CODE TRANSLATION OF SINC-ZOOMING APPLICATION FROM C/C++ TO PURE C LANGUAGE: A COMPUTER ENGINEERING EXPLORATION

A THESIS SUBMITTED TO
THE FACULTY OF ARCHITECTURE AND ENGINEERING
OF
EPOKA UNIVERSITY

BY

GRISILDA DODA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF THE MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JULY, 2023

Approval sheet of the Thesis

This is to certify that we have read this thesis entitled **"Code translation of sinc-zooming application from C/C++ to pure C language: A computer engineering exploration"** and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

____

Dr. Arban Uka
Head of Department
Date:

Examining Committee Members:

Dr. Arban Uka	(Computer Engineering)	_________________
Dr. Carlo Ciulla	(Computer Engineering)	_________________
Dr. Florenc Skuka	(Computer Engineering)	_________________

ABSTRACT

CODE TRANSLATION OF SINC-ZOOMING APPLICATION FROM PURE C LANGUAGE: A COMPUTER ENGINEERING EXPLORATION

Doda, Grisilda

M.Sc., Department of Computer Engineering

Supervisor: Dr. Carlo Ciulla

This thesis presents a computer engineering exploration of the code translation process of a sinc-zooming application from C/C++ to pure C language. Sinc zooming algorithm is a common technique used in signal processing and image resizing applications. The original code is written in C/C ++ which includes object-oriented programming features such as classes. For the method part we have used sinc-zoom theory and translated the original code that involves removing object-oriented features and converting to pure C language. The challenges encountered during this process will be shown primarily related to the differences between these two programing languages and steps taken to successfully translate it. As a result, we generate the same exact images from different percentages of zoom-in, zoom-out and no zoom from both codes. This thesis documents the translation process and analyzes the resulting code by trying to reach the goal of generating the same exact results from both of the codes.

Keywords: *Sinc-zooming, C/C++, pure C programming, translated code, image processing*

ABSTRAKT

PËRKTHIMI I KODIT TË NJË APLIKACIONI NË NJË ZMADHIM TË SIKRONIZUAR NGA GJUHA C/C++ NË C: NJË EKSPLORIM I INXHINIERISË KOMPJUTERIKE

Doda, Grisilda

Master Shkencor, Departamenti i Inxhinierisë Kompjuterike

Udhëheqësi: Dr. Carlo Ciulla

Kjo tezë paraqet një eksplorim të inxhinierisë kompjuterike të proçesit të përkthimit të kodit të një aplikacioni zmadhimi te sinkronizuar nga C/C ++ në gjuhën e pastër C. Algoritmi i zmadhimit të sinkronizuar është një teknikë e zakonshme që përdoret në përpunimin e sinjalit dhe në aplikacionet e ndryshimit të madhësisë së imazhit. Kodi original është i shkruar në C/C ++ i cili përfshin veçori programimi te orientuar nga objekti si klasat. Për pjesën e metodës ne kemi përkthyer kodin që përfshin heqjen e veçorive te orientuara nga objekti dhe konvertimin në gjuhen C. Sfidat e hasura gjatë këtij proçesi do të shfaqen kryesisht në lidhje me ndryshimet midis këtyre dy gjuhëve programuese dhe hapat e ndërmarrë për ta përkthyer me sukses atë. Si rezultat, ne gjenrojmë të njëjtat imazhe të sakta nga përqindje të ndryshme zmadhimi nga të dyja kodet. Kjo tezë dokumenton proçesin e përkthimit dhe analizon kodin duke u përpjekur për të gjeneruar te njëjtat imazhe nga të dyja kodet.

Fjalët kyçe: *Zmadhim i sinkronizuar, C/C++, gjuha C e pastër, kodi i përkthyer, proçesim imazh*

ACKNOWLEDGEMENTS

This thesis contains materials that are sought to be published in a Journal paper. Hence, this thesis cannot be published because it intersects with the current manuscript submission.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Sinc-zooming

On this thesis you will be introduced with image interpolation which is also known as image zooming. We will list the interpolation techniques and show where they are used and also how they are grouped and their formulas. We will also see zooming in optical and digital and have a comparison of both. We will have different zooming methods where authors have done their research and have showed their own proposed method by comparing and trying to obtain better results than with the existing ones. We will translate a sinc-zooming application from C/C++ to pure C programing language and we will set the goal to achieve the same generated images from both codes.

The challenges encountered during this process will be shown that will be primarly related to the diferences between these two programming languages and the steps taken to overcome challenges and to successfully translate it. This thesis documents the translation process and analyzing the resulting code in terms of performance, accuracy and maintanability by trying to reach the goal of generating the same excact results from both of the codes.

1.2 Processing and analyzing results

In this thesis different zooming methods will be shown and compared with other existing methods and will be analyed into showing which one is better and what qualities do they have. Some of our objectives are:

1. Having a general understanding of image interpolation and how it is grouped.

2. Having general information of optical and digital zooming and where are they used.
3. Having different zooming methods being compared with other existing ones and generating images and results.
4. Changes being made from a C/C++ code into a pure C programming language.
5. Generating the same images from both codes.
6. Analyzing and explaining the images that were obtained.

The main objective for this work is to understand every concept that I have mentioned above when you finish reading this thesis. After having general information of sinc zooming tou will see the practical part which is translating the code and generating images.

1.3 Translating the application from C/C++ into pure C programing language

In this section the practical part of the thesis is translating the sinc-zooming application from C/C++ into pure C programming language. After trasnalting the code, we will try to generate zoomed images from both codes and the goal is to obtain the same images. Images are zoomed-in, zoomed-out and no-zoom. Every percentage of the zoom has its own bandwith and sampling rate. ImageJ was used for making the comparison of images generated from code in C/C++ with images generated from the translated code in C.

1.4 Organization of the thesis

This research paper is divided into 5 chapters. The organization of the chapters are done as followed:

Starting with Chapter 1, sinc-zooming, processing, and analyzing results, translating the application from C/C++ into pure C programming language. In Chapter 1, statement of the problem, the objective that thesis has, and scope of work is presented. Going on with Chapter 2 by including the literature review, most of it is about different zooming methods from different authors comparing their proposed method with the existing ones and analyzing which one is more convenient for the zoomed images. Also, we have shown general information of image interpolation, which are the techniques and optical vs digital zooming. Chapter 3 consists of the methodology by using the sinc theory and translating the code into pure C. In Chapter 4, we have results and discussion where we show results of two sample case studies, sample results obtained using theoretical and a magnetic resonance image and results generated from the translated code being compared with the images generated from the original code. In Chapter 5 we have a conclusion.

CHAPTER 2

LITERATURE REVIEW

2.1 What is image interpolation?

Image interpolation is when you resize or distort an image from one pixel grid to another [1]. Image resizing is needed when you want to increase or decrease the number of pixels like zooming or shrinking an image and that can occur under a wide variety of scenarios. Even when it is performed the same image resizing or remapping, results depend on the interpolation algorithm. Each time the interpolation is performed, the image will lose quality. It is done by using known data for estimating values at unknown points. The basic idea of this concept is by first reconstructing a 'continuous' image from the discrete input image and sampling the image into the grid of the output image [2]. Non-adaptive method includes a variety of algorithms (Figure 1) which are [3]:

i. Nearest neighbor is the simplest technique that decides the value of the intensity from the closest pixel to the specified input coordinates and assigning the value to the coordinates of the output. The interpolation kernel of each direction is: u(x) = $\left\{ \begin{array}{ll} 0, & |x| > 0.5 \\ 1, & |x| > 0.5 \end{array} \right\}$

Where x is the distance between the interpolating point and grid point.

ii. Bilinear interpolation uses a weighted average of four neighbourhood pixels for calculating the final interpolated value. This technique generated better results than nearest neighbour interpolation and the time that it takes for computing is less than bicubic interpolation. The interpolation kernel is:

$$u(x) = \left\{ \begin{array}{ll} 0, & |x| > 1 \\ 1 - |x, & |x| < 1 \end{array} \right\}$$

iii. Bicubic image interpolation is the most effective of non-adaptive techniques. It uses a weighted average of sixteen pixels for calculating the final interpolated value. The interpolation kernel is [4]:

$$u(x) = \left\{ \begin{array}{ll} (a+2)|x|^3 - (a+3)|x|^2 + 1, & 0<= |x| > 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a, & 1<=|x| < 2 \end{array} \right\}$$

where a is the free variable.

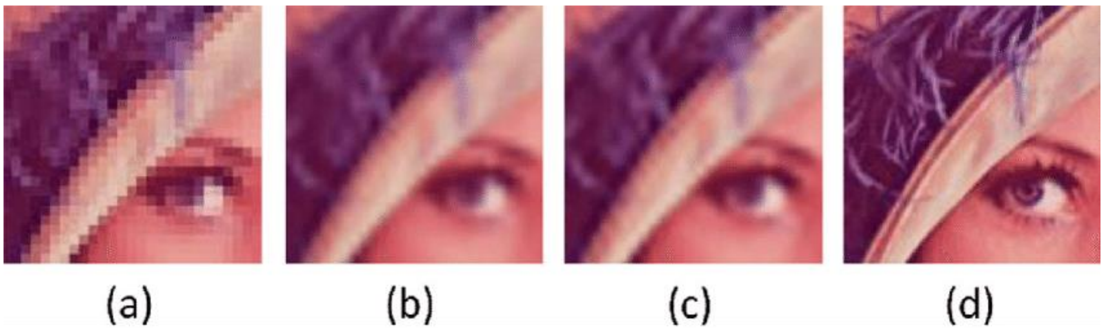


Figure 1.Visual comparison of different interpolation methods. (a) Nearest neighbor. (b) Bilinear. (c) Bicubic. (d) Original HR image (4x) Picture is found at 'Super-resolution via adaptive combination of color channels' [5]

iv. Sinc function

Sinc function (Figure 2) or else known as sine function is a symmetric function denoted by sinc(x) [6]. It is used in mathematics, physics and engineering and it has two definitions:

- Unnormalized sinc function which is usually used in mathematics:

$$\text{Sinc}(x) = \frac{\sin x}{x}$$

- Normalized sinc function which is used in signal processing that includes radio transmission and sound recording:

$$\text{Sinc}(x) = \frac{\sin \pi x}{\pi x}$$

The difference that stands between these two is in the scaling of the independent variable which is on the x-axis by a factor of π. The function has a limit of value 1.

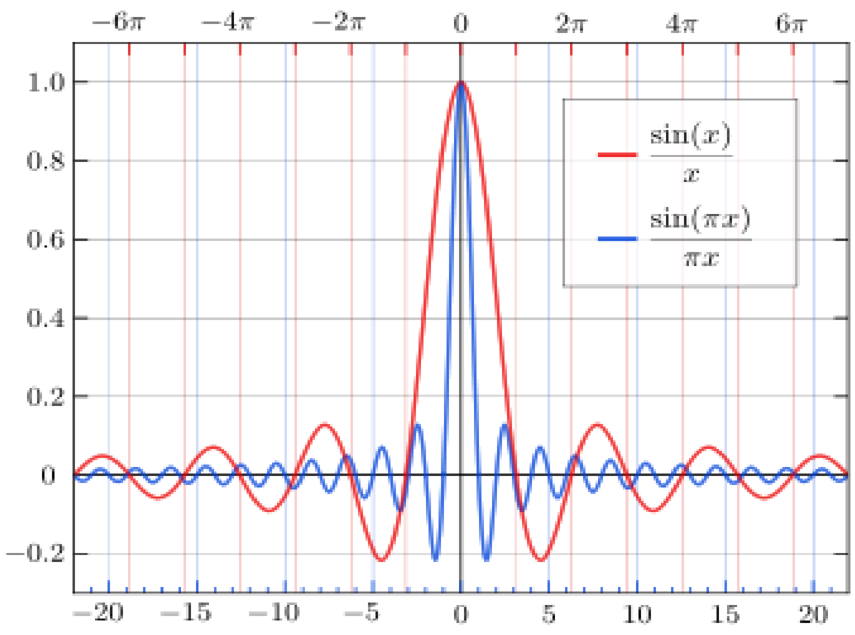


Figure 2.Sinc Function. Figure by Georg-Johann (2010) [7]

2.2 Optical vs Digital zooming

Sinc zooming is a digital signal processing technique used in image and video processing to enlarge or zoom into an image or video while minimizing the loss of quality and sharpness. In traditional image and video zooming, when an image is enlarged, the software simply duplicates pixels, resulting in a blurry and pixelated image. Sinc zooming uses a mathematical function called the sinc function to interpolate new pixels based on the existing ones, resulting in a smoother and more accurate zoomed-in image. It is often used in applications such as video editing, digital photography, and medical imaging, where high-quality zooming is crucial for accurate analysis and diagnosis.

The concept of zooming is for enlarging an image so that the details can be clearer and more visible. The applications that zooming has are wide by using it on a camera lens and ranging to zooming an image on the internet. Zooming once was referred to in photography as using zoom lenses to change the distance that it has between the camera and the subject. With the introduction of digital technology, the concept of zooming has changed and has become more complicated [8].

	Optical Zoom	Digital Zoom
About	For wanting to get a zoomed view of a subject while taking a picture with a camera, without moving pgysically close to the object, optical zoom is used from photographers.	Digital zooming being part of the digital camera, helps croping the entire image and then on a digital way enlarging the size of the object that is needed for zooming on.
Function	The function of the ratio of a digital camera is for measuring how much the lens zoom into the subjects for making them appear closer. The picture is enlarged, but at the same time the resolution and sharpness of the image is kept high.	The function of digital zoom is for croping the image on an area that is centered within the same ratio as the original image and having results back with the same pixel values as the original. While using this method we crop, by reducing the resolution and the quality.
Usage	Whenever we want to get a closer view of an object while taking a picture without reducing teh quality of	Digital zoom is used from the user to get closer the object whenever the photographer has to be

	the entire image of the subject, optical zoom comes in hand.	discrete about taking certain pictures.
Resolution and image quality	Optical zoom zooms the image or the object within a specific range, so there there will be no relation bewtween optical zoom and the resolution of the image.	With digital zoom a portio of the picture is cropped and then it is enlarged back to its original size. After that image quality will be reduced by comparing it with the original one.

Table 1. Comparison table [9].

2.2.1. Different zooming methods

According to the research reported in [10], it is shown a technique used to zoom images based on vector quantization approximation. This technique is used mainly for compressing data called approximation vector. Their idea is to apply this technique to the process of zooming the images, by making sure to have a better zoomed image with all the details. The proposed method wants to achieve improving the quality of the zoomed images using an advanced technique of vector. The paper starts by analyzing the challenges during zooming images and shows the existing methods of zooming images. This method comes with an algorithm to determine quantization vectors and to minimize the loss of data during the zooming process. This method comes with an algorithm to determine quantization vectors and to minimize the loss of data during the zooming process.

It starts by using a vectorization process to separate the image into small blocks. These blocks are later presented with their respective vectors. A certain base of vectors is used to represent these small blocks. Then an approximate vectorization process is used for finding the most nearby vectors with the image blocks. This process uses a form of compressing data, replacing the original vectors of the blocks with their vectors. This creates a form of concentration of data, by easing the zooming process of the image.

With the experiments that are done with different images, authors demonstrate that the proposed method offers improved results compared with the existing methods of zooming images. The quality of the zoomed images generated from this method is approximately the same of the original images, by securing the details and avoiding the loss of data. After the approximated vectors, a specific algorithm is used for improving the quality of the zoomed image. This algorithm works in an iterative way, and it is seen through the approximated vectors. It performs operations for rebuilding the blocks of the original images, by using the data of the approximate vectors found. Authors at the end, give conclusions for the perspective for work on the future in the field of zooming images and improving the existing methods.

According to another author [11] a representation is given of a method of discrete scaling based on the operator theory. Authors show a new method that is based on using discrete operators for zooming in and out images by keeping the data and the main characteristics of the original image. They present an innovative method that allows zooming signals in a discrete way by using a matrix acquisition. It begins with examining the existing methods of discrete zooming signals and the challenges that are encountered. After identifying the challenges, a new method is proposed that used the operator theory for zooming signals in a discrete way. The proposed method is about the matrix acquisition using a transforming matrix. This matrix acquisition allows zooming the signal using specific mathematical operations that guarantee good results in the aspect of the quality of the zoomed signal.

On the paper a mathematical model is used to describe the acquisition operations for it to describe changes on the structure and image details. After this, a discrete algorithm is used for implementing the acquisition using the operators identified on the first step. The results show the advantages of the proposed method in comparison with other existing methods of the acquisition of images. The method proposed provides a suitable result on keeping the data of the original image and keeping the most necessary characteristics of the image.

Through these experiments and other analyses performed, authors demonstrate that the proposed method offers other advantages comparing it with other existing methods of discrete zooming images. The proposed method assures correct zooming and good quality of the zoomed image using a good mathematical base. The paper presents a contribution on the field of image processing and offers an appropriate algorithm needed to complete a discrete acquisition of the images with a high quality.

On their results, authors show that their method offers an improvement on the quality of the zoomed images comparing it with other techniques. Usage of the approximate vector allows keeping more details of the original image, producing a better and more clear result for the human eye. Also, authors evaluate the sensitivity of their method for most of the details, using specific indexes of their performance. Compared with the existing techniques, their method shows a better sensibility, keeping more precisely the structures and details of the original image.

Concerning the duration of the process, authors show that their method is effective and can be used in real time. Using the techniques of the approximate vector allows a quicker processing of the images, offering a practical solution for zooming images on different applications. This paper presents a contribution to the field of image processing and offers an efficient method for zooming images with high quality and a low duration of the processing. For summarizing the paper, it describes the main steps of the method and the results achieved compared with the other existing methods.

Research reported in [12] presents an algorithm completely automatic for adaptive zooming for the colored images. On the paper, authors review the challenges of zooming colored images and the disadvantages of the existing methods. Then they present an appropriate algorithm that uses a combination of the techniques that determine zooming and analyzing the context for zooming in an automatic way the colored images on a single scan.

The proposed method uses specialized algorithms to identify the details and the context of the images and to define what parts of the image need zooming. The usage of the adaptive techniques and analyzing the context assures that the algorithm determines the necessary zooming for every part of the image in an automatic way. On their paper authors present a new method for an adaptive zooming of these colored images. The proposed method works on a single scan, and it is completely automatic, making it suitable for different applications that are looking for a quick and efficient zoom.

An analysis of the image is used for discovering the main characteristics of the image and identifying which are the parts that need zooming. This preliminary research makes the usage of the spectral and topological data of the image to define the details and the boundaries of different objects. After that, a specific algorithm is used for identifying the key factors that need to be considered for zooming the image. This algorithm comes from the analysis done and uses filtering and evaluating techniques to determine the most important factors for every pixel of the image. Also, an adaptive zooming process is used for fixing and adapting the fixing levels for every pixel of the image. This is done by using the values that are determined from the algorithm by applying a transformation that it needs for achieving the final zoom of the image.

At the end, authors present their results by comparing their method with those existing methods. They use a set of data images for testing the performance of their method in some respects like the image quality, precision, and duration of the process. Authors value the performance of their method by analyzing the criteria of the image quality, including the level of the details, contrast, visual perception, and sensitivity from the image deformation.

On their results, authors show that the proposed algorithm offers a clear improvement on the zoomed image quality compared with other existing techniques. Their method of adaptive zoom works in a single scan and offers a good compromise between creating details and not losing the contrast of the image. This results in zoomed images with more details and better visual perception. Also, authors display that their method is efficient from the perspective of the duration of processing. Using a single scan allows the algorithm to work in a more efficient way, by reducing the time needed to zoom the images.

Based on the research reported in [13], it shows a method of zooming images by using directional cubic convolution interpolation. The method that they propose secures a way for increasing the resolution of the image by not losing details and minimizing the effects of changing the view of the zoomed image.

On the paper, authors explain the problem of zooming images and the challenges of the interpolation existing methods. They display a new method that combines cubic convolution with analyzing the direction of the text of the image, this technique allows precise zooming of the detail of the font and improves the quality of the zoomed images. The proposed method uses directional cubic convolution interpolation, by having on account the direction of the font on these images. This convolution is convenient for zooming the details and suitable for the objects of the text. Usage of directional text during the interpolation gives a clearer and better result for visual perception.

