Yarmouk University

Hijjawi Faculty for Engineering Technology

Computer Engineering Department

Speed Enhancement and Parallelization of Saw-Tooth Fringe Projection Technique

This Thesis was Submitted to the Department of Computer Engineering in Partial Fulfillment of the Requirements for the Master's Degree of Science of computer engineering/Industrial automation at Yarmouk University

By

Mohammad Majed Ahmad Saleem

Supervisor

Dr. Sami Hamdan

April, 2017

Speed Enhancement and Parallelization of Saw-Tooth Fringe Projection Technique

By

Mohammad Majed Ahmad Saleem

This Thesis was submitted to the department of Computer Engineering in Partial Fulfillment of the Requirements for the Master's Degree of Science in Computer Engineering/Industrial Automation.

Signature of Author: [illegible]

Committee Member **Signature and Date**

Dr. Sami Al-Hamdan (Chairman) ٢٠١٧/٥/١١ [illegible]

Assistant Professor, Computer Engineering, Yarmouk University

Dr. Osameh Al-Kofahi (Member) ٢٠١٧/٥/١٢ [illegible]

Assistant Professor, Computer Engineering, Yarmouk University

Dr. Ahmad Al-Mousa (External Examiner) 14/5/2017 [illegible]

Assistant Professor, Telecommunications Engineering, Yarmouk University

Dedication

This thesis is dedicated to:

My great parents, you are the candle that light up my life. Your encouragement and believing in me were the reasons why I studied Master. Without your help I couldn't be able to make any success in my life. I dedicate this thesis to you, I wish that graduating as a Master student will make you happy and proud of me. I wish I can put smiles on your faces for the rest of my life. No words can explain how much I am happy and proud to be your son. Thanks for all of your support in all the ways you could.

My great brothers: Yazan, Laith, Ghaith and Aiham. I would like to thank all of you for your support and encouragement. I am sincerely thankful for having you in my life. I believe in all of you and proud to be your brother, I hope we will success in our life together and will make our parents proud and happy.

My great friends, who always supported me in many ways and wished me a successful life, I am glad that I have such friends. We lived our life like brothers; any success of any one of us is a great successful for the all. I dedicate this thesis to you and I hope that our friendship will continue for the rest of our life.

I would like also to dedicate this thesis to my best friend and my brother; Mohammad Al-Ruhaimi. This is a great opportunity to thank you for all of your support and help in my life. No words will explain how much I am thankful to have you in my life. You were a great friend and brother for me, you supported in all the ways you could, you always believe I will be a successful person in my life, you provided me with a strong confident in myself. I wish we will be the persons that each one of us wants the other to be.

Acknowledgment

In the Name of Allah, the Most Merciful, the Most Compassionate all praise is to Allah, the Lord of the worlds; and prayers and peace be upon Mohamed His servant and messenger. Thanks to Allah, the Ever-Magnificent; the Ever-Thankful, for His help and bless. This work would have never been completed without His guidance.

I would like to express my sincere gratitude to my advisor Dr. Sami Al-Hamdan for the continuous support of my Master research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Master study.

I would like also thank all the people who helped me by providing the equipment and preparations for the applied part of the thesis: Y. Khamaisah, M. Al-Ruhaimi, M. Bawaneh, A. Obeidat, M. Jararwah, D. Al-sha'bi, A. Jaradat and H. Al-radaideh.

Last but not the least, I would like to thank my family: my parents for their support and encouragement to continue my study, they believed in me and supported me in all the ways they could.

Table of contents

Table of Figures

Table of tables

List of Abbreviations

DFD	Depth From Defocus
TOF	Time Of Flight
DFP	Digital fringe projection
FPP	Fringe Pattern Profilometry
SIMD	Single Instruction, Multiple Data
SISD	Single Instruction, Single Data
MISD	Multiple Instruction, Single Data
MIMD	Multiple Instruction, Multiple Data
UMA	Uniform memory access
NUMA	Non-uniform memory access
CCD	Charge-coupled device
FTP	Fourier transform profilometry
DFFTs	Dimensional discrete fast Fourier transforms
MKL	Math Kernel Library
EFFT	Enormous Fast Fourier Transforms
KLT	Kanade-Lucas-Tomasi
SIFT	Scale Invariant Feature Transform
LAN	Local Area Network
PSSD	Partitioning and scheduling following divisible
EQS	Traditional equal-partitioning
LCD	Liquid Crystal Display
WRF	Weather forecasting
IDE	Integrated development environment

ABSTRACT

Saleem, Mohammad Majed. Speed Enhancement and Parallelization of Saw