tested	normalized, denoised, binarized	0.0
all tested	resized to 50%	0.0
all tested	normalized, binarized	0.0
all tested	normalized	0.0
all tested	denoised, binarized	0.0
all tested	denoised	0.0
all tested	normalized, denoised	0.0
all tested	depth changed to 1b	0.0
all tested	normalized, denoised, depth changed to 1b	0.0
0	none	165.30780387125566

Table F.5: The ratios and filters for the worst PNG minutiae-based match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score
all tested	resized to 150%	0.0
all tested	resized to 125%	0.0
all tested	resized to 50%	0.0
all tested	resized to 80%	0.0
all tested	normalized, denoised	0.0
all tested	denoised	0.0
0.1	normalized	0.0
0.75	none	0.0

Table F.6: The encodings and filters for the worst WSQ minutiae-based match. Encoding rates of 0.10, 0.25, 0.50, 0.75, 1.00 are understood by all tested

F.1.3 The best cross-correlation scores

Ratio	Filters	Score [%]
0, 50, 100	resized to 150%	99.0
25	resized to 80%	99.0
all tested	depth changed to 8b	99.0
25, 50, 75, 100	normalized, denoised, binarized	99.0
all tested	binarized	99.0
all tested	depth changed to 1b	99.0
50	denoised	99.0
all tested	normalized, binarized	99.0
all tested	resized to 125%	99.0
all tested	normalized	99.0
100	none	99.0

Table F.7: The ratios and filters for the best JPEG cross-correlation match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Ratio	Filters	Score [%]
all tested	depth changed to 1b	99.0
all tested	normalized, binarized	99.0
all tested	normalized	99.0
all tested	resized to 125%	99.0
all tested	depth changed to 8b	99.0
all tested	normalized, denoised, binarized	99.0
all tested	resized to 150%	99.0
all tested	binarized	99.0
0	none	99.0

Table F.8: The ratios and filters for the best PNG cross-correlation match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score [%]
0.75	depth changed to 8b	99.0
0.25	resized to 150%	99.0
0.1	normalized	99.0
0.75	resized to 125%	99.0
0.75	none	99.0

Table F.9: The encodings and filters for the best WSQ cross-correlation match

F.1.4 The worst cross-correlation scores

Ratio	Filters	Score [%]
all tested	all tested	0.0
100	none	0.0

Table F.10: The ratios and filters for the worst JPEG cross-correlation match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Ratio	Filters	Score [%]
all tested	all tested	0.0
0	none	0.0

Table F.11: The ratios and filters for the worst PNG cross-correlation match. Ratios of 0, 25, 50, 75, 100 are understood by all tested

Encoding	Filters	Score [%]
all tested	resized to 50%	0.0
all tested	resized to 125%	0.0
all tested	normalized	0.0
all tested	resized to 80%	0.0
all tested	denoised	0.0
all tested	normalized, denoised	0.0
all tested	resized to 150%	0.0
all tested	depth changed to 8b	0.0
0.75	none	0.0

Table F.12: The encodings and filters for the worst WSQ cross-correlation match. Encoding rates of 0.10, 0.25, 0.50, 0.75, 1.00 are understood by all tested

F.1.5 The best PSNR scores

Ratio	Filters	Score [dB]
100	depth changed to 8b	65.90799998896398
100	none	65.90799998896398

Table F.13: The ratios and filters for the best JPEG PSNR match

Encoding	Filters	Score [dB]
1.0	depth changed to 8b	36.11691443992307
1.0	none	36.11691443992307

Table F.14: The encodings and filters for the best WSQ PSNR match

F.1.6 The worst PSNR scores

Ratio	Filters	Score [dB]
75	depth changed to 8b	12.153884281751854
75	none	12.153884281751854

Table F.15: The ratios and filters for the worst JPEG PSNR match

Encoding	Filters	Score [dB]
0.1	depth changed to 8b	6.470472992165894
0.1	none	6.470472992165894

Table F.16: The encodings and filters for the worst WSQ PSNR match

F.2 Compression times

F.2.1 The shortest compression times

Ratio	Filters	Time [ms]
0	resized to 50%	0.918237
0	none	3.119148

Table F.17: The ratios and filters for the shortest JPEG compression time

Ratio	Filters	Time [ms]
100	resized to 50%	0.647011
100	none	2.127085

Table F.18: The ratios and filters for the shortest PNG compression time

Encoding	Filters	Time [ms]
0.1	resized to 50%	3.304723
0.1	none	10.823851

Table F.19: The encodings and filters for the shortest WSQ compression time

F.2.2 The longest compression times

Ratio	Filters	Time [ms]
0	normalized, denoised, binarized	37.090806
100	none	17.954151