An analysis of the direction of the points of the images is used for determining the main direction of the image structures. This direction is important for being used later the processing the interpolation direction. Then the directional cubic convolution interpolation is used for enlarging the image. This process does a directional move onto the direction of the points, by applying the cubic convolution onto the values of the points to find the values of the zoomed points. This directional interpolation helps not to lose the details of the image and improves the quality of the zoomed result. The results are then compared with the results of the existing techniques. They use a set of image data for testing the performance of their method on the aspects of image quality, precision and not losing the details.

Additional research [14] has as an objective to analyze the methods and the main developments on the field of deep learning for image super resolution. Authors analyze methods that use deep learning and mainly the techniques of the deep neural networks, to address the problems that are faced during super zooming of the images. They describe the development and the progress of these methods by reviewing the main aspects of the architecture of these deep networks, deep learning algorithm and other techniques that bring these techniques on these results.

Methodology on the paper shows a deep review of the literature, by analyzing the scientific papers and other work that address all the challenges of the super resolution of the images using deep learning. Authors identify and classify different methods used in this field, by grouping them on deep convolutional networks (CNN), neural networks and deep architecture with residual (ResNet).

On the paper, authors analyze challenges and problems that they have faced with the traditional methods of zooming images and how deep learning has offered a good solution for these challenges. They consider every architecture and the models of deep learning that are used for zooming images and describe the training process and the usage of these models.

Through their paper, authors put on focus advantages and disadvantages of these methods on super resolution of the images, by considering parameters like the performance of the model, speed of the process and the way of choosing these parameters. They also show different actual trends and the discovering being made on deep learning for the super resolution images. The paper presents an important contribution to the field of processing images and deep learning.

Furthermore, research presented in [15] reports a method for super-resolution by using discrete cosine transform (DCT) with local binary patterns (LBP) like a characteristic pattern. Authors propose a method based of discrete cosine transform for realizing image super-resolution. They use LBP like a characteristic model to determine colors and texture of an image. The proposed method uses a division process of an image into blocks and zooms every block with DCT.

They propose a method based on the usage of DCT and LBP, by combining the benefits of these two techniques by generating zoomed images with high quality and improved details. The proposed method uses DCT for determining the coefficient of the high frequency of the zoomed images, and on the other hand LBP is used as a characteristic pattern to determine the texture and the details of an image. The combination of these two techniques ensures improved quality and better details of the zoomed images.

The given image is divided into small blocks and LBP is used for determining the characteristics of the texture for every block. Then, the transformation DCT is done for identifying the changes made on the frequency and for zooming the image. At the end, the process of composing for not losing the edges and for generating the final zoomed image.

A scheme called Error-Amended Sharp Edge (EASE) for zooming the images is presented in [16]. Authors present an innovative scheme for zooming images that uses an access based on the sharp edges of the images. The scheme EASE combines a method called Directional Interpolation (DBI) with a correction process of the errors and improving the quality of the results of the zoomed images.

It explains the challenges and the problems that have encountered during the process of zooming images, by emphasizing problems of error-amended edges and the weaknesses of the quality of the zoomed images. They propose a method based on a detailed review of the structure of the original image and the application of the necessary changes made for improving the quality of the zoomed images. Method EASE combines two key steps which are processing of the edges and improving the quality. On the first step, authors evaluate and reprocess the data of the edges of the original image to identify the edges of the objects and the important structures. After this step, they use adaptive scaling that is used for changing values of the pixel and for achieving an image with an improved quality.

Firstly, DBI is used for zooming the image by having on considerate the sharp edges. Adter that, a correction error algorithm is used for fixing and improving the quality of the zoomed image. This algorithm identifies and corrects the possible errors made from the zoomed process, especially those on sharp edge of the images, by ensuring a better result. Authors present their results by comparing scheme EASE with other zooming techniques. The results show that scheme EASE offers an improvement in the quality of the zoomed image, especially those on the sharp edge and onto the details of the image.

Authors of the research reported in [17] present a general study that reviews the techniques used for retargeting the images. Authors analyze and review different techniques used for changing the size and the form of the image, a process that is known as retargeting images. Retargeting the images is a field that is widely studied on the processing of images and has a lot of applications on different fields like for example the visualization of the images on different devices and adapting the content of the images for the mobile devices.

Their objective is to analyze the existing methods of re-orientation and offer a general view of them. The paper starts with the explanation of the need for new re-orientation techniques and the challenges that are related to the changes of their size. They classify the methods of reorientation into different categories and investigate them in the most detailed way. The methods that have been reviewed include techniques based on the transformation of the texture, gambling and losing data, the segmentation of objects and distributing energy. Authors present the benefits and restrictions of every single method and evaluate them based on their performance and keeping all the details, perceptual suitability, and visual results.

The methodology on the paper includes a detailed analysis of the techniques used for retargeting images. Authors classify the methods that are used in different categories like geometric transformation, energy optimizing, and the methods based on the analysis of the content of the image. They present technical details and describe the advantages and the restrictions of every technique.

Methods are being compared in aspects like resistance to deformation, maintaining the important data, subjective perception, and the processing cost. The results and the achievements are used to present a general perspective on efficiency and application of different techniques in image retargeting.

Furthermore, a fast and correct method is presented for super resolution of red infrared images [18]. Authors present a method based on the mechanisms of zooming for improving the quality of the red infrared images. This method uses a fast and correct strategy for achieving a super resolution image by using the data detected from the zooming mechanism.

The zooming mechanism is used for generating a zoomed image based on the disposable information. After this an optimizing process is used for improving the quality of the zoomed image and for adapting the details of the red infrared image. The results are presented by comparing the results of their proposed method with other techniques of zooming the red infrared image.

They present a method that used the zooming mechanism and intercommunion between pixels of the image for improving the quality of the infrared red images. Method includes two key steps which are base zooming and improving the resolution.

At the first step, authors use a zooming mechanism that uses data of the original image to improve the size of the red infrared image. After that, they developed a fast algorithm to improve the resolution by using interaction between pixels and spectral information.

A fuzzy adapted algorithm for zooming images by using linear interpolation is also studied in [19]. The authors present an improved method for zooming images and the algorithm uses a fuzzy technique to determine the weights of the points in the process of the linear interpolation, by keeping in mind the level of importance and the influence on the zooming result.

The proposed method uses a combination of the linear interpolation algorithm and fuzzy techniques for achieving an improved result on the aspect of the quality of the zoomed images. On the paper authors explain the need for zooming techniques and present the challenges related to this process. They propose an algorithm based on linear interpolation that is being changed and adapted with the help of the techniques fuzzy. This algorithm allows the system to take all the detailed data from the original image and to adapt it to the context for generating better results on the aspect of zooming.

Method includes some key steps. Into the first step, authors discover the differences between the pixels of original image with those of the zoomed image to identify the parts with big changes. Then they use the fuzzy techniques to determine how to treat pixels on these places based on the certain fuzzy rules. This allows the algorithm to make adaptive decisions according to the detailed changes of the image. According to this method in the beginning a fuzzy matrix is calculated to represent the weights of the points of the original image. This process uses fuzzy techniques to classify the points into different categories with the importance base. After that, a linear interpolation is used by considering the fuzzy matrix to select the needed weights for the new points created during the zoomed process.

Based on the research from authors Danilo Costarelli et al. in [20], a study is presented that shows the difference between Kantorovich algorithm for digital image processing with some interpolation methods and quasi-interpolation. The Kantorovich algorithm is a method based on the transport theory, which can be used to find a function that minimizes the differences between two images.

They focus on the Kantorovich algorithm which is based on the theory of transport and has as an objective to improve the quality of the images through the model of distribution of the intensity of pixels. Authors develop a comparison between Kantorovich algorithm and other interpolation methods like linear interpolation, spline interpolation and other similar methods with interpolation. They evaluate the performance of every method on the aspect of the quality of the zoomed image, cleaning noises and the time of execution.

The methodology on the paper includes the steps of comparing and evaluating the performance of the Kantorovich algorithm with other methods of interpolation and quasi-interpolation. Authors use a set of data of the images to test the performance of their algorithm on the parameters like the quality of the generated image, objective perception, and stability of the algorithm on the cases when the parameters change. From the experiments that were done with different images, authors highlight the advantages and disadvantages of Kantorovich algorithm compared with other interpolation methods. They conclude that the Kantorovich algorithm offers a better performance on the aspects of processing digital images, by improving and keeping all the details of the image.

According to authors Pierre-Yves Laffont et al. [21] a method is presented for interactive zooming of images by using the content information. Authors present an advanced algorithm for zooming images, the content, and the details of the image for ensuring the wanted result. The proposed method is based on an extended effort for determining the important parts of an image and regulating zoom in accordance with these parts.

First a technique for identifying and determining the important part of the image is used like, for example, the faces of people or the main objects. After this, an algorithm is used for enlarging the image by taking on account these important parts and ensuring an appropriate zooming. On the paper, authors propose an innovative technique that allows the user to enlarge the details of an image by not losing the important parts and by avoiding unnecessary distortions.

The paper starts with presenting the challenges that occur during the zooming process of the images and the need for a method that offers interactive control and the important content on the results of zooming. Authors explain that most of the existing methods have difficulty in keeping all the details and avoiding unnecessary distortions, especially in the case when image includes a text, a face, or other important objects.

The method proposed, ‘Interactive Content-Aware Zooming’, uses a technique called ‘graph-cut-base zooming’ to identify and treat the most important parts of the image. This technique uses a graph to present the structure of the image and uses an algorithm based on the detachment of the graph to identify and save the parts that have the most important information.

An important aspect of this method is interactivity, which allows the user to select the parts of the image that he wants to save with all the details during zooming. Users can use a simple interface to select and determine the important pixels. Method then uses the information given from the user by achieving results with good details and with no unnecessary deformations.

CHAPTER 3

METHODOLOGY

3.1 Methods

3.1.1 Sinc-zoom theory

The Whittaker-Shannon interpolation formula and the Nyquist-Shannon sampling theorem set as theoretical basis of the technique. The Whittaker-Shannon interpolation formula is reported in equation (1), where the sequence $x[n]$ allows reconstruction in the domain of the time variable 't', T is the period, and the bandlimit $\frac{1}{2T}$ is measured in Hertz. The sampling frequency is $f_s = \frac{1}{T}$ and the Nyquist frequency is $\frac{f_s}{2}$ [22].

$$\boldsymbol{x(t) = \sum_{n=-\infty}^{\infty} x[nT] \cdot sinc\left(\frac{t - nT}{T}\right)} \qquad (1)$$

The Nyquist-Shannon sampling theorem (reported in equation (2)) states that "If a function $x(t)$ contains no frequencies higher than B hertz, it is completely determined by giving its ordinates at a series of points spaced $1/(2B)$ seconds apart [23]." Moreover, perfect reconstruction is guaranteed possible for $B < \frac{f_s}{2}$. With x_n the nth sample of the sequence, the function $x(t)$ is calculated as:

$$\boldsymbol{x(t) = \sum_{n=-\infty}^{\infty} x_n \cdot \frac{sin[\pi(2Bt - n)]}{[\pi(2Bt - n)]}} \qquad (2)$$

The Nyquist-Shannon sampling theorem is implemented in equation (3).

$$\boldsymbol{I(i,j) = \sum_{s=1}^{N_x}\sum_{p=1}^{N_y} I(p,s) \cdot Sinc_x \cdot Sinc_y} \qquad (3)$$

Where equations (4) and (5) calculate the sinc functions

$$\boldsymbol{Sinc_x = Sinc\left\{\pi\left[2.0 \cdot B \cdot \left(p - \frac{N_x}{2}\right) - S \cdot \left(j - \frac{N_x}{2}\right)\right]\right\}} \qquad (4)$$

$$\boldsymbol{Sinc_y = Sinc\left\{\pi\left[2.0 \cdot B \cdot \left(s - \frac{N_y}{2}\right) - S \cdot \left(i - \frac{N_y}{2}\right)\right]\right\}} \qquad (5)$$

The rectangular image space region R_{xy} that determines the sampling interval and so the magnitude of the zoom is:

$$\boldsymbol{R_{xy} = R_x \cdot R_y} \qquad (6)$$

$$\boldsymbol{R_x = \left[2.0 \cdot B \cdot \left(p - \frac{N_x}{2}\right) - S \cdot \left(j - \frac{N_x}{2}\right)\right]} \qquad (7)$$

$$\boldsymbol{R_y = \left[2.0 \cdot B \cdot \left(s - \frac{N_y}{2}\right) - S \cdot \left(i - \frac{N_y}{2}\right)\right]} \qquad (8)$$

3.1.2 Code Translation from C/C++ to pure C programming language

This is a Sinc zoom application [24] which is written in C/C++ programming language and will be translated in pure C language.

```
#include <iostream>
#include <fstream>
#include <string>
#include <io.h>
#include <dos.h>
#include <conio.h>
#include <stdlib.h>
#include <sstream>
#include <stdio.h>
#include <iomanip>
#include <istream>
#include <math.h>

using namespace std;

void OnZoom_Sinc(char imageFileName[], int rcxres, int rcyres, double m_Bandwidth,
double m_TheSamplingRate);
void OnFourierTransform(char imageFilename[], int rcxres, int rcyres);

class Sinc_Int_2022 {


      int n1;
      int n2;
```

Figure 3.Removing unnecessary libraries

In Figure 3 we have all the libraries that are used both in C++ and C. There are two functions that are declared which are OnZoom_Sinc() and OnFourierTransform().These functions are also used in pure C. In C++ we have a class Sinc_Int_2022 declared and it has got two variables int n1 and int n2.

```
struct data {

            double** Signal;

            double** Edge_X;

            double** Edge_Y;

            double** Region_XY;

      }*pointer;
```

Figure 4. Pointers in C++

In C++, ‘pointer’ is a pointer to an object of type ‘struct data’ in Figure 4. So the pointer is pointer to the element of the structure ‘data, which points to the memory address.

```
#define _CRT_SECURE_NO_DEPRECATE

#include <stdlib.h>
#include <math.h>
#include <stdio.h>


#define endl "\n"

void OnZoom_Sinc(char imageFileName[], int rcxres, int rcyres, double
m_Bandwidth, double m_TheSamplingRate);
void OnFourierTransform(char imageFilename[], int rcxres, int rcyres);

struct data {
      double** Signal;
      double** Edge_X;
      double** Edge_Y;
      double** Region_XY;
};

//typedef struct Sinc_Int_2022 {
struct Sinc_Int_2022 {
      int n1;
      int n2;
      struct data* pointer;
};
```

Figure 5.Libraries in C and no ‘public’ keyword

The changes in Figure 5 were made to remove the C++ specific syntax and libraries from the code and convert it to pure C language syntax. The libraries that are specific for C++ such as <iostream>, <fsream>, <string>, <iomanip> etc. were removed and replaced with equivalent C libraries. In C++, the keyword ‘class’ is used to define a class, whereas in C language there is no such keyword. The class body contains the member variables and member functions of the class, which are declared and defined inside the class. In C language, a structure can be used to a group of variables and functions, but the functions must be defined outside the structure. Replace '*pointer' with 'struct data* pointer'. In C++, 'pointer' is a pointer to an object of type 'struct data'. In pure C, we need to specify the type of the pointer by using 'struct data* pointer' instead of '*pointer'.

```
public:

       struct data {

              double** Signal;

              double** Edge_X;

              double** Edge_Y;

              double** Region_XY;

       }*pointer;

public:

       Sinc_Int_2022(int x, int y) : n1(x), n2(y) { pointer = 0; };
       void allocateData();
       void save();
       ~Sinc_Int_2022() { }

};

void Sinc_Int_2022::allocateData() {

       pointer = new data;

       pointer->Signal = new double* [this->n2];
```

Figure 6.Classes, pointers and declaring functions inside the class

This code in Figure 6 defines a C++ class named 'Sinc_Int_2022' that contains a structure named 'data'. The 'data' structure includes four double pointer members, 'Signal', 'Edge_X', 'Edge_Y', and 'Region_XY', which are used to represent a matrix or an image.

The class 'Sinc_Int_2022' has a constructor that takes two integer arguments 'x' and 'y'. The constructor initializes two private variables 'n1' and 'n2' with the values of 'x' and 'y', respectively. It also sets the 'pointer' member variable of the 'data' structure to 0 (null pointer).

The class ‘Sinc_Int_2022’ has two public functions, ‘allocateData()‘ and ‘save()’. ‘allocateData()’ is a method of the class that allocates memory for the ‘Signal’, ‘Edge_X’, ‘Edge_Y’, and ‘Region_XY’ members of the ‘data’ structure using nested ’for’ loops. ‘save()’ is another method that saves data to a file.

```
void allocateData(Sinc_Int_2022* sinc) {
	sinc->pointer = (struct data*)malloc(sizeof(struct data));
	sinc->pointer->Signal = (double**)malloc(sinc->n2 * sizeof(double*));
	for (int v = 0; v < sinc->n2; v++) {
		sinc->pointer->Signal[v] = (double*)malloc(sinc->n1 * 
sizeof(double));
	}
	for (int v = 0; v < sinc->n2; v++) {
		for (int f = 0; f < sinc->n1; f++) {
			sinc->pointer->Signal[v][f] = (double)0.0;
		}
	}
}
void save(Sinc_Int_2022* sinc) {
	FILE* savedata;
	char outputFile[128];
	sprintf(outputFile, "%s", "Signal.img");
	if ((savedata = fopen(outputFile, "wb")) == NULL)
	{
		printf("Cannot open output file, Now Exit...\n");
		exit(0);
	}
	else {
		for (int v = 0; v < sinc->n2; v++) {
			for (int f = 0; f < sinc->n1; f++)
				fwrite(&sinc->pointer->Signal[v][f], sizeof(double), 1, 
savedata);
		}
		fclose(savedata);
	}
}
```

Figure 7. Body of method allocateData() and method save()

In figure 7 in C, we would need to use ‘struct’ instead of ‘class’, and we would not have access specifiers such as ‘public:’ or ‘private:’.

- Remove constructor and destructor: In C++, classes can have constructors and destructors that are automatically called when an object is created. In pure C, we don't have classes, so we don't need constructors or destructors.

- Declare functions outside the struct: In C++, member functions of a class are declared inside the class using the ‘className::functionName()’ syntax. In pure C, we declare the functions outside the struct and use the 'structName.functionName()' syntax to call them.

- Replace 'malloc()' with 'new': In C++, the 'new' keyword is used to allocate memory dynamically, whereas in C we use 'malloc()' to allocate memory dynamically. Therefore, we need to replace all instances of 'new' with 'malloc()'.

```
std::cout << endl;
std::cout << "Please type the image file name" << endl;
std::cout << "Please make sure that the image format is Analyze 'double': 64 bits
real" << endl;
std::cout << "Please enter the following values: " << endl;
std::cout << "The number of pixels along the X direction (integer)" << endl;
std::cout << "The number of pixels along the Y direction (integer)" << endl;
std::cout << "The Bandwidth (double) in [0.1, 4.0]" << endl;
std::cout << "The Sampling Rate (double) in [0.1, 4.0]" << endl;
std::cout << endl;
```

Figure 8. Output in C++

In C++ we use cout with << for executing the program and printing the ouput. We can use as many cout as we need to (Figure 8). We have used endl because in this case we want to insert a new line between outputs.

```
printf("%s", endl);
printf("Please type the image file name%s", endl);
printf("Please make sure that the image format is Analyze 'double': 64 bits real%s",
endl);
printf("Please enter the following values: %s", endl);
printf("The number of pixels along the X direction (integer)%s", endl);
printf("The number of pixels along the Y direction (integer)%s", endl);
printf("The Bandwidth (double) in [0.1, 4.0]%s", endl);
printf("The Sampling Rate (double) in [0.1, 4.0]%s", endl);
```

Figure 9. Output in C

In pure C programming language (Figure 9) we use printf for printing an input instead of cout that we used in C++. Like in C++ we can use as many printf as we need and in order for the output to be inserted in a new line, we use endl but not with the operator << before it.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Sample Cases Study Results

Original and zoomed images are presented in Figure 17. The values of B and S were: 0.3 and 0.71 (zoom-out); 0.31 and 0.3 (zoom-in); 0.3 and 0.6 (no-zoom).

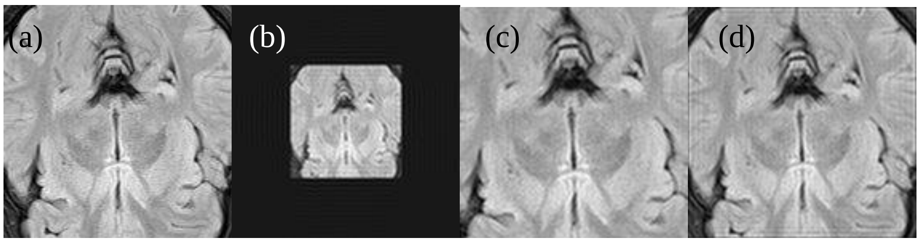


Figure 10. Sample images**.** From left to right: departing image (a), zoom-out (b), zoom-in (c), and no-zoom (d) images. Note the Gibbs effect in (b) by zoom-out and (d) by no-zoom. The image is free of artifacts when oversampling

The images presented in Figures 18 and 19 illustrate two of four cases that are possible in each of the scenarios of zoom-out (first column from left), zoom-in (second column from right), and no-zoom (right-most column). The map of $Sinc_x \cdot Sinc_y$ shows that the source of Gibbs effect for the two cases of zoom-out and no-zoom (see bottom rows).

Case 1. When $2.0 \cdot B \cdot \left(p - \frac{N_x}{2}\right) < S \cdot \left(j - \frac{N_x}{2}\right)$, and $2.0 \cdot B \cdot \left(s - \frac{N_y}{2}\right) < S \cdot \left(i - \frac{N_y}{2}\right)$, it follows from Equations (7) and (8), that the edge $R_x < 0$ and the edge $R_y < 0$. Figure 2 shows from left to right, the map of R_x (top row) and R_y (second row from the top), and R_{xy} (bottom row), for zoom-out, zoom-in, and no-zoom, respectively.

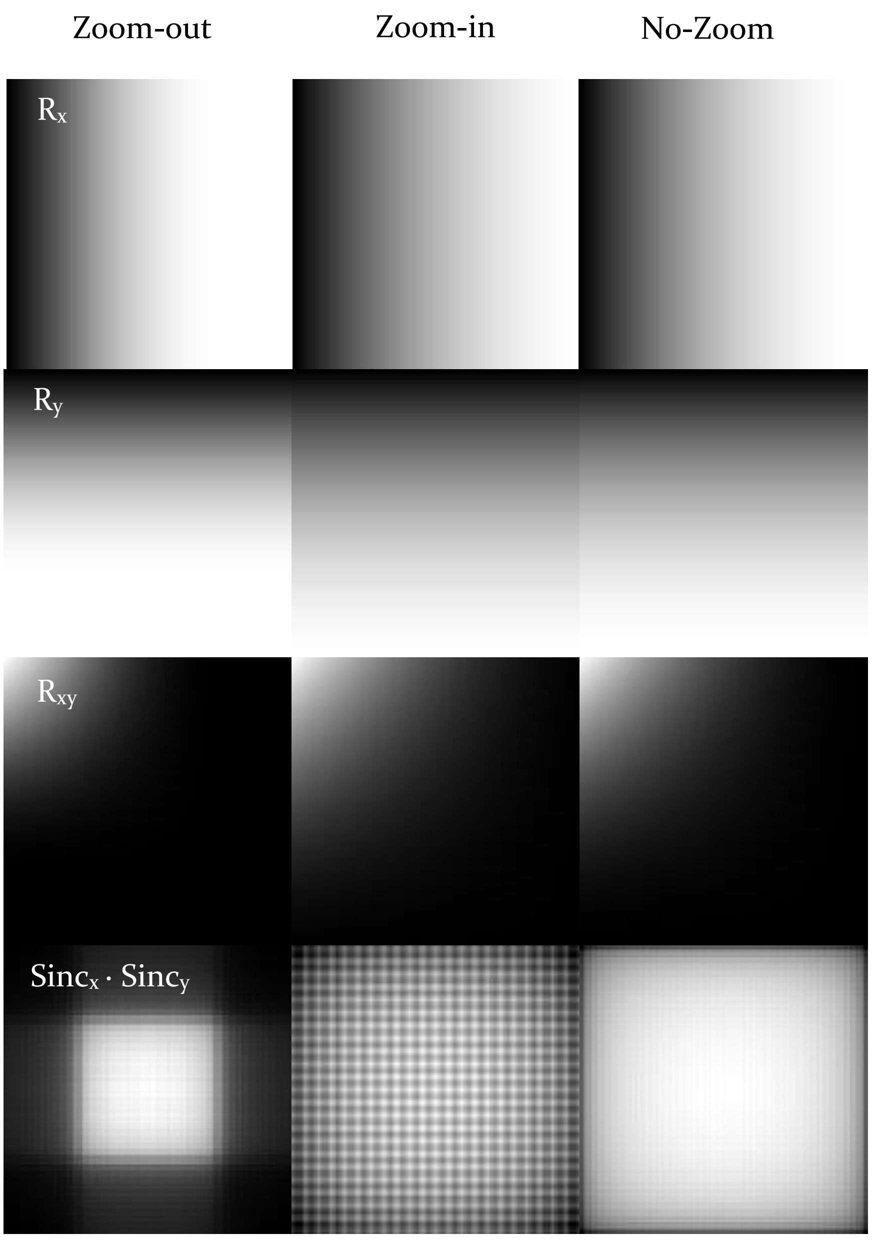


Figure 11. Case 1. For the three cases of zoom-out, zoom-in, and no-zoom, and from left to right the figure shows the following maps: edge $R_x < 0$ (in the top row), edge

$R_y < 0$ (in the second row from the top), region R_{xy} (in the third row from the top), $Sinc_x \cdot Sinc_y$ (in the bottom row).

Case 2. When $2.0 \cdot B \cdot \left(p - \frac{N_x}{2}\right) > S \cdot \left(j - \frac{N_x}{2}\right)$, and $2.0 \cdot B \cdot \left(s - \frac{N_y}{2}\right) < S \cdot \left(i - \frac{N_y}{2}\right)$, it follows from Equations (7) and (8), that the edge $R_x > 0$ and the edge $R_y < 0$. Figure 3 shows from left to right, the map of R_x (top row) and R_y (second row from the top), and R_{xy} (bottom row), for zoom-out, zoom-in, and no-zoom, respectively.

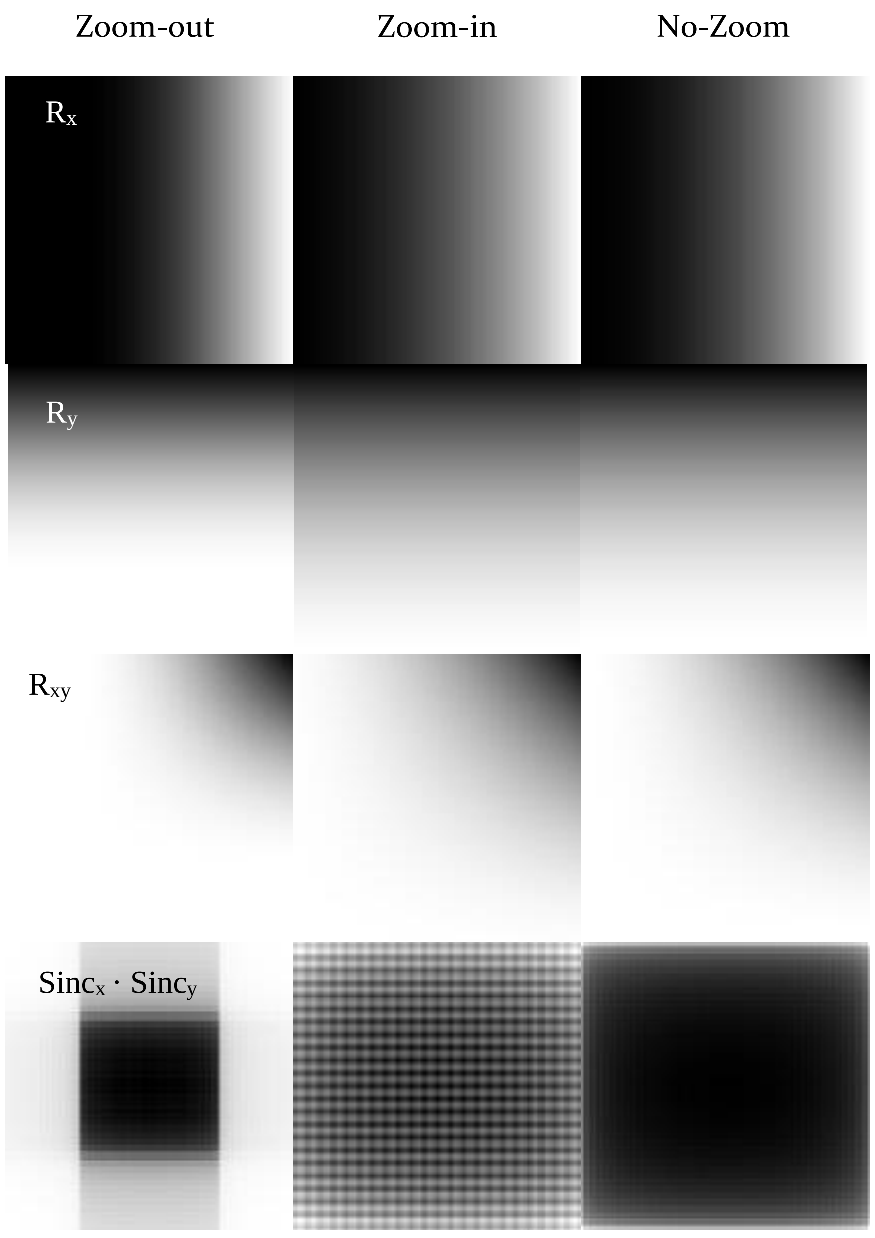


Figure 12. Case 2. For the three cases of zoom-out, zoom-in, and no-zoom, and from left to right the figure shows the following maps: edge $R_x > 0$ (in the top row), edge $R_y < 0$ (in the second row from the top), region R_{xy} (in the third row from the top), $Sinc_x \cdot Sinc_y$ (in the bottom row).

4.2 Sample Results Obtained Using Theoretical Images

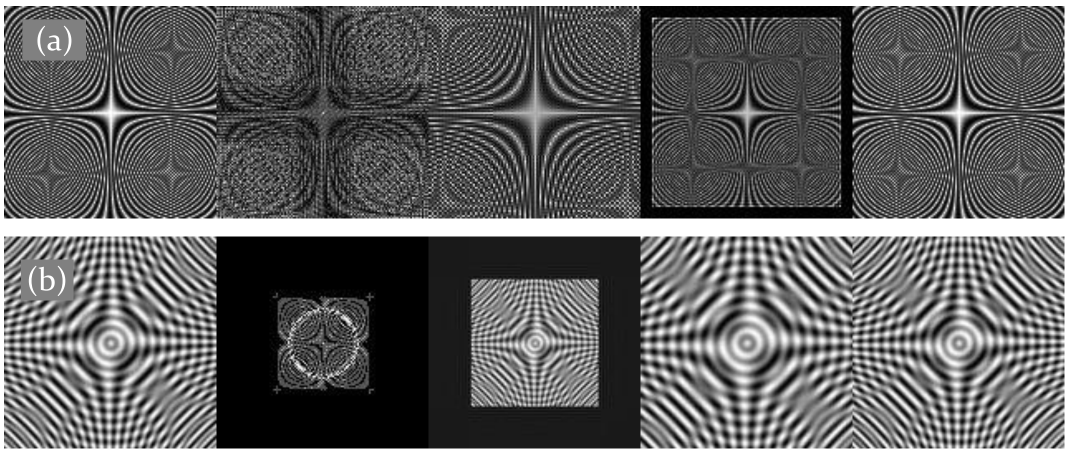
In Figure 13 there is the preliminary analysis of the images provided with a series of test images that are (a, b, c, d) on different zoom levels along with their corresponding pixel-local and k-space magnitudes. Starting from the left of (a) top row and then proceeding towards the right and then with the next rows (b, c, d) we observe the patterns as followed:

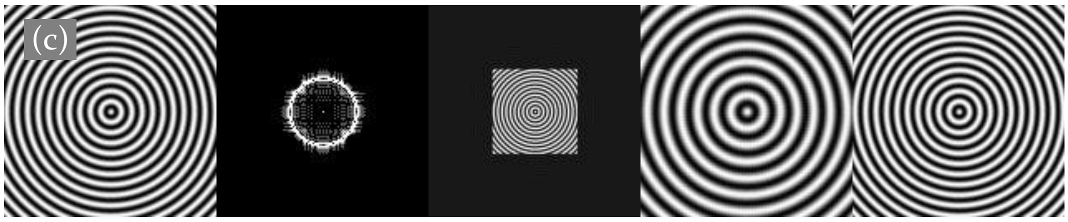
On test image (a) we have a 10% zoom-out, with a sampling rate of 1.8 spatial frequency units (sfu). On the right a 55% zoom-in is applied resulting a sampling rate of 1.3 sfu. The last image is the original image applying no zoom with a sampling rate of 2.0 sfu. On the second row with the test image (b) we start with a 40% zoom-out with a sampling rate of 1.2 sfu. Moving to the 20% zoomed-in image with a sampling rate of 2.4 sfu. Similar with the first row, the last image has no zoom applied with a sampling rate of 2.0 sfu. On the third row with the test image (c) we start with a 60% zoom-out with a sampling rate of 0.8 sfu. Moving to the 60% zoomed-in image with a sampling rate of 3.2 sfu. Similar with the other rows, the last image has no zoom applied maintaining a sampling rate of 2.0 sfu. On the last row, the test image (d) has an increased zoom-out percentage of 70% with a sampling rate of 0.6 sfu. Image is zoomed-in on 100%, but the reconstruction is unsatisfactory with a sampling rate of 4.0 sfu. The last grid portrays no zoom with a sampling rate of 2.0 sfu.

When the zoom-in effect is applied onto the image, it allows us to examine finer details. In Figure 6 we see different percentages of zoom-ins on different test images. In test image (a), a 55% zoom-in is applied, by enlarging the image and increasing the spatial resolution. When the percentage is increased, the sampling rate is increased too by having a potential degradation and a loss of image quality, which that is not a good case. When the percentage is low (b), with a 20% zoom in and with a low sfu, the zoom-in effect enhances the visibility of the details within this image.

Different from the zoom-in effect, zoom-out reduces the scale of the image, by portraying wider field of the view and resulting in loss of details. With the increased percentage of the zoom-out effect, the sampling rate is reduced. On test image (a) with a 10 % zoom-out effect and a sampling rate of 1.8 sfu, allows us for a broader view of the image and emphasizing the global features that are extracted from the overall image. With the decreased sampling rate, there is an emphasized overview of the image, but with a loss of detail and clarity.

The no-zoom images are the unaltered versions of the test images. The sampling rate that is maintained is 2.0 sfu by keeping the level of detail and the original scale that is on the original images. The no-zoomed images are used as a reference for comparing with the versions of zoom-in and zoom-out and to see the impact and the quality of the images of these applied effects. Zoom-in and zoom-out effects reveals different levels of detail or enlargement of large structures. They show the loss or enhancement of the details from image to image.





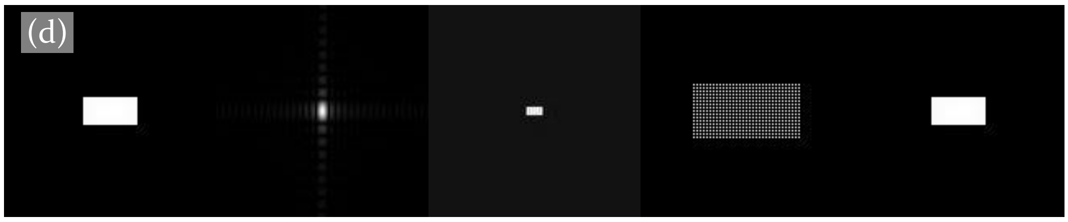
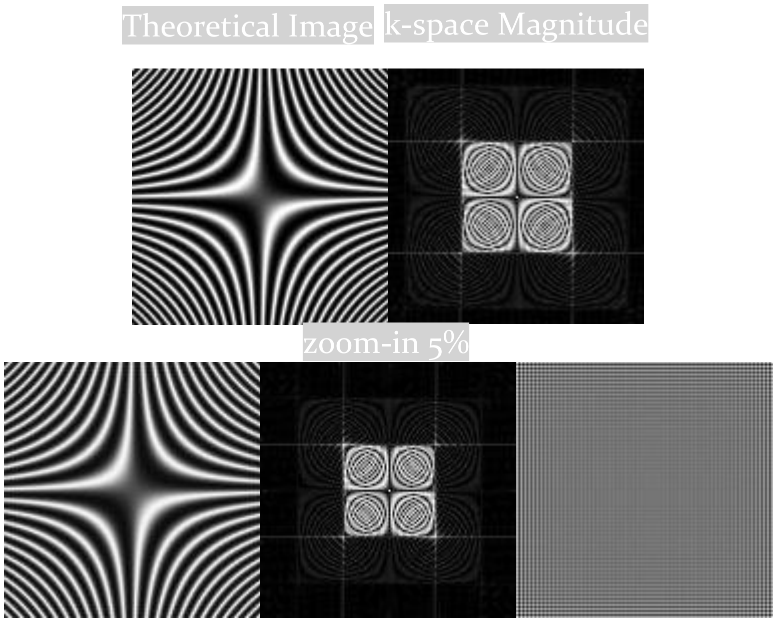


Figure 13. Preliminary analysis.

A theoretical image is used for zooming-in with four different percentage zooming with a k-space magnitude and map of the sinc function across the image grid (Figure 14). Second row: Zoom-in 5%. Third row: Zoom-in 7.5%, Fourth row: Zoom-in 10%. Fifth row: Zoom-in 15%. The percentages that are used are 5%, 7.5%, 10 % and 15 % where we can see that from figure to figure there is a gradually change of zooming-in, with the increase of the percentage factor. With zoom-in we get larger images by focusing on finer details.