Table F.20: The ratios and filters for the longest JPEG compression time

Ratio	Filters	Time [ms]
0	normalized, denoised	217.210143
0	none	69.039863

Table F.21: The ratios and filters for the longest PNG compression time

Encoding	Filters	Time [ms]
1.0	resized to 150%	122.256735
1.0	none	75.895407

Table F.22: The encodings and filters for the longest WSQ compression time

F.3 Compressed file sizes

F.3.1 The smallest file sizes

Ratio	Filters	Size [B]
0	resized to 50%	1674
0	none	4302

Table F.23: The ratios and filters for the smallest JPEG file size

Ratio	Filters	Size [B]
0	depth changed to 1b	697
0	none	7257

Table F.24: The ratios and filters for the smallest PNG file size

Encoding	Filters	Size [B]
0.1	normalized, denoised	667
0.1	none	866

Table F.25: The encodings and filters for the smallest WSQ file size

F.3.2 The biggest file sizes

Ratio	Filters	Size [B]
100	resized to 150%	293448
100	none	130347

Table F.26: The ratios and filters for the biggest JPEG file size

Ratio	Filters	Size [B]
100	resized to 150%	1574199
100	none	699860

Table F.27: The ratios and filters for the biggest PNG file size

Encoding	Filters	Size [B]
1.0	resized to 150%	39399
1.0	none	16080

Table F.28: The encodings and filters for the biggest WSQ file size

F.4 The average values

F.4.1 The average minutiae-based scores

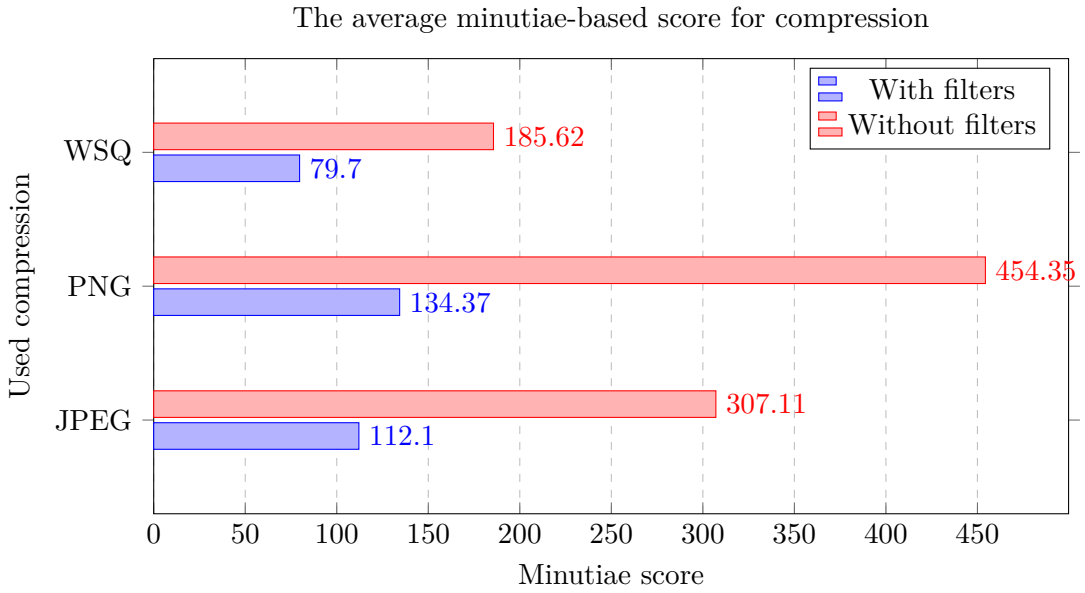


Figure F.1: The average minutiae-based score for compression. Score more than 40 means the fingerprints matched.

F.4.2 The average cross-correlation scores

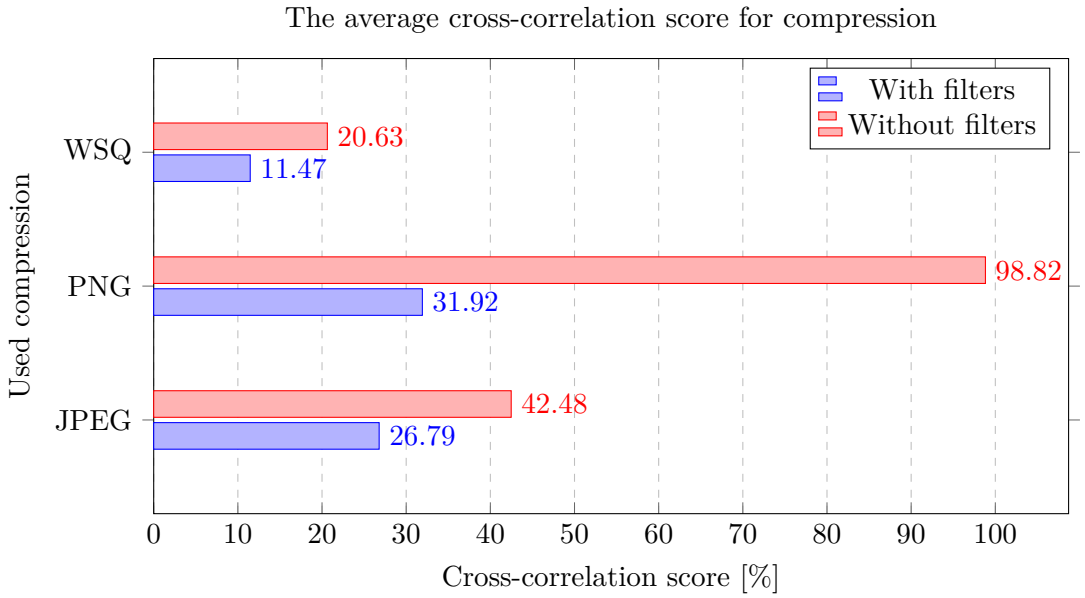


Figure F.2: The average cross-correlation score for compression. Score expresses percentage match between fingerprints.

F.4.3 The average PSNR scores

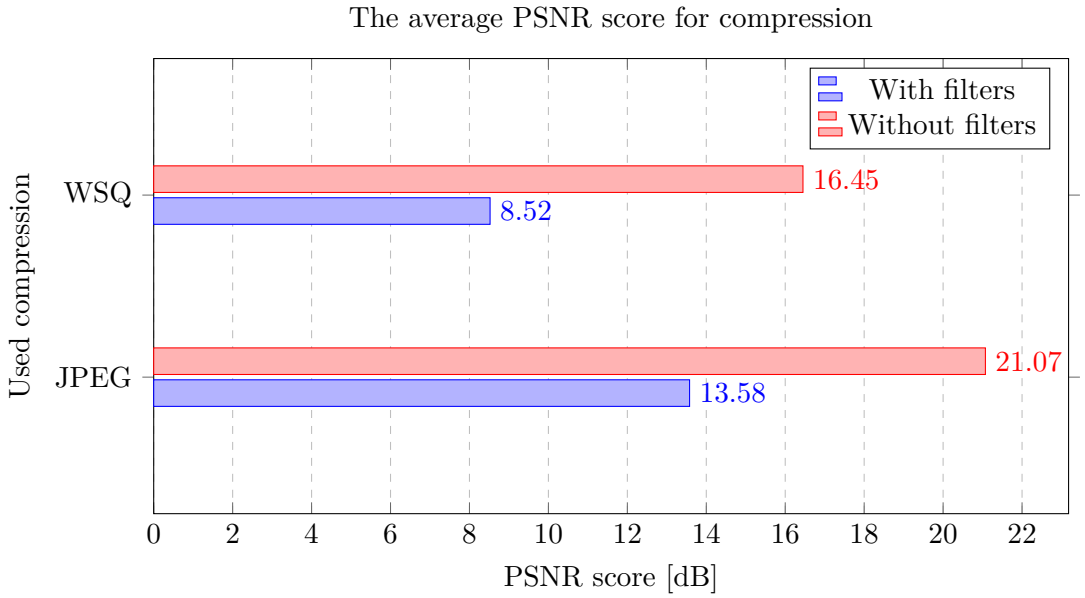


Figure F.3: The average PSNR score for compression. Score more than 40 means the fingerprints matched.

F.4.4 The average compression times

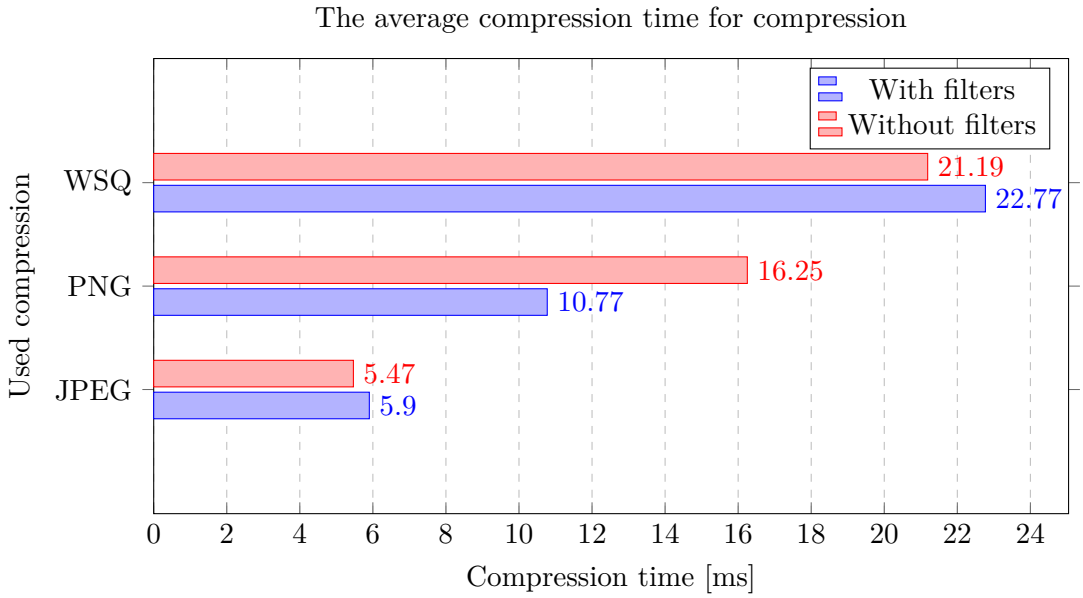


Figure F.4: The average compression time for compression in milliseconds.