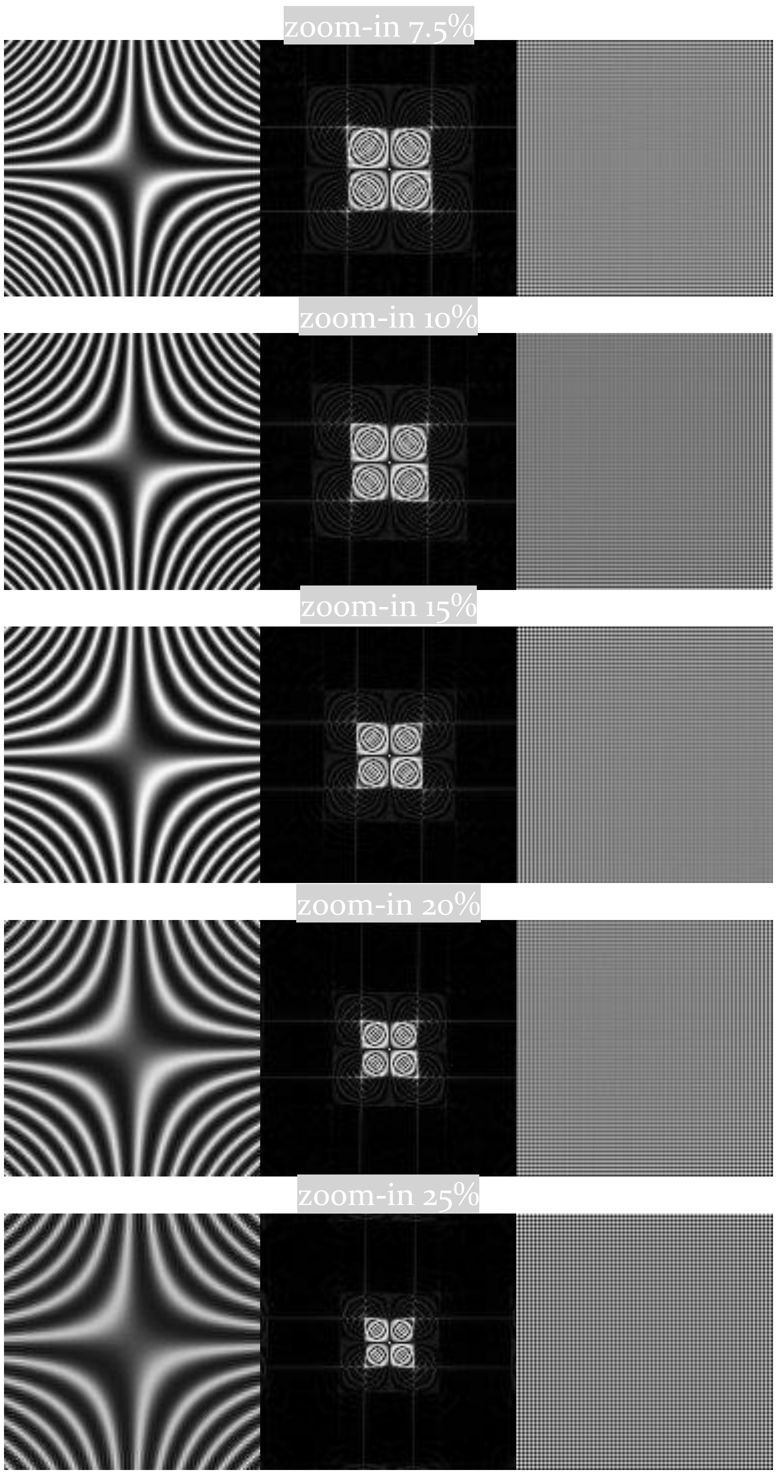
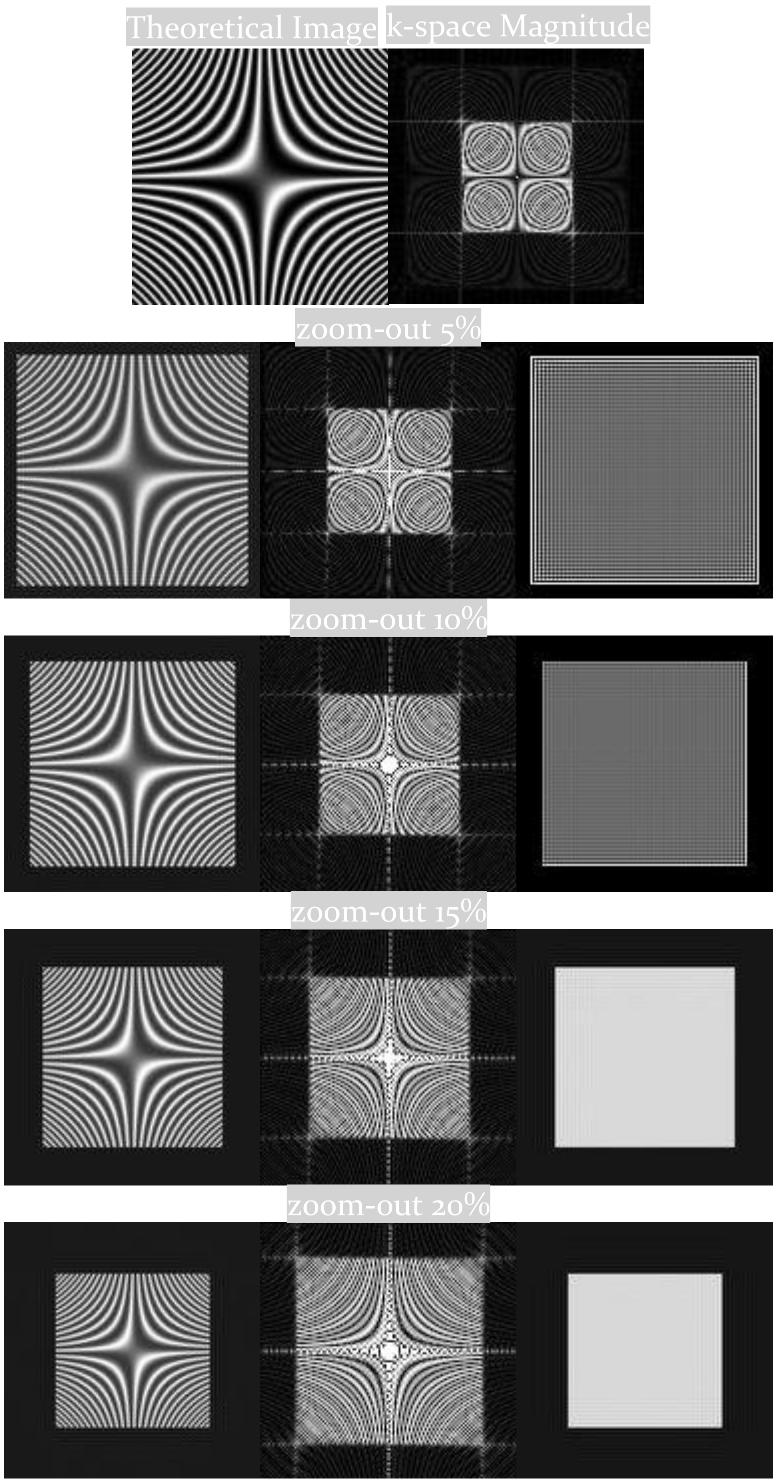


Figure 14. Non-Adaptive Sinc-Zoom-in. Top row: Theoretical image (left) and its k-space magnitude (right).

The same theoretical image was used (Figure 15), but on this case for zooming-out with the percentages: 5%, 10%, 15%, 20%, 25% with a k-space magnitude. Grids of the rows are enlarged from row to row with the percentage being increased.


Theoretical Image
k-space Magnitude
zoom-out 5%
zoom-out 10%
zoom-out 15%
zoom-out 20%

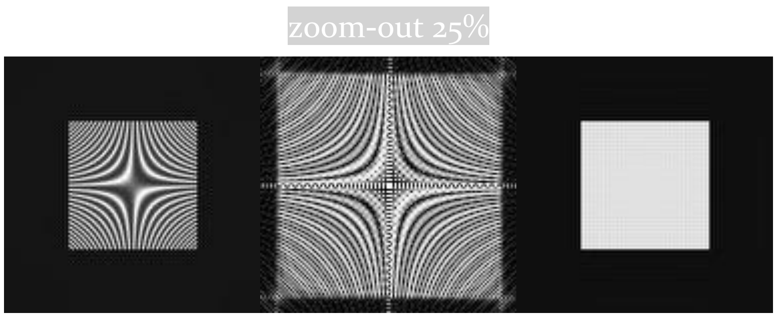


Figure 15. Non-Adaptive Sinc-Zoom-out. Top row: Theoretical image (left) and its k-space magnitude (right).

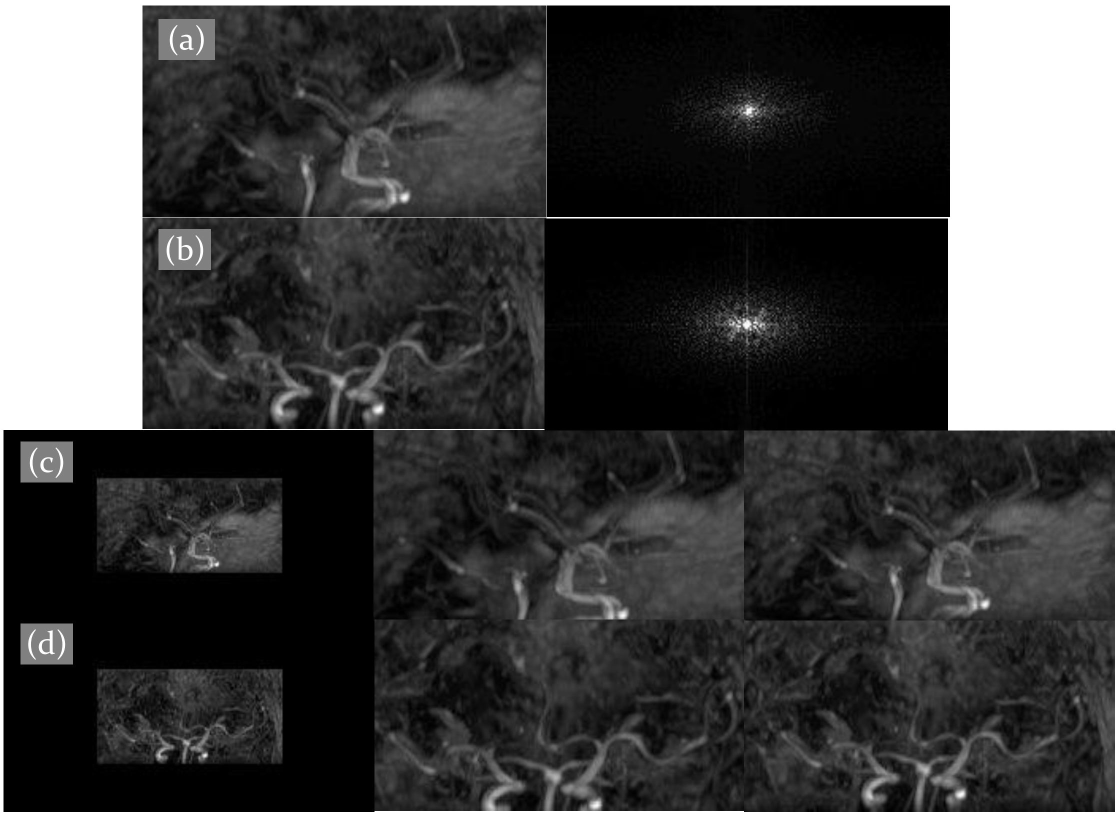
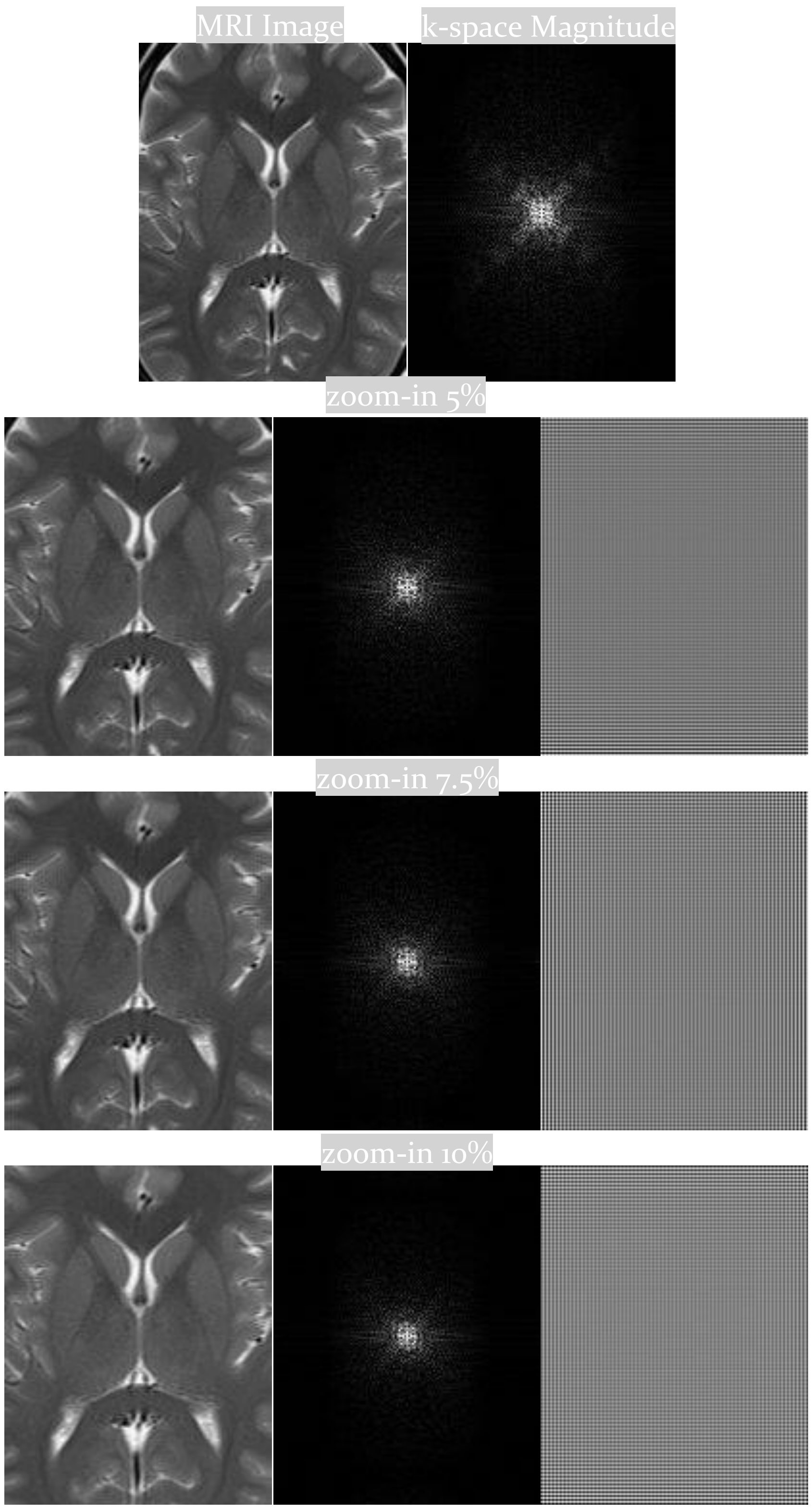


Figure 16. Sinc-based zooming on Magnetic Resonance Angiography images. Rows labelled with (a) and (b), show from left to right: the departing MRA, the k-space of the departing MRA. Rows labelled with (c) and (d), present the case of non-adaptive Sinc-based zooming and show from left to right: zoom-out MRA 50%, zoom-in MRA 15%, and no-zoom MRA.

4.3 Sample Results Obtained Using a Magnetic Resonance Image



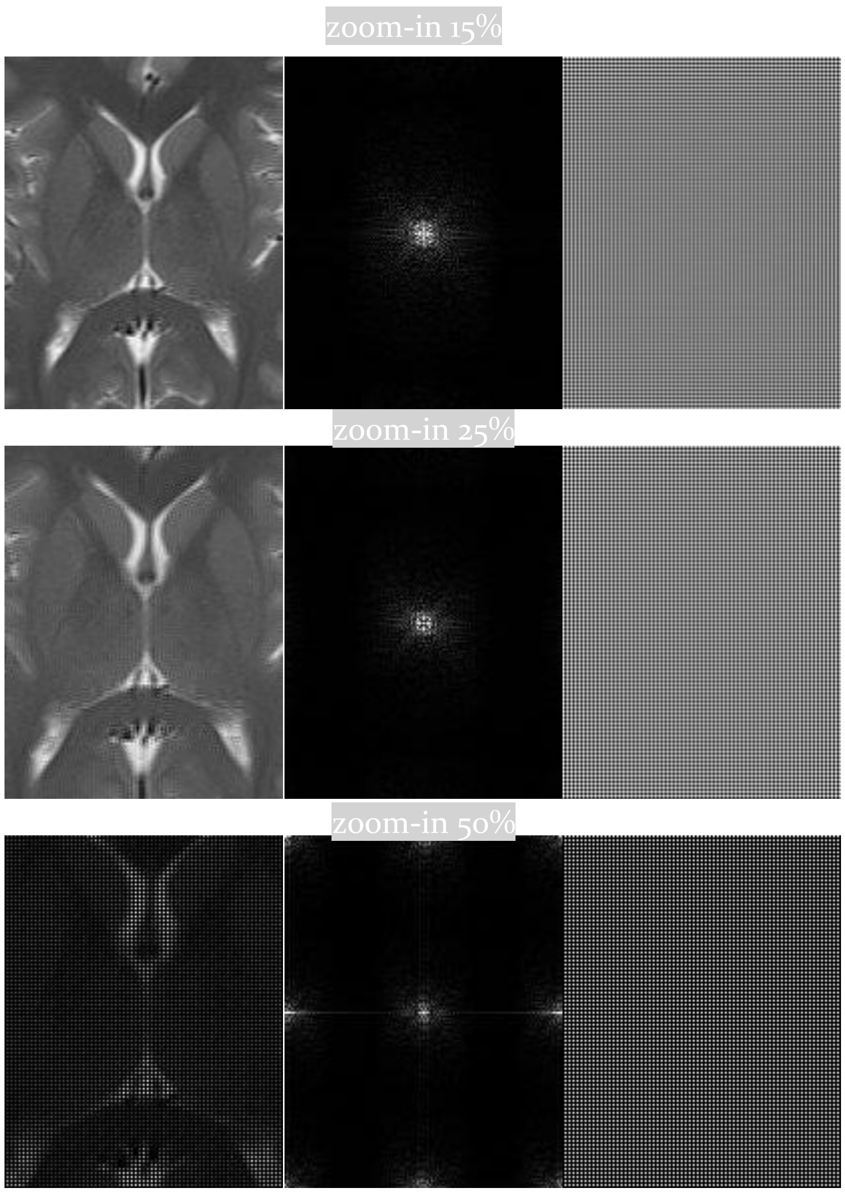
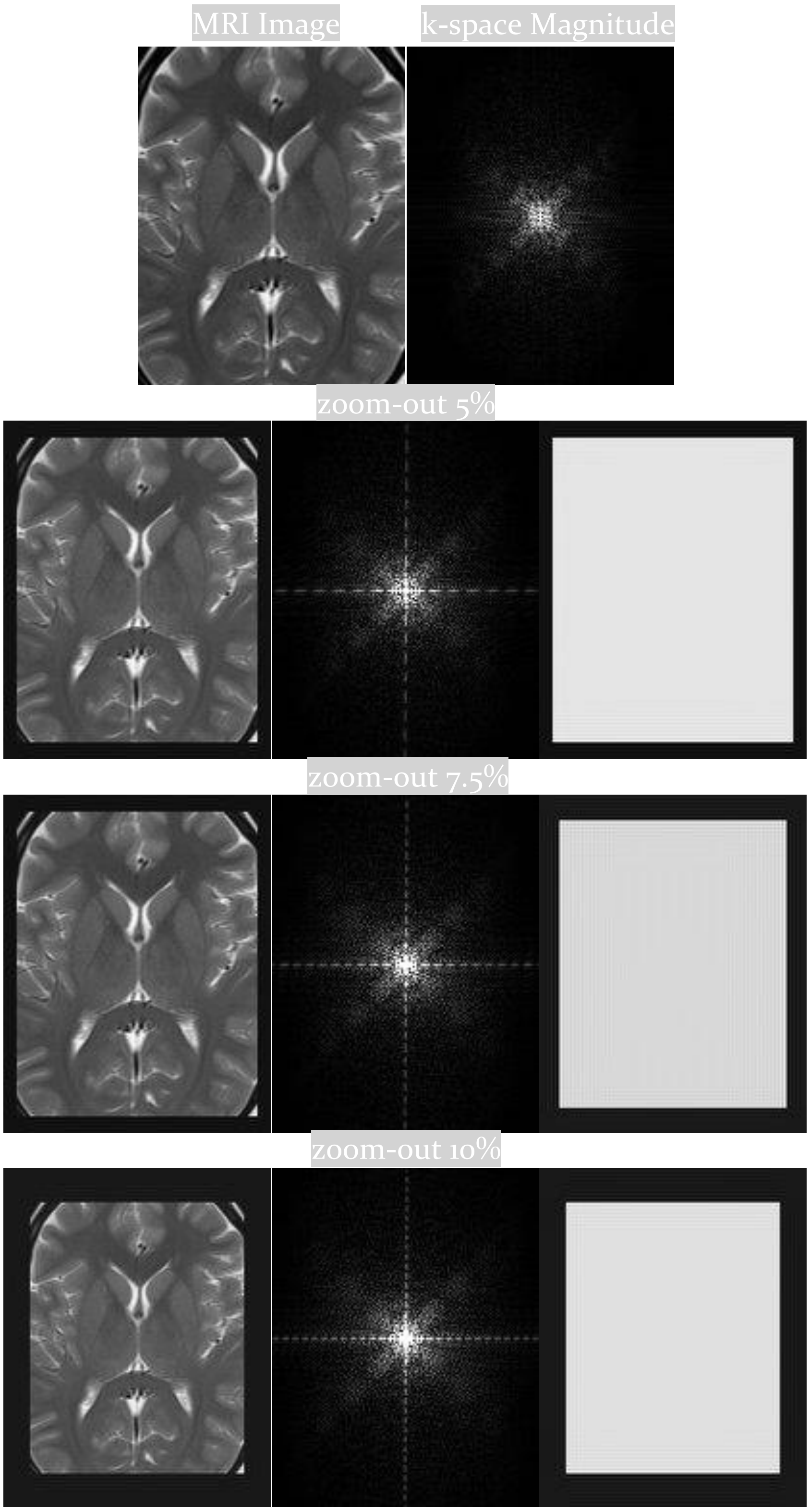


Figure 17. Non-Adaptive Sinc-Zoom-in. Top row: MRI (left) and its k-space magnitude (right).


MRI Image
k-space Magnitude
zoom-out 5%
zoom-out 7.5%
zoom-out 10%

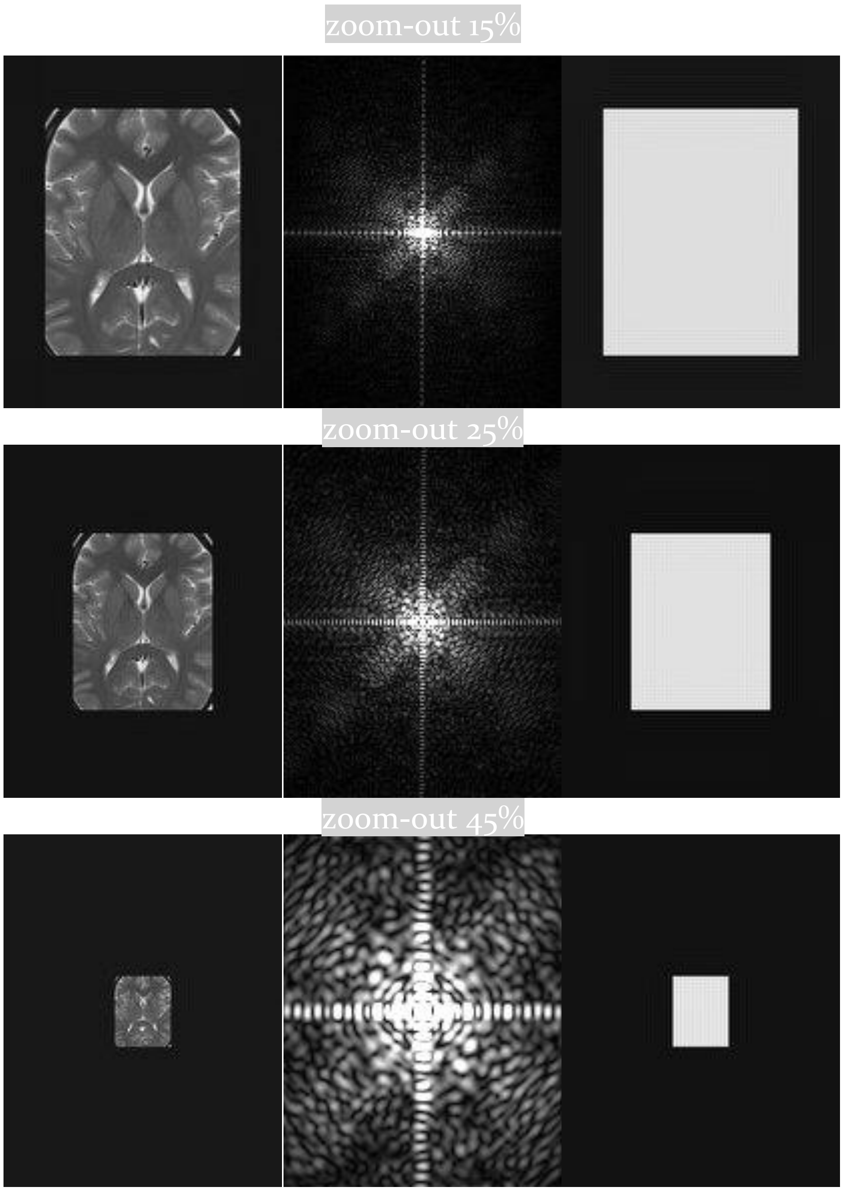
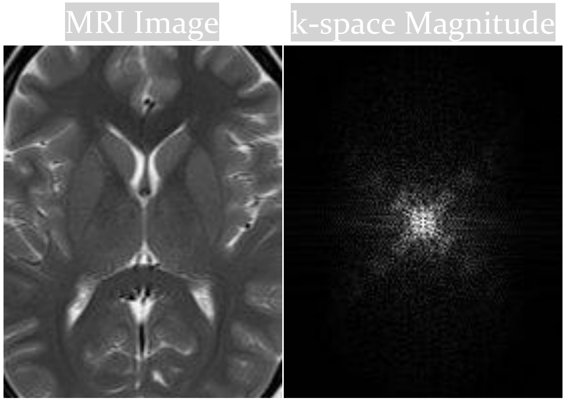
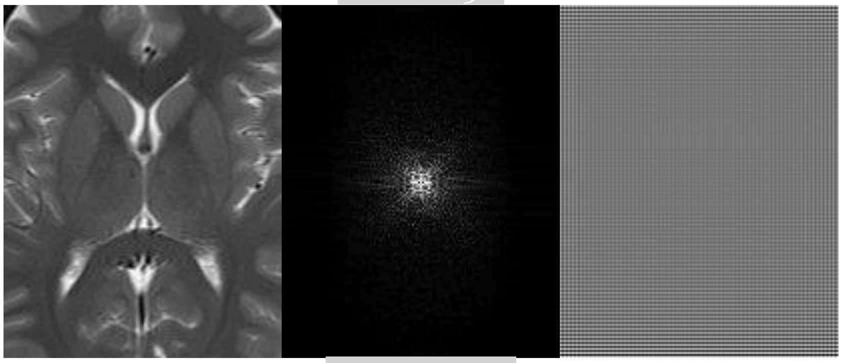


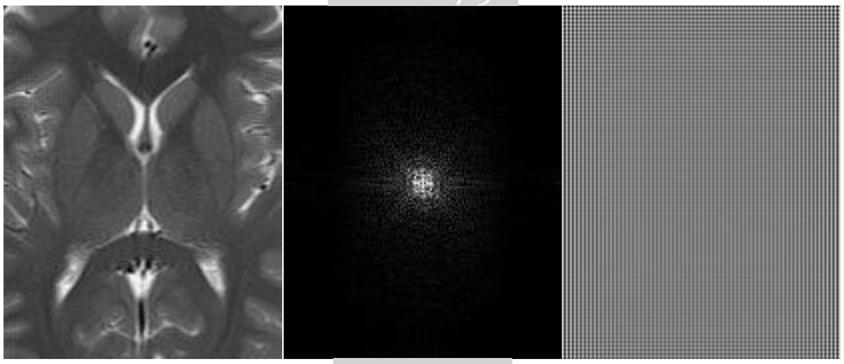
Figure 18. Non-Adaptive Sinc-Zoom-out. Top row: MRI image (left) and its k-space magnitude (right).



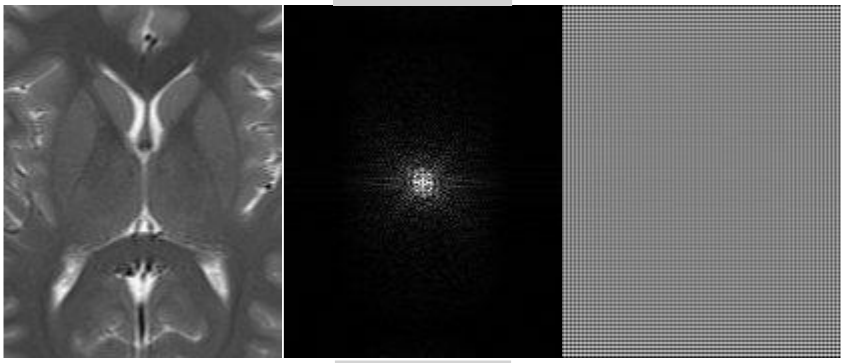
zoom-in 5%



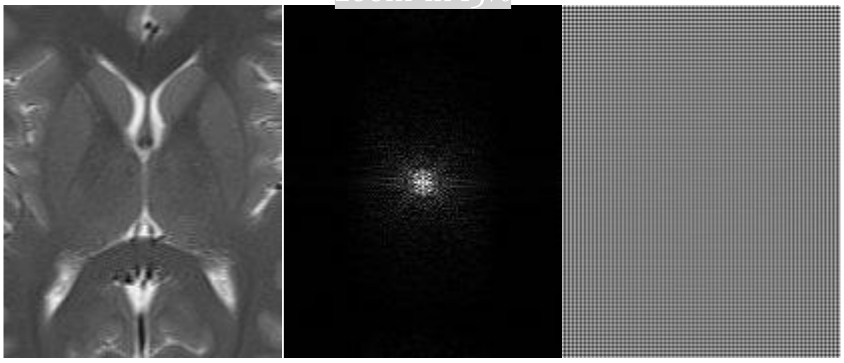
zoom-in 7.5%



zoom-in 10%



zoom-in 15%



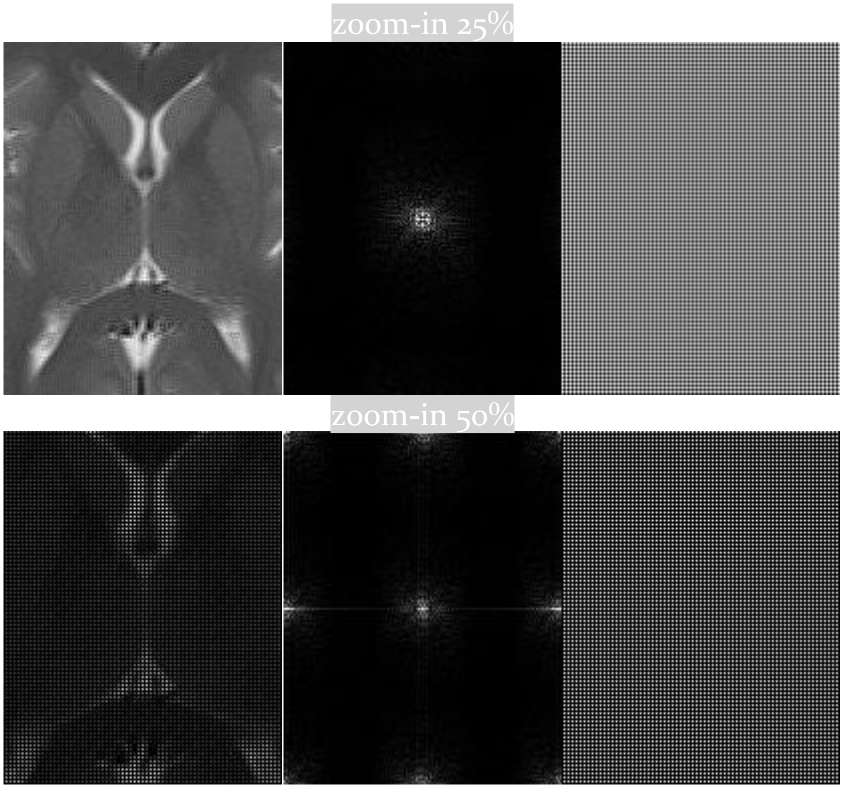
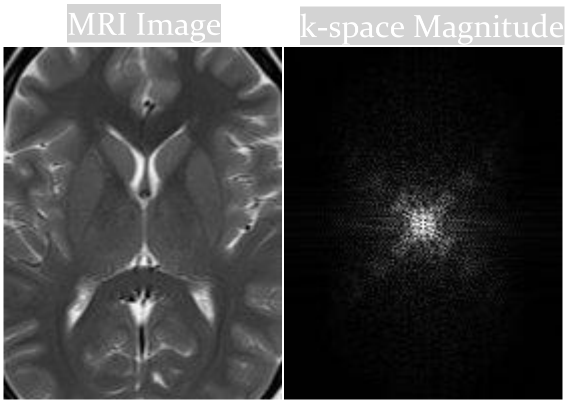
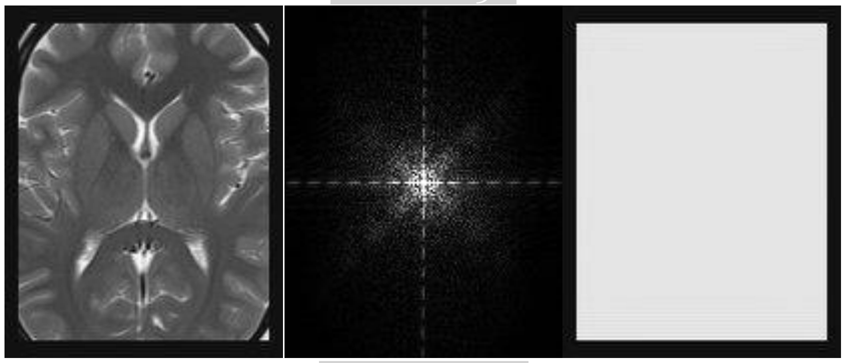


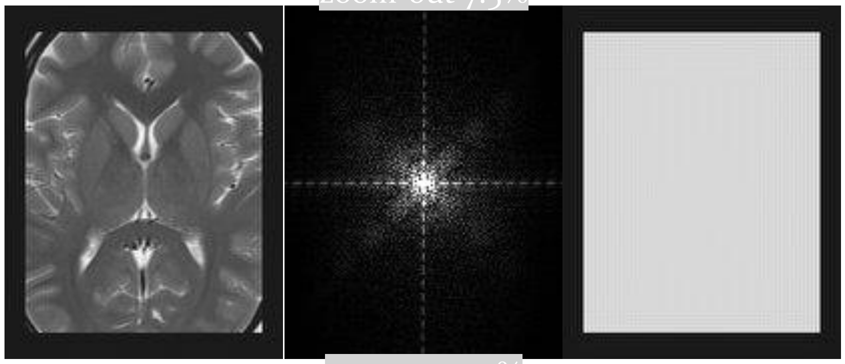
Figure 19. Adaptive Sinc-Zoom-in. Top row: MRI image (left) and its k-space magnitude (right).



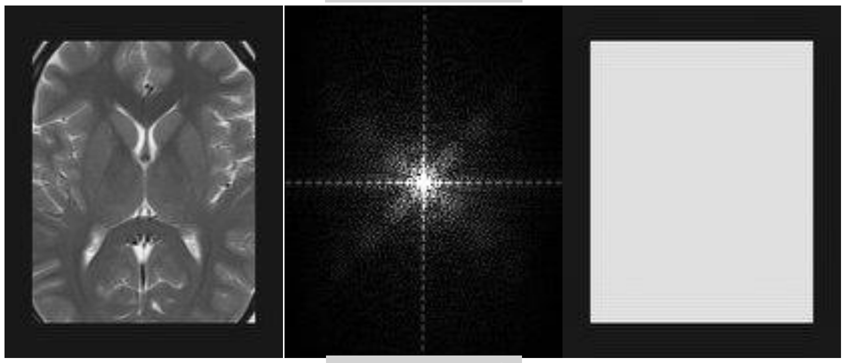
zoom-out 5%



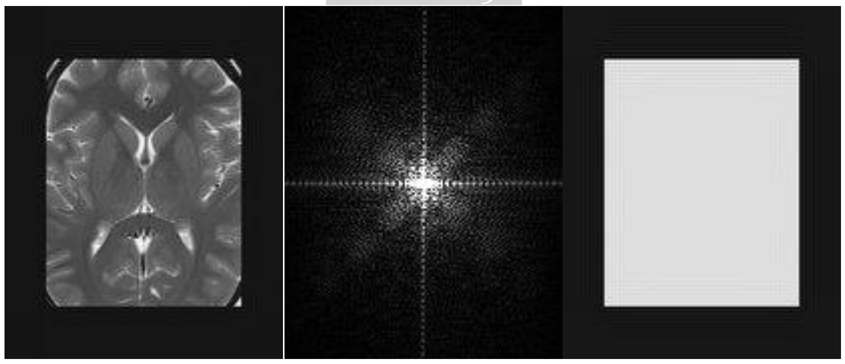
zoom-out 7.5%



zoom-out 10%



zoom-out 15%



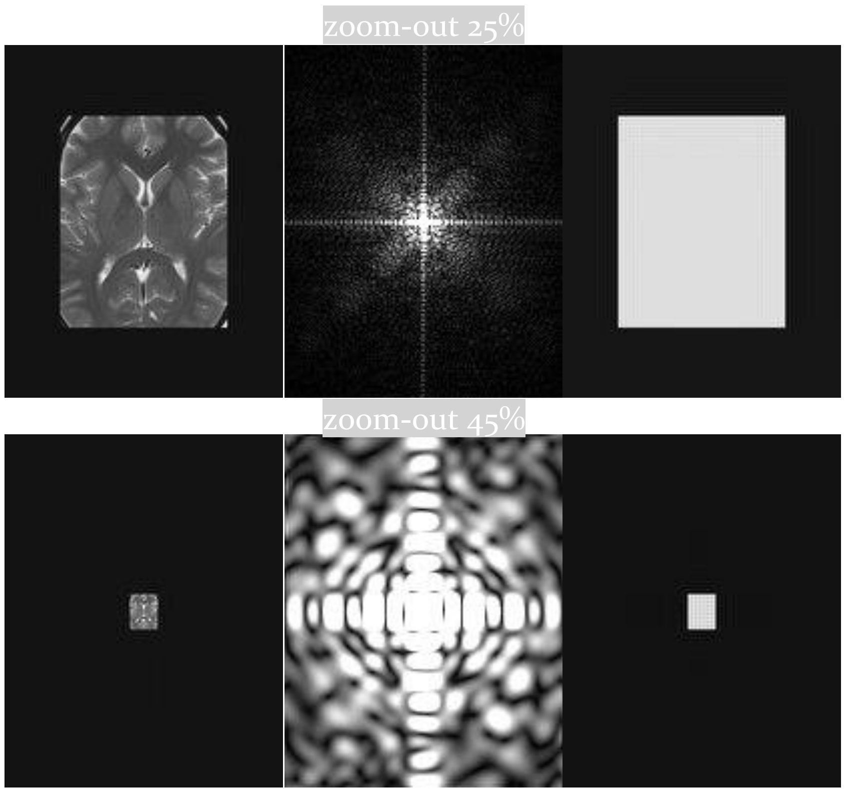
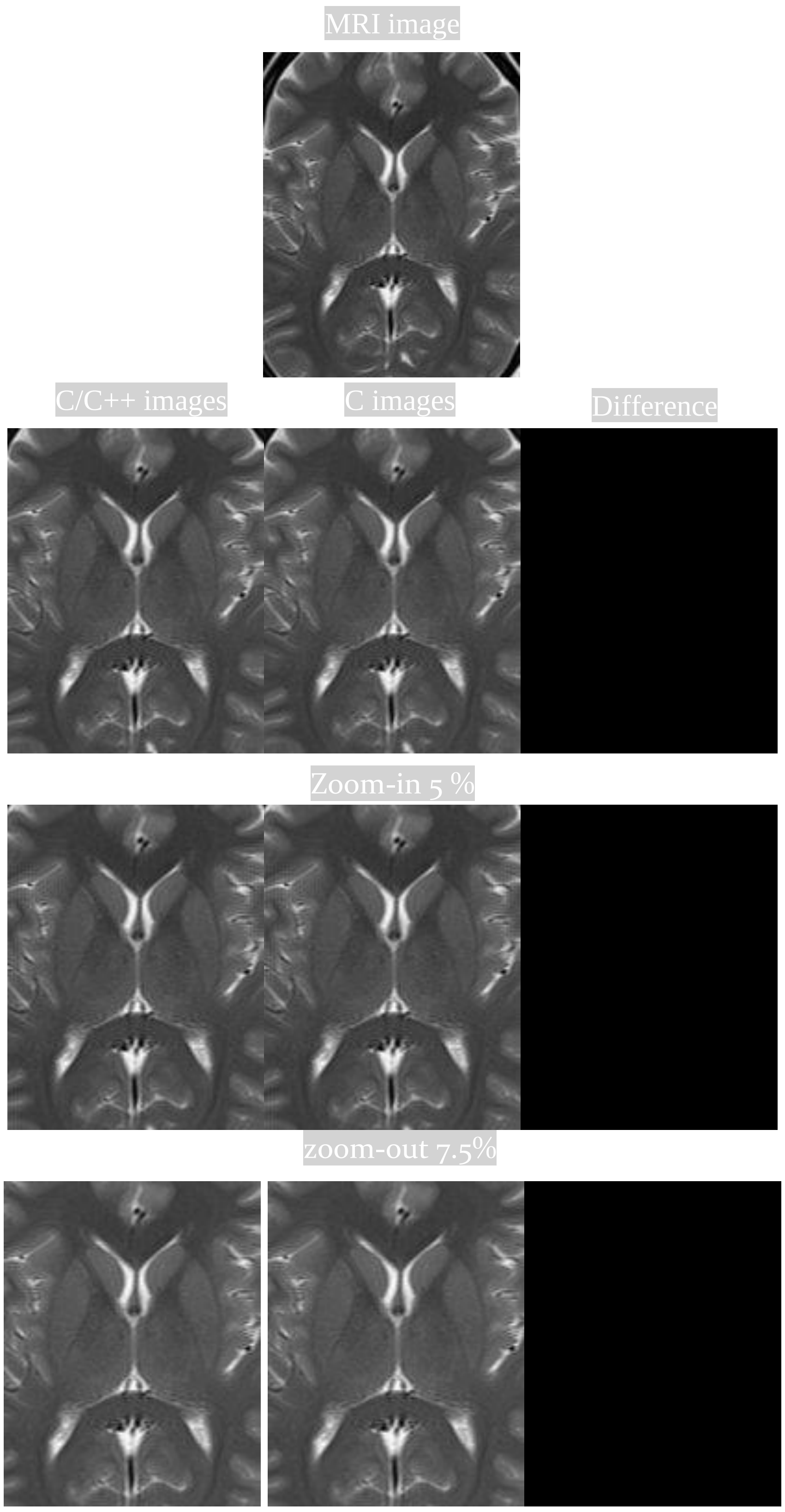


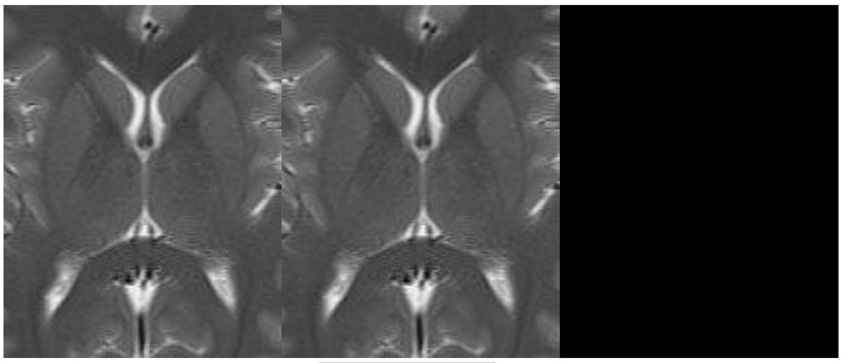
Figure 20. Adaptive Sinc-Zoom-out. Top row: MRI image (left) and its k-space magnitude (right).

4.4 Results and Discussion of the Theory

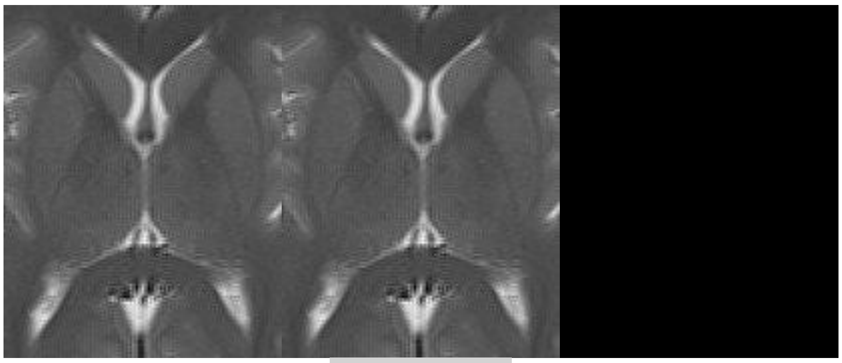
We compare the images generated from the code in C/C++ with the images generated from the code in pure C programming language (Figure 27).


MRI image
C/C++ images
C images
Difference
Zoom-in 5 %
zoom-out 7.5%

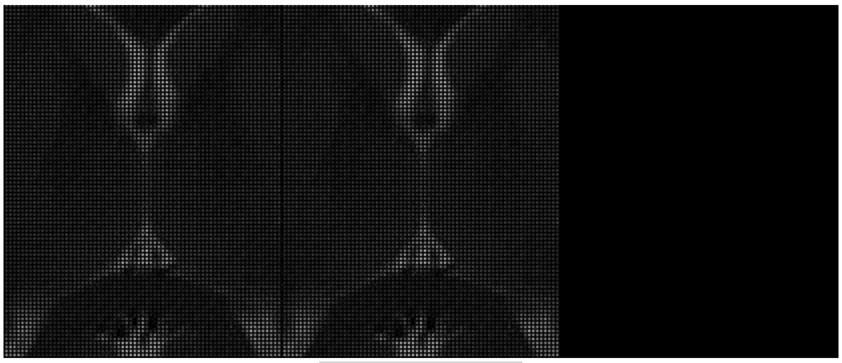
zoom-in 10%



zoom-in 15%



zoom-in 25%

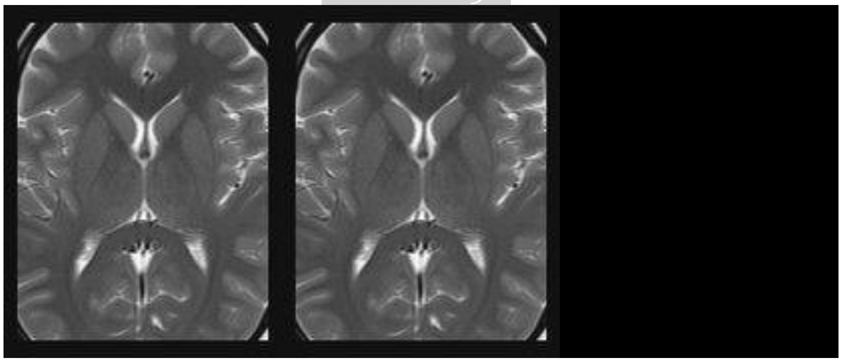


zoom-out 50%

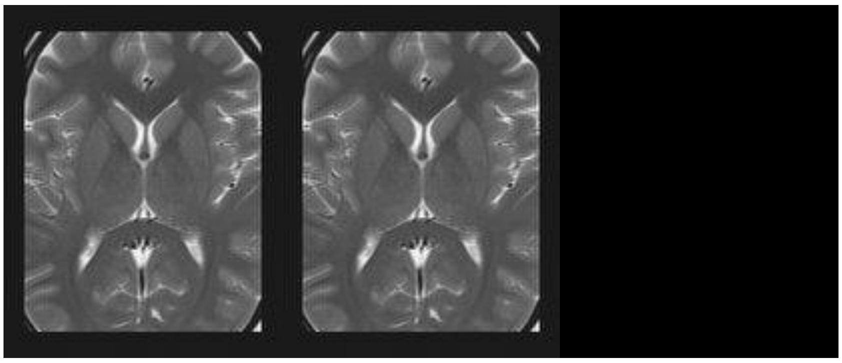
Figure 21. Zoom-in images of C/C++ and pure C

C/ C++images C images Difference

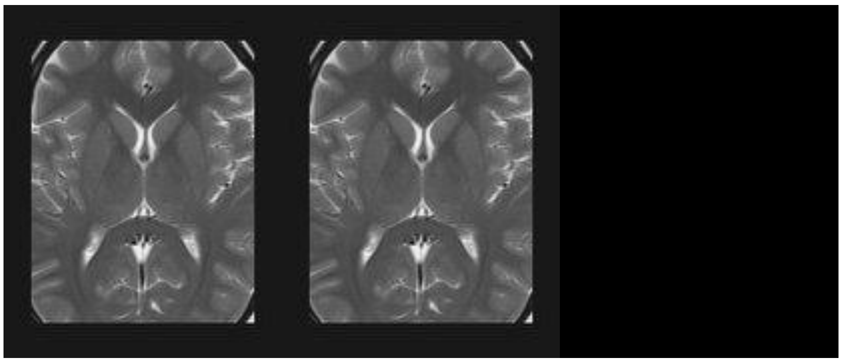
Zoom-out 5%



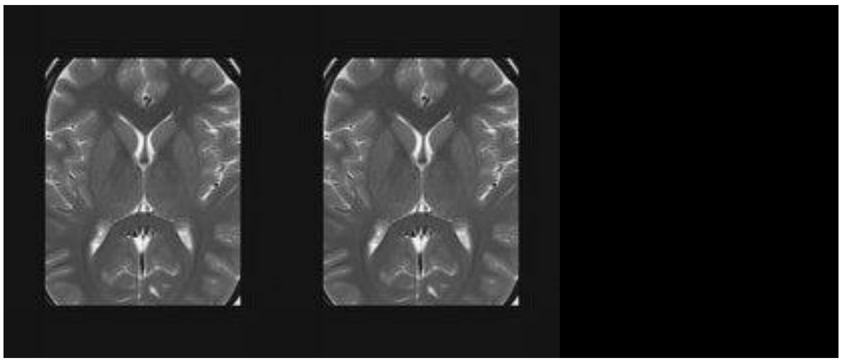
Zoom-out 7.5%



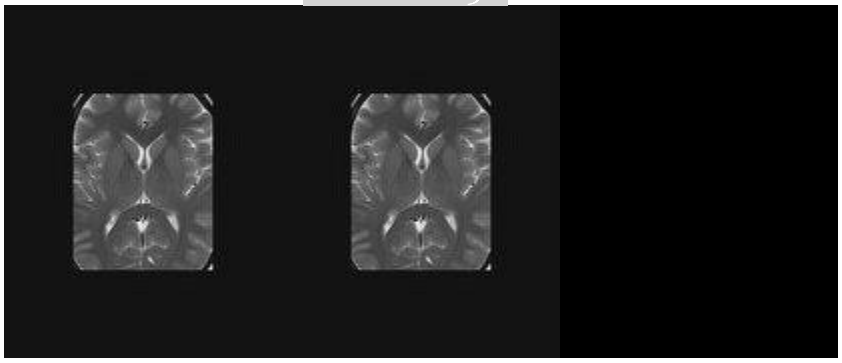
Zoom-out 10%



Zoom-out 15%



Zoom-out 25%



Zoom-out 40%

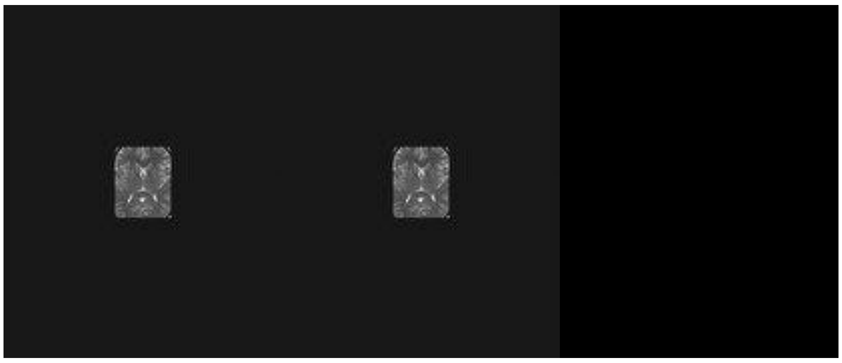
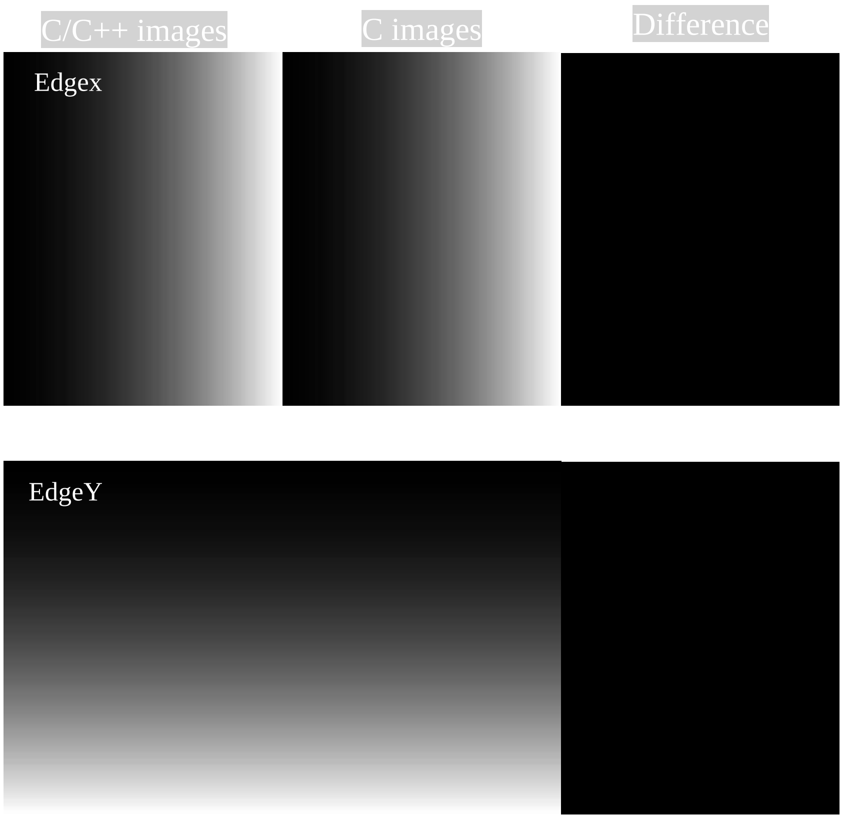
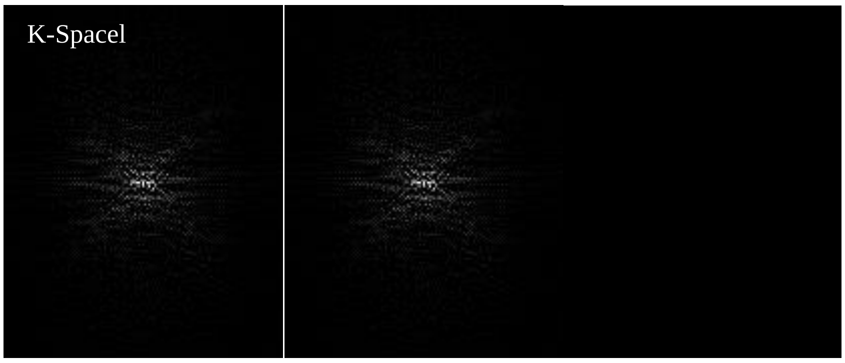


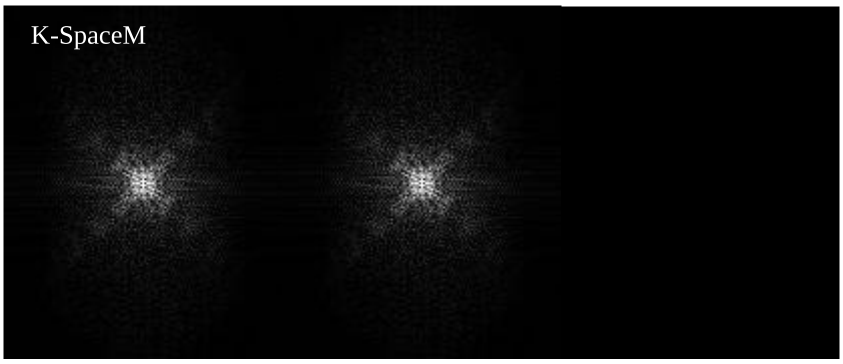
Figure 22. Zoom-out images generated from C/C++ and pure C

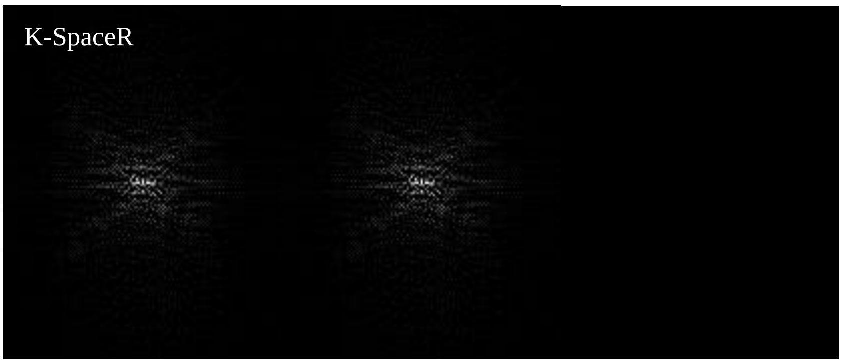
We take zoom-in 5% and compare the images generated from the program in C/C++ with the images generated from the program in pure C programming language (Figure 29) of the same percentage. After we subtract those two images on ImageJ, we get the results with x and y equal with 0.

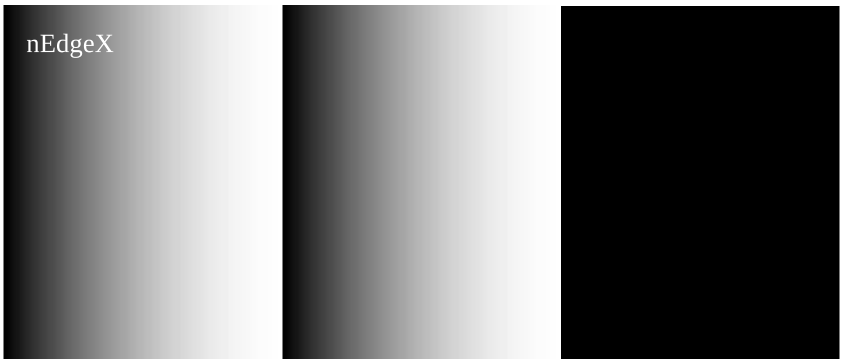
zoom-in 5%

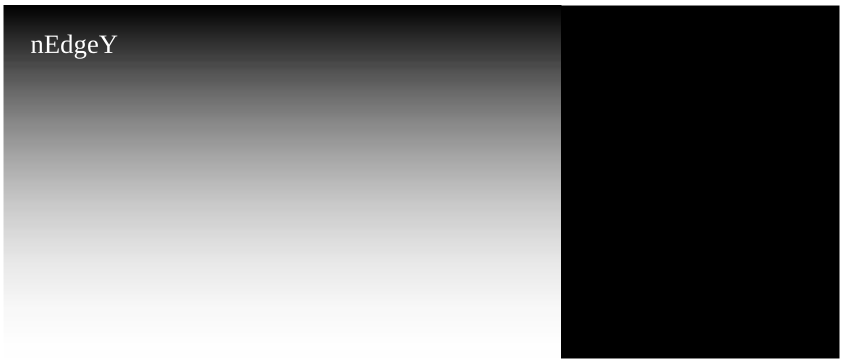


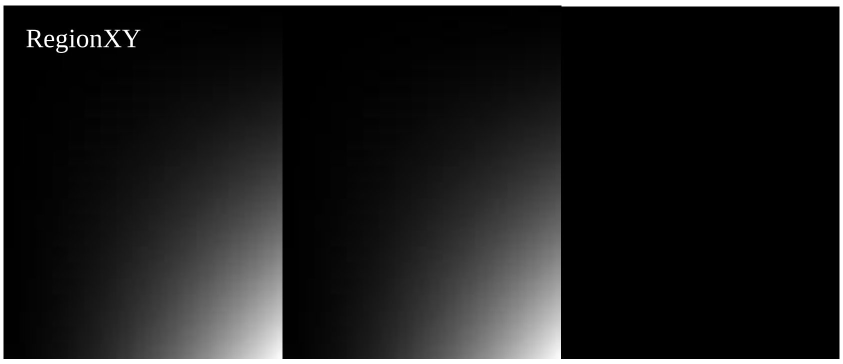

K-SpaceI


K-SpaceM

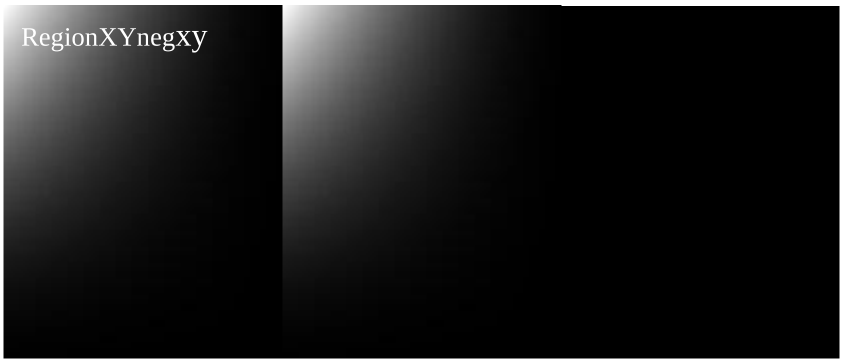

K-SpaceR

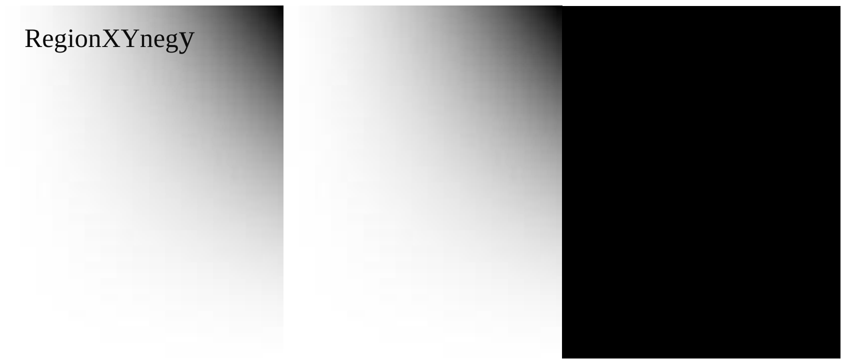

nEdgeX

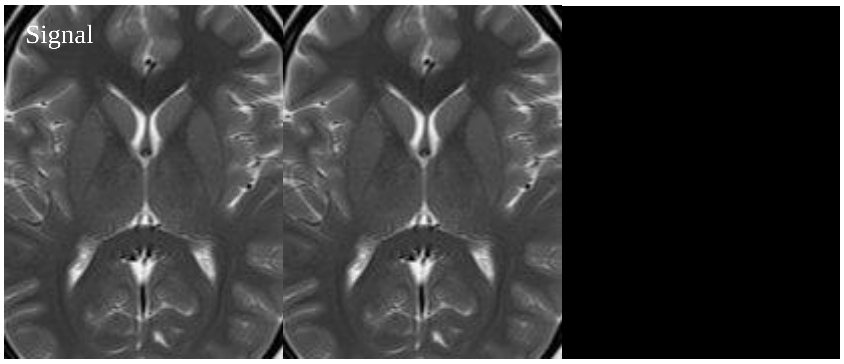

nEdgeY

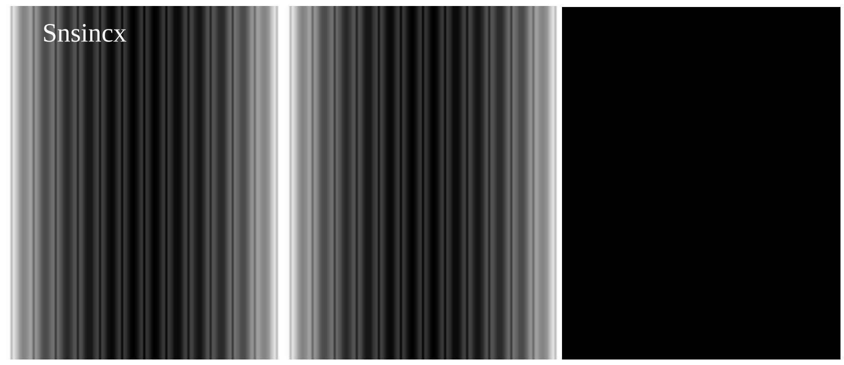

RegionXY

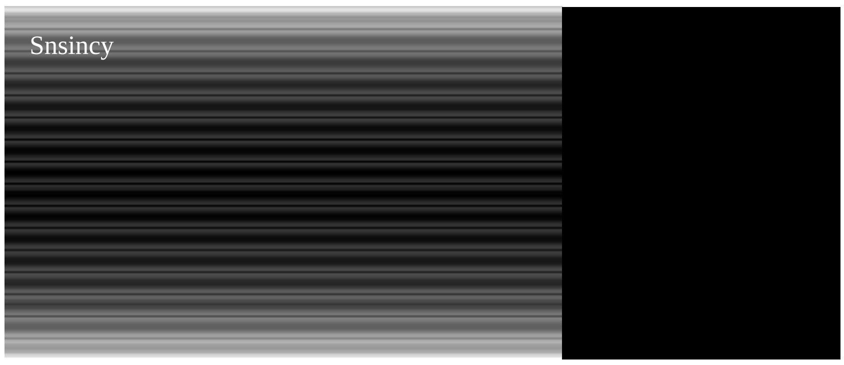
RegionXYnegx

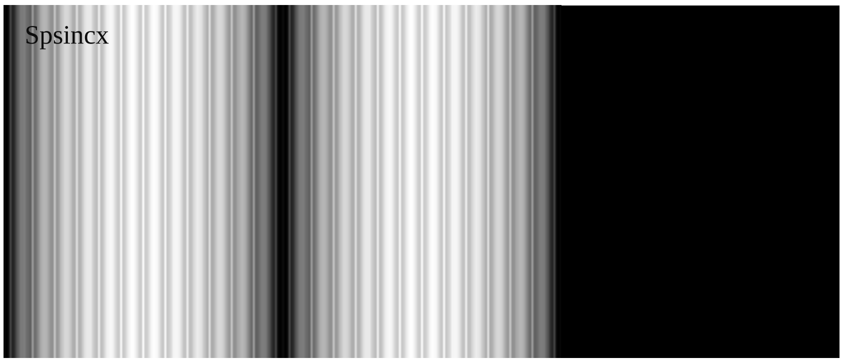

RegionXYnegXY

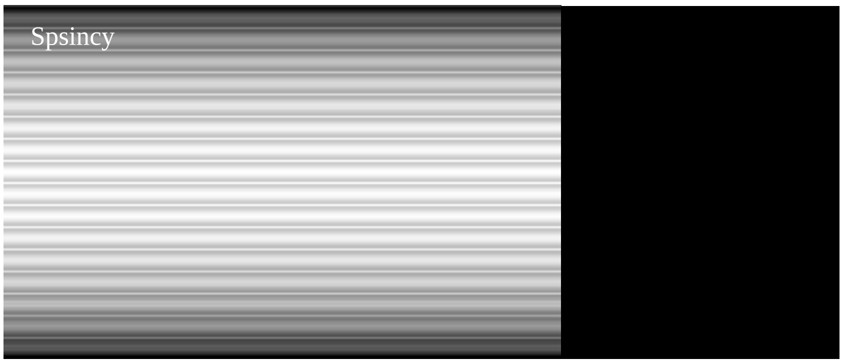

RegionXYnegy

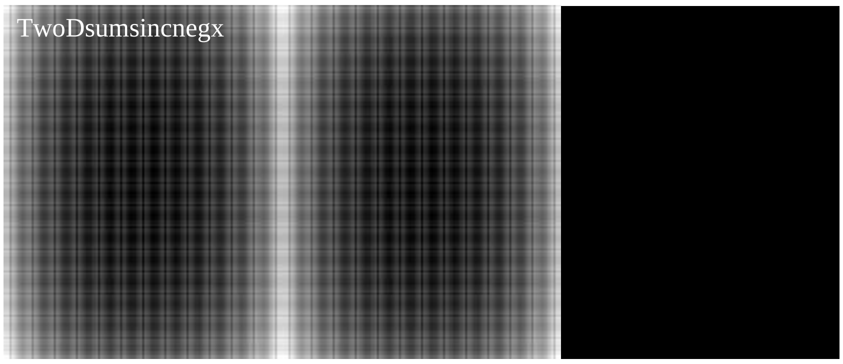

Signal

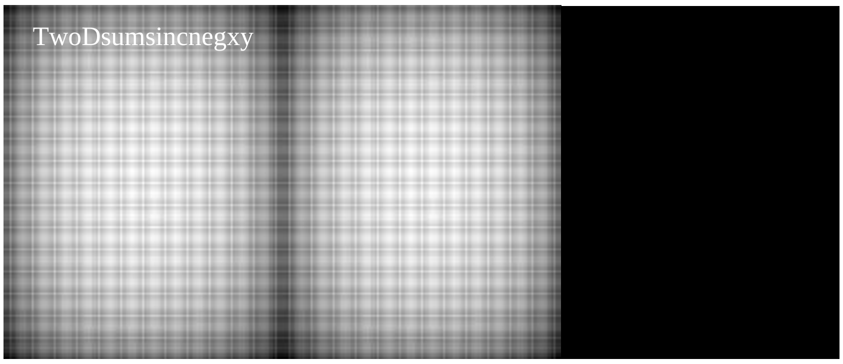

Snsincx

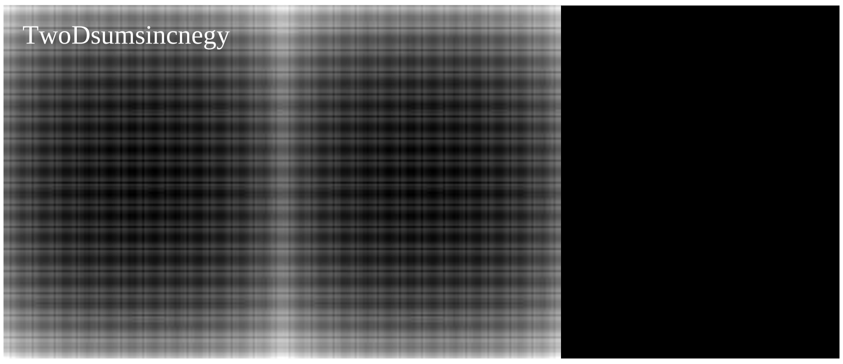

Snsincy


Spsincx


Spsincy


TwoDsumsincnegx


TwoDsumsincnegxy



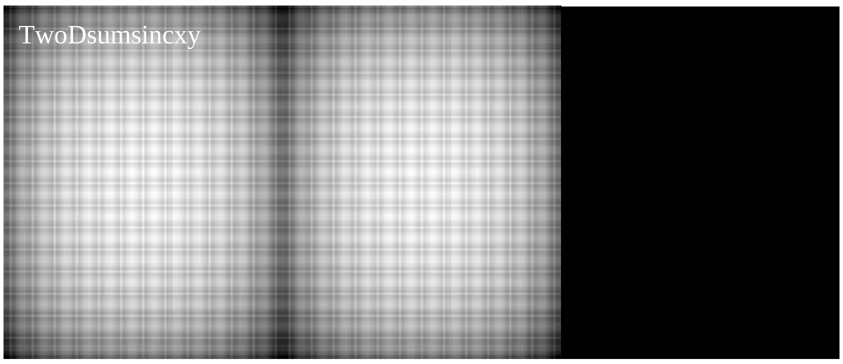


Figure 23. Zoom-in images. For the two cases of zoom-in for 5%, and from left images generated from the program in C/C++ to right images generated in C

On Figure Edge X and Edge Y are sampling interval edge images. The former refers to the Rx image when Rx is positive, and the latter refers to Ry image when Ry is positive.

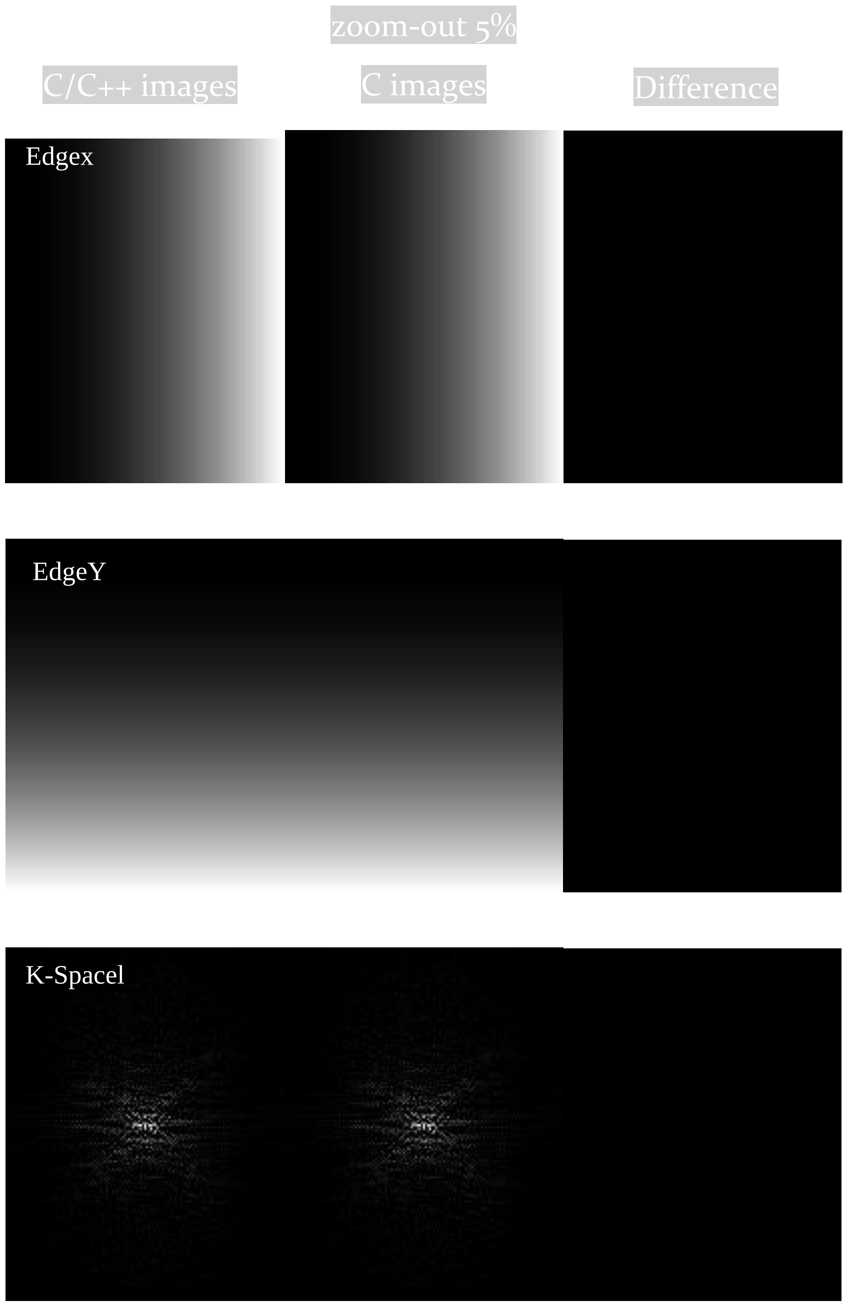
Region XY, Region XY neg, Region XY neg X and Region XY neg Y are all regions' images (they are the sampling intervals of the Sinc functions). Region XY is the Rxy image when both Rx and Ry are positive, Region XY neg is the Rxy image when both Rx and Ry are negative, Region XY neg X is the Rxy image when Rx is negative, and Ry is positive and Region XY neg Y is the Rxy image when Rx is positive and Ry is negative.

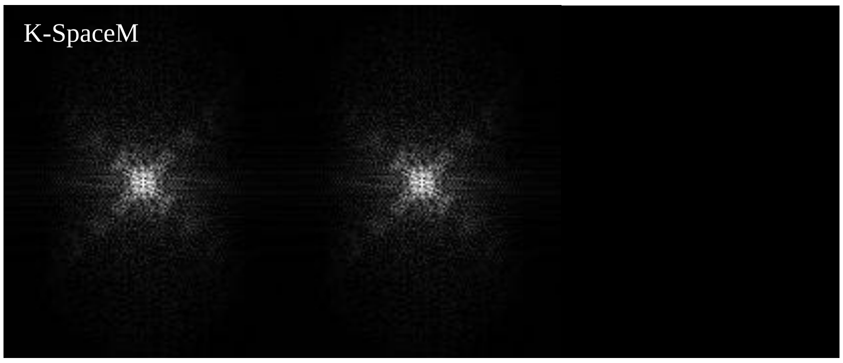
Sinc neg X, Sinc neg Y, Sinc pos X and Sinc pos Y are images obtained from 1D sinc functions. Sinc neg X is sinc of Rx negative, Sinc neg Y is sinc of Ry negative, Sinc pos X is the sinc of Rx positive and Sinc pos Y is the sinc of Ry positive.

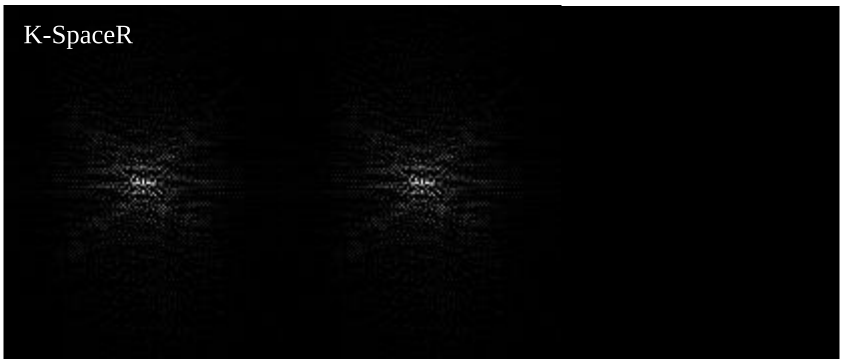
Sinc XY, Sinc neg X, Sinc neg Y and Sinc neg XY are images obtained from 2D sinc functions. Sinc XY is sinc of Rxy positive, Sinc neg X is sinc of Rxy when

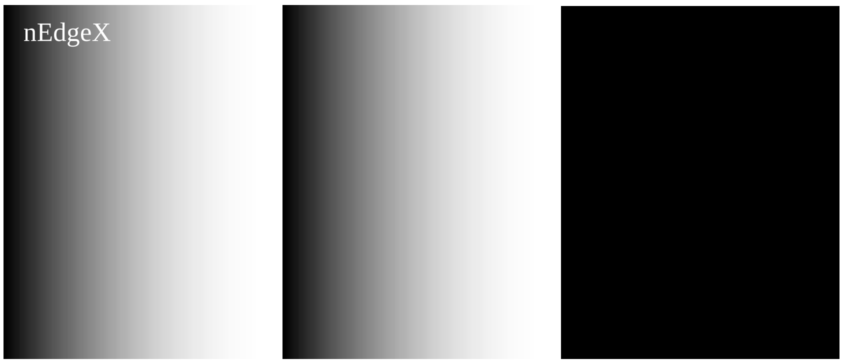
Rx negative and Ry positive, sinc neg Y is Sinc of Rxy when Rx positive and Ry negative and Sinc neg XY is sinc of Rxy when Rx negative and Ry negative.

We take zoom-out 5% and compare the images generated from the program in C/C++ with the images generated from the program in pure C programming language (Figure 24) with this percentage. After we subtract those two images on ImageJ, we get the results with x and y equal with 0.

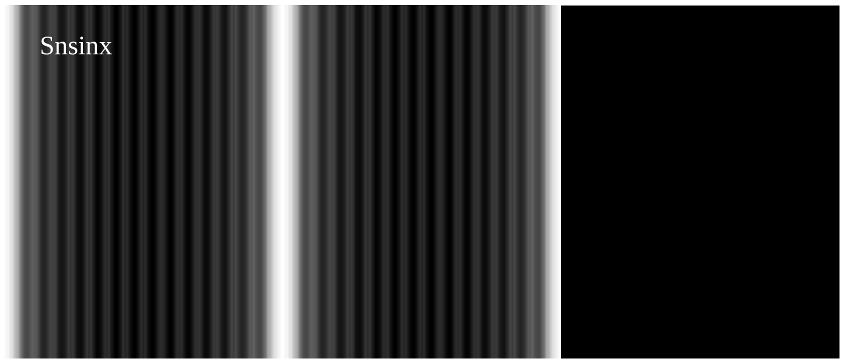



K-SpaceM

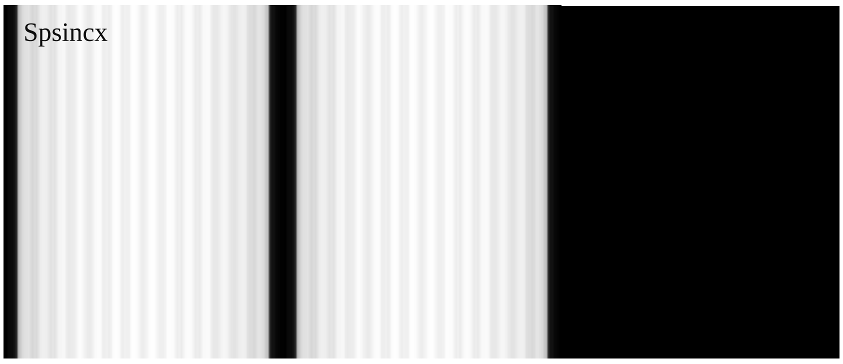

K-SpaceR

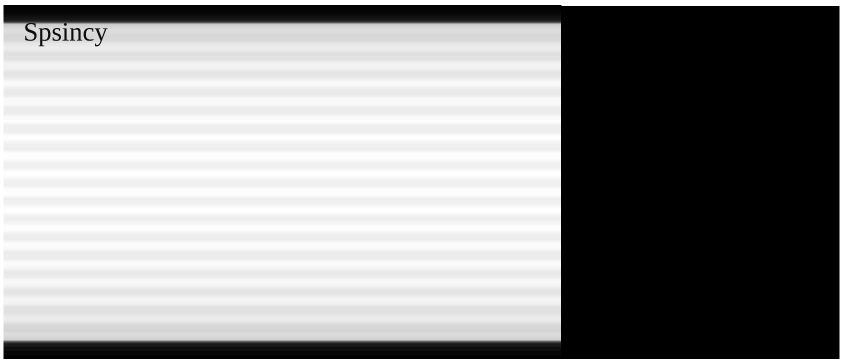

nEdgeX

nEdgeY


Snsinx

Snsincy


Spsincx


Spsincy

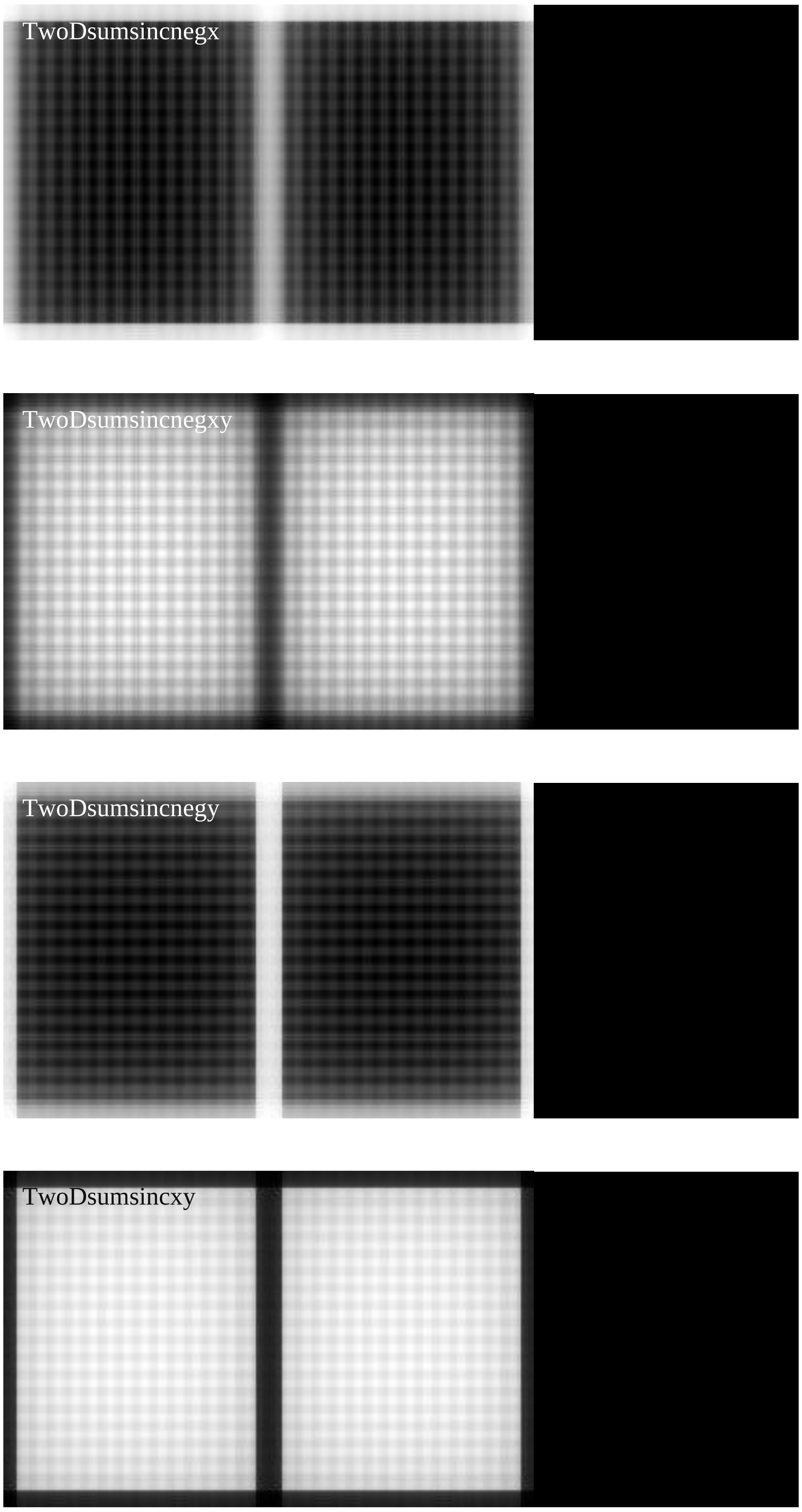


Figure 24. Zoom-out images. For the two cases of zoom-out for 5%, and from left images generated from the program in C/C++ to right images generated in C

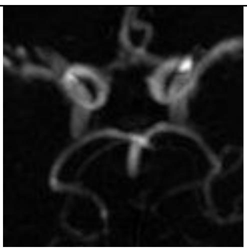
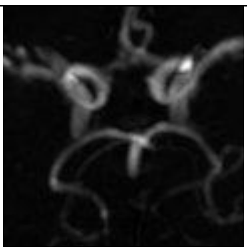
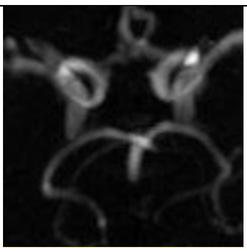
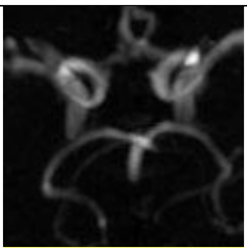
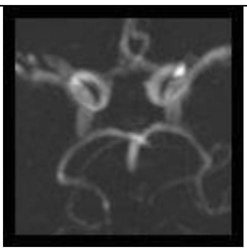
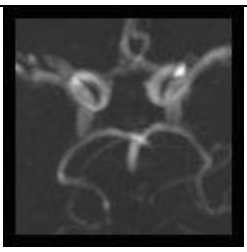
Image in size 120x120	Images generated in C/C++	Images generated in C	Running time of images in C/C++	Running time of images in C
No-zoom			135 seconds	129 seconds
Zoom-in			144 seconds	108 seconds
Zoom-out			154 seconds	109 seconds

Table 2. Image 120x120

On Table 2 images with no-zoom, zoom-in and zoom-out are generated from both programs, code in C/C++ and code in pure C. From the running time of these images, we could see that the code in C generated images in a shorter period than from the code in C/C++.

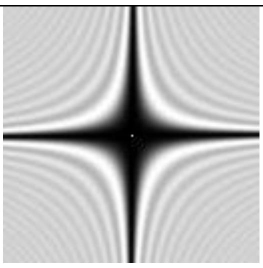
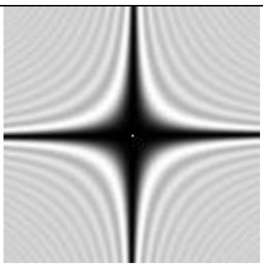
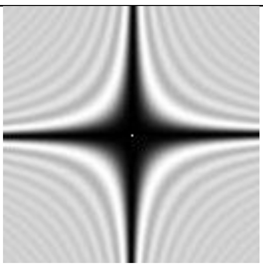
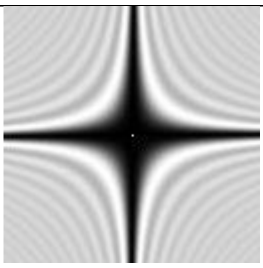
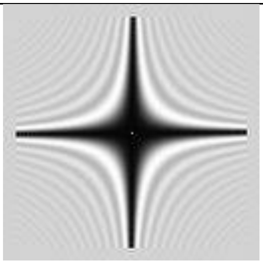
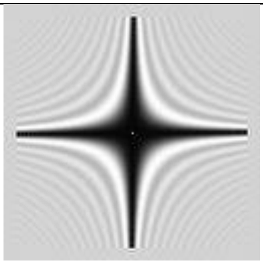
Image in size 128x128	Images generated in C/C++	Images generated in C	Running time of images in C/C++	Running time of images in C
No-zoom			184 seconds	159 seconds
Zoom-in			181 seconds	150 seconds
Zoom-out			179 seconds	155 seconds

Table 3. Image 128x128

On Table 3 images of size 128x128 with no-zoom, zoom-in and zoom-out are generated from both programs, code in C/C++ and code in pure C. From the running time of these images, we could see that the code in C generated images in a shorter period than from the code in C/C++.

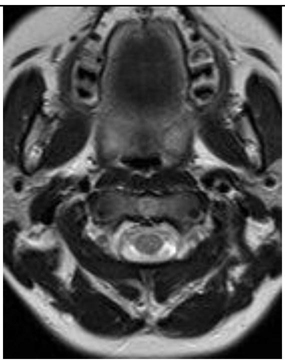
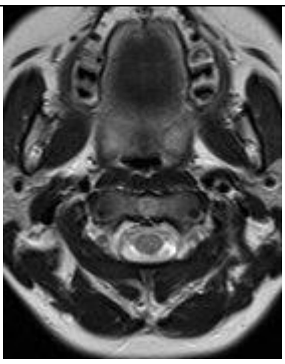
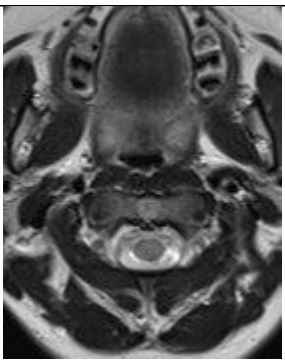
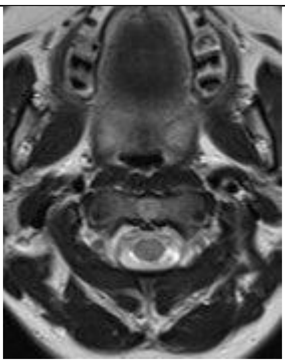
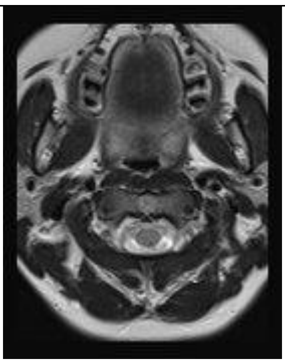
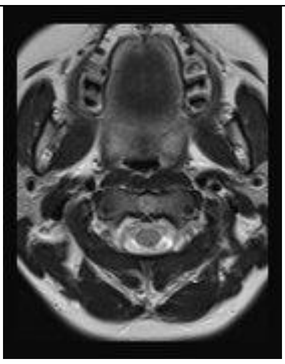
Image in size 139x176	Images generated in C/C++	Images generated in C	Running time of images in C/C++	Running time of images in C
No-zoom			386 seconds	246 seconds
Zoom-in			408 seconds	395 seconds
Zoom-out			404 seconds	363 seconds

Table 4. Image 139x176

On Table 4 images of size 139x176 with no-zoom, zoom-in and zoom-out are generated from both programs, code in C/C++ and code in pure C. From the running time of these images, we could see that the code in C generated images in a shorter period than from the code in C/C++.

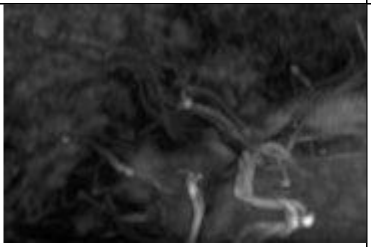
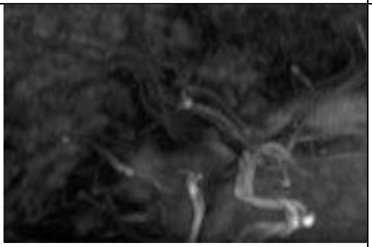
Size 234 x119	Images generated in C/C++	Images generated in C	Running time of images in C/C++	Running time of images in C
No-zoom			493 sec	308 sec
Zoom -in			518 sec	453 sec
Zoom -out			382 sec	463 sec

Table 5. 234x119

On Table 5 images of size 234x119 with no-zoom, zoom-in and zoom-out are generated from both programs, code in C/C++ and code in pure C. From the running time of these images, we could see that the code in C generated images in a shorter period than from the code in C/C++.

CHAPTER 5

CONCLUSIONS

5.1 Conclusions

This thesis translates a novel Sinc Zooming application from C/C ++ to pure C programming language and achieves the same results as generated from two programming languages. The first step for the methods part was to translate the application from C/C++ into pure C programming language. During the translation we removed the libraries that were specific only on C++ and replaced them with equivalent C libraries. It was observed that there was the keyword ‘class’ that was used to obtain C classes, whereas there is no such a keyword in C. Pointers in C++ are pointers to ‘struct data’, but in C we specify the type of pointer. For printing something which in our case was printing an image, in C++ it is used cout, while in C it is used printf.

After the code was translated in pure C, we started generating zoomed images from the program written in C/C++. The images were generated with cmd for every percentage of zoom-in and zoom-out with their respective bandwidth (B) and sampling rate (SR). With the program in C/C ++ adaptive and non-adaptive zoom images were generated in different percentages by generating the zoom-in, zoom-out and no-zoom images with the k-space magnitude. Zooming-in (image magnification) was for us to see finer and more visible details of an image, while zooming-out (image reduction) was for showing a wider view of an image. With zooming-in, the magnification level of the image was increased, by focusing on a specific area of the image and making it appear larger. With zooming-out the image is reduced making the image appear smaller and with less visible details by providing the context of a larger picture. No-zoom was the same as the original image 100%.

With the increased percentage of zoom-in, sampling rate was increased too which meant potential degradation and loss of image quality. When the percentage was low, for example 20% and with a low sfu, zoom-in effect enhanced the visibility of the details on the image. In zoom-out with the increased percentage, sampling rate was reduced. With a 10% zoom-out, the global features extracted from the overall image were emphasized. When the sampling rate decreased, then there was loss of detail and clarity. No-zoom was for the unaltered versions of test images, by maintaining a sample rate of 2.0 by keeping the level of detail from the original images.

Then images like Sinc image, k-space of the Sinc image and SincXY, with the program that was translated into pure C programming language, were generated again with the same percentages as the program in C/C++. We compared those three images from both programs and saw that the generated images were the same. Also, to be more precise with the results, we took a 5% zoom-in and generated all the images in ImageJ and compared them with the 5% of the zoom-in of the program in C and generated the same images. Then we also took a 5% zoom-out and did the same thing and generated the same exact images from the two programs. To see if there were any differences in those images, we used subtraction on ImageJ and plotted histograms to be more correct.

The results that were obtained at the end from the programs, were the results that show that we can generate the same zoom images using two programs written with different programming languages. In our case we had a combination of two programming languages (C and C++) and another one in a pure C programming language.

When we estimated the running time of the images of four different sizes from both programs in C/C++ and in C, the running time differ from one code to another. With the increasing of the image sizes, we could see that the images needed more time to generate and from both programs we could see that the program in C took less time which means that it was faster.

In conclusion we have concluded that we can get the same zoomed images from two different programming languages, like in our case that the same images were generated from both the code in C/C++ with the code in C. The goal was reached by obtaining no errors after the substruction of two images with a result with x and y = 0.

REFERENCES

[1] *Digital Image Interpolation*. Understanding Digital Image Interpolation. (n.d.). https://www.cambridgeincolour.com/tutorials/image-interpolation.htm

[2] Vincent Mazet (Université de Strasbourg). (2020). *Basics of Image Processing*. Interpolation - Basics of Image Processing.

[3] Mahajan, S., & Harpale, V. K. (2015). *Adaptive and Non-adaptive Image Interpolation Techniques*

[4] Y. Zhang, Y. Li, J. Zhen, J. Li, and R. Xie, "The hardware realization of the bicubic interpolation enlargement algorithm based on FPGA," in Proc. 3rd Int. Symp. Inform. Process., pp. 277-281, Oct. 2010

[5] Xu, J., Chang, Z., Fan, J., Zhao, X., Wu, X., Wang, Y., & Zhang, X. (2015). Super-resolution via adaptive combination of color channels. Multimedia Tools and Applications, 76(1), 1553–1584.

[6] Mircea Merca. (2015, October 22). *The cardinal sine function and the chebyshev–stirling numbers*. Journal of Number Theory.

[7] Wikimedia Foundation. (2021, August 26). *SINC function*. Wikipedia.

[8] *Concept of zooming*. Online Courses and eBooks Library. (n.d.).

[9] *Digital Zoom vs optical zoom*. Diffen. (2014).

[10] Chang, C.-C., Chou, Y.-C., Yu, Y.-H., & Shih, K.-J. (2005). *An image zooming technique based on vector quantization approximation. Image and Vision Computing, 23(13), 1214–1225.*

[11] Koç, A., Bartan, B., & Ozaktas, H. M. (2020). *Discrete scaling based on operator theory. Digital Signal Processing, 102904.*

[12] Arcelli, C., Brancati, N., Frucci, M., Ramella, G., & Sanniti di Baja, G. (2011). *A fully automatic one-scan adaptive zooming algorithm for color images. Signal Processing, 91(1), 61–71.*

[13] *Zhou, D., Shen, X., & Dong, W. (2012). Image zooming using directional cubic convolution interpolation. IET image processing, 6(6), 627-634.*

[14] Wang, Z., Chen, J., & Hoi, S. C. (2020). *Deep learning for image super-resolution: A survey*. IEEE transactions on pattern analysis and machine intelligence, 43(10), 3365-3387.

[15] Doshi, M., Gajjar, P., & Kothari, A. (2022). *Zoom based image super-resolution using DCT with LBP as characteristic model*. Journal of King Saud University-Computer and Information Sciences, 34(2), 72-85.

[16] Cha, Y., & Kim, S. (2007). *The error-amended sharp edge (EASE) scheme for image zooming.* IEEE Transactions on Image Processing, 16(6), 1496-1505.

[17] Vaquero, D., Turk, M., Pulli, K., Tico, M., & Gelfand, N. (2010). *A survey of image retargeting techniques. Applications of Digital Image Processing XXXIII.*

[18] Sun, C., Lv, J., Li, J., & Qiu, R. (2018). *A rapid and accurate infrared image super-resolution method based on zoom mechanism. Infrared Physics & Technology, 88, 228–238.*

[19] Chen, H.-C., & Wang, W.-J. (2009). *Fuzzy-adapted linear interpolation algorithm for image zooming. Signal Processing, 89(12), 2490–2502..*

[20] Costarelli, D., Seracini, M., & Vinti, G. (2020). *A comparison between the sampling Kantorovich algorithm for digital image processing with some interpolation and quasi-interpolation methods.* Applied Mathematics and Computation, 374, 125046.

[21] Laffont, P., Jun, J. K., Wolf, C., Tai, Y., Idrissi, K., Drettakis, G., & Yoon, S. (2010). Interactive content-aware zooming. In *Graphics Interface* (pp. 79–87).

[22] Wikimedia Foundation. (2010, March 10). *Whittaker-Shannon interpolation formula*.Wikipedia.https://en.wikipedia.org/wiki/Whittaker-Shannon_interpolation_formula

[23] Wikimedia Foundation. (2010a, February 20). *Nyquist-Shannon sampling theorem*.Wikipedia.https://en.wikipedia.org/wiki/Nyquist-Shannon_sampling_theorem

[24] Almira, J.M. and Romero, A.E. (2011) *Image zooming based on - mat.uab.cat*. Available at: https://mat.uab.cat/~matmat/ebook2011/V2011n01-ebook.pdf.