F.4.5 The average compressed file sizes

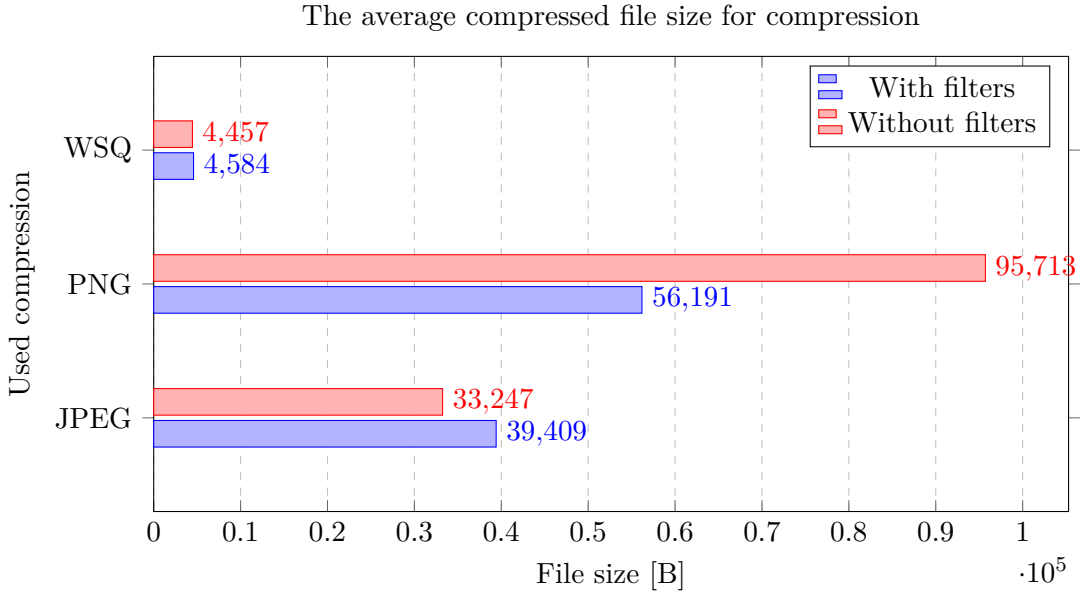


Figure F.5: The average compressed file size for compression in Bytes

F.5 Compression time dependency on the bitmap size

F.5.1 The fastest JPEG compression times for bitmap size

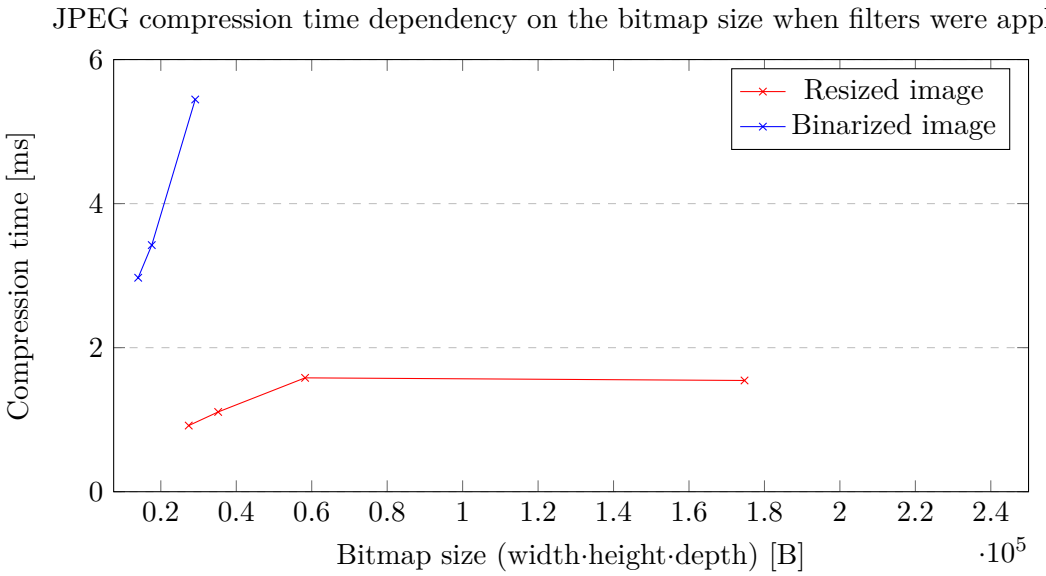


Figure F.6: Plot shows the fastest JPEG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

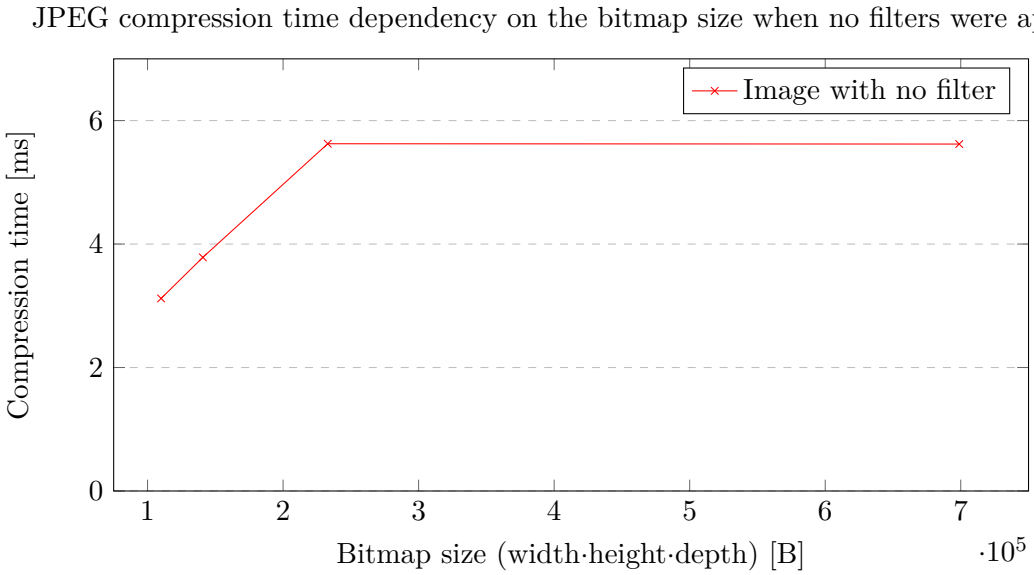


Figure F.7: Plot shows the shortest JPEG compression time for each bitmap size when image was not pre-processed (filters were not applied)

F.5.2 The longest JPEG compression times for bitmap size

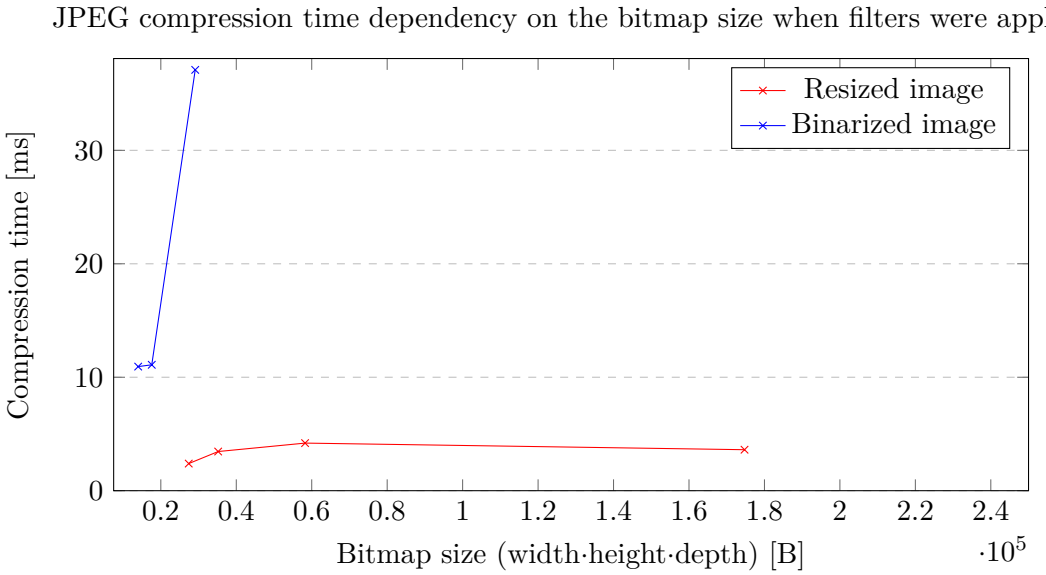


Figure F.8: Plot shows the longest JPEG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

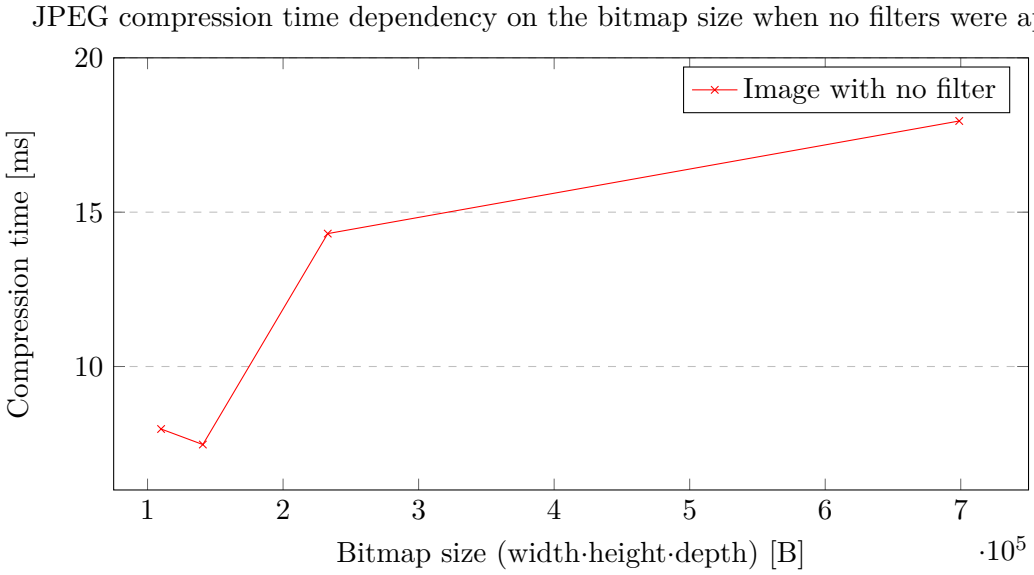


Figure F.9: Plot shows the longest JPEG compression time for each bitmap size when image was not pre-processed (filters were not applied)

F.5.3 The fastest PNG compression times for bitmap size

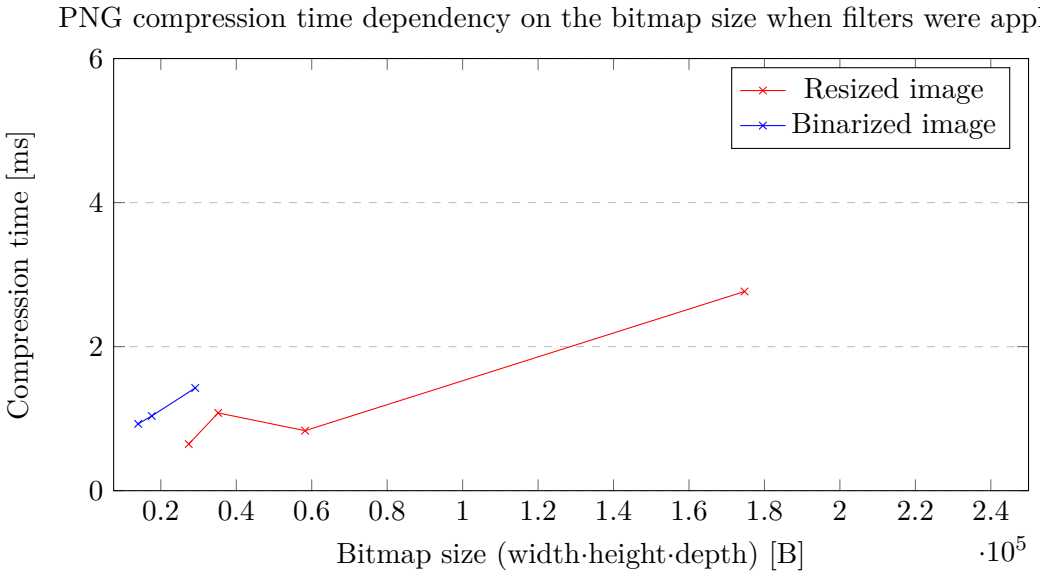


Figure F.10: Plot shows the shortest PNG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

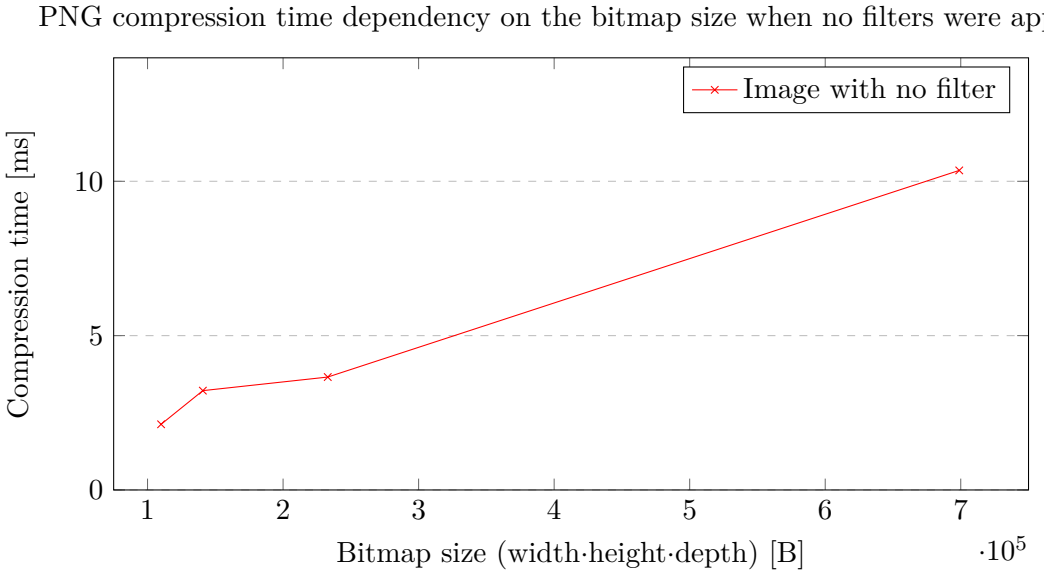


Figure F.11: Plot shows the shortest PNG compression time for each bitmap size when image was not pre-processed (filters were not applied)

F.5.4 The longest PNG compression times for bitmap size

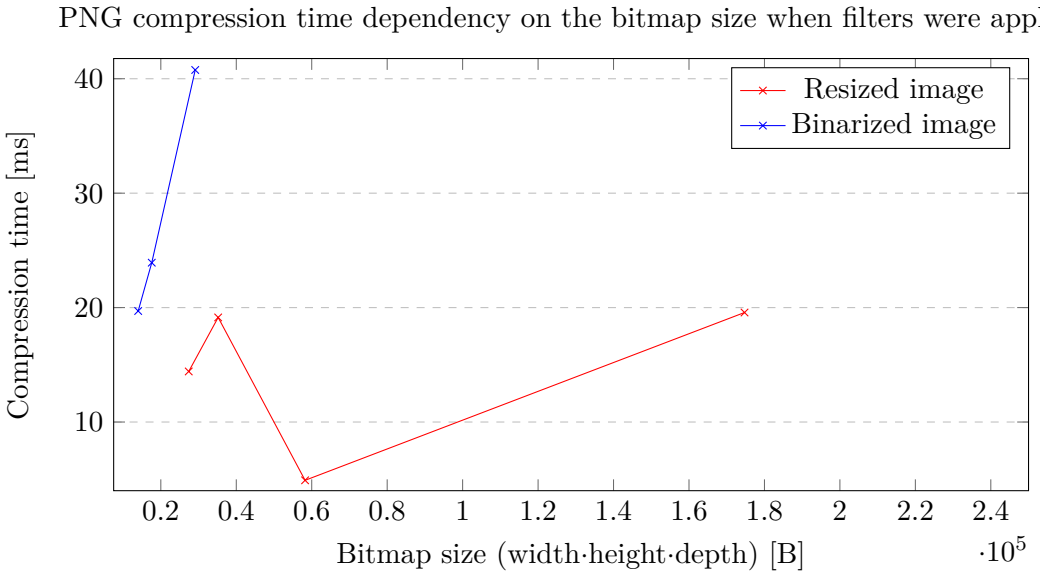


Figure F.12: Plot shows the longest PNG compression time for each bitmap size when image was either resized to 50% of the original size or binarized

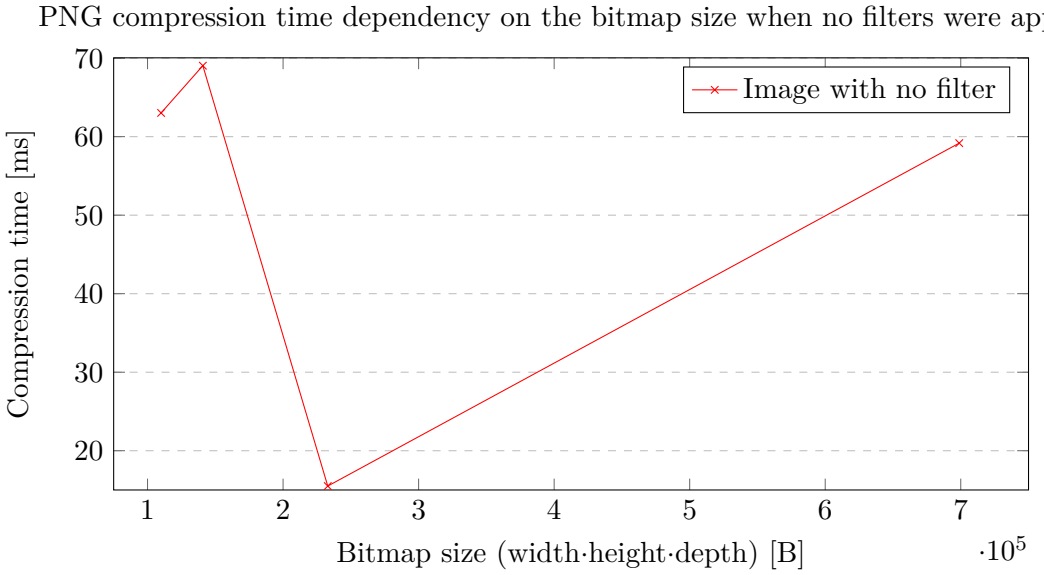


Figure F.13: Plot shows the longest PNG compression time for each bitmap size when image was not pre-processed (filters were not applied)

F.5.5 The fastest WSQ compression times for bitmap size

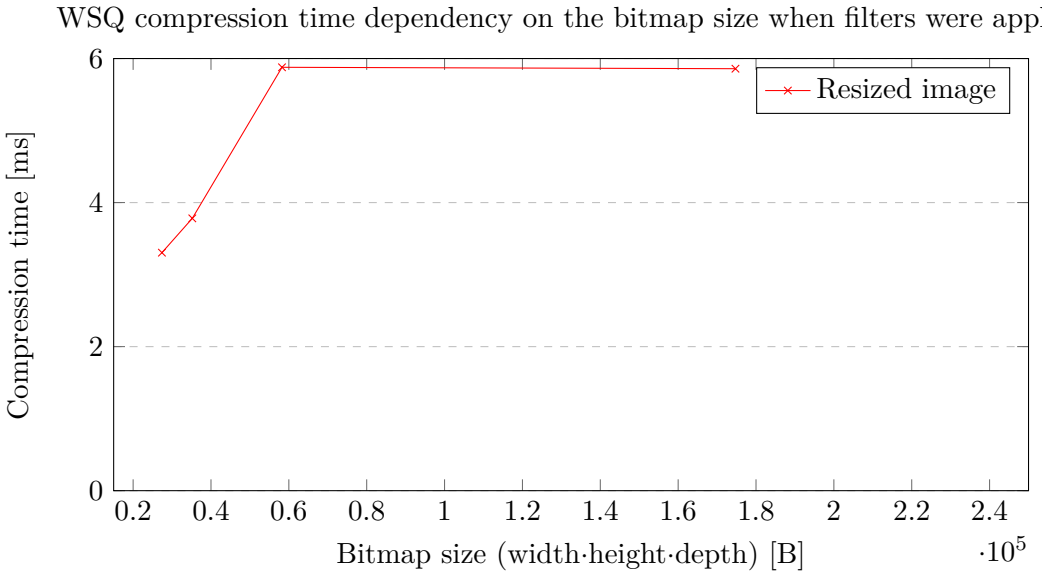


Figure F.14: Plot shows the shortest WSQ compression time for each bitmap size when image was resized to 50% of the original size

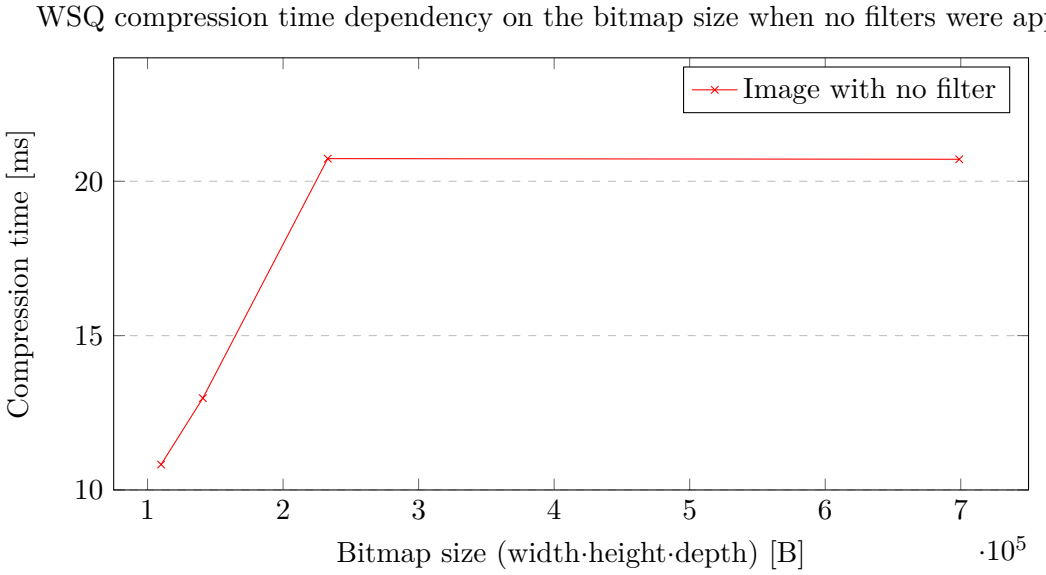


Figure F.15: Plot shows the shortest WSQ compression time for each bitmap size when image was not pre-processed (filters were not applied)

F.5.6 The longest WSQ compression times for bitmap size

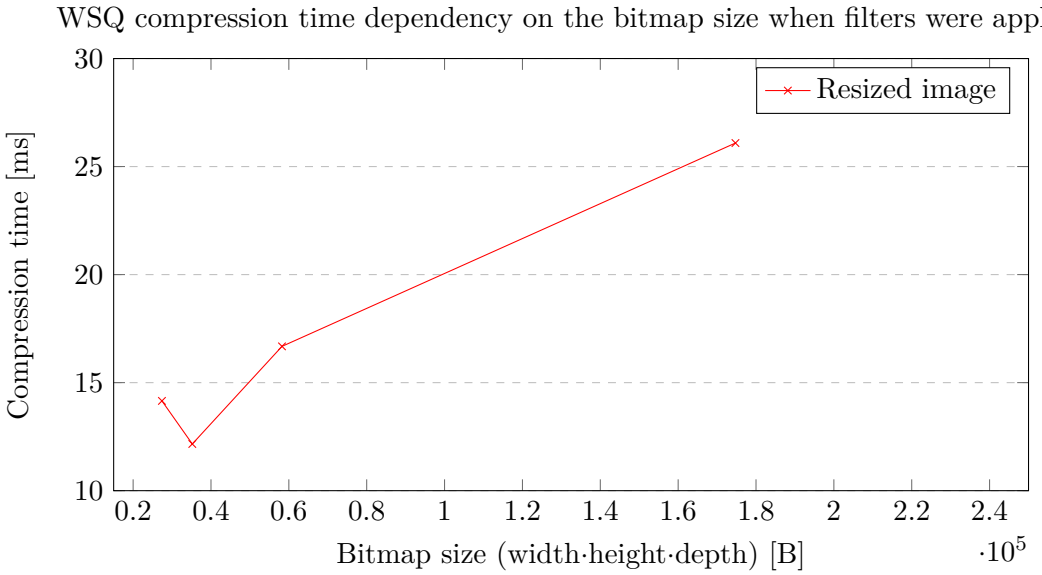


Figure F.16: Plot shows the longest WSQ compression time for each bitmap size when image was resized to 50% of the original size

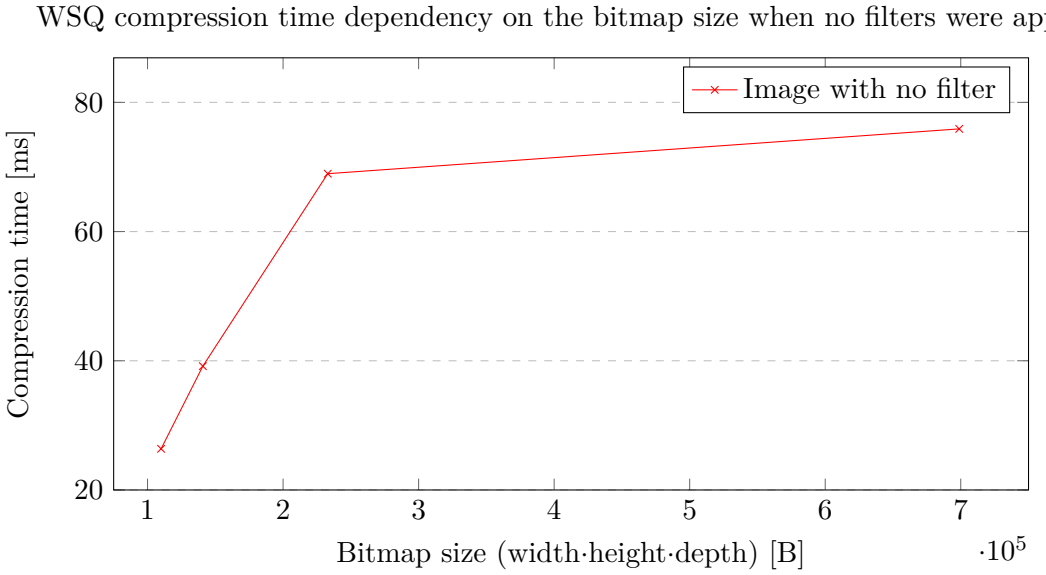


Figure F.17: Plot shows the longest WSQ compression time for each bitmap size when image was not pre-processed (filters were not applied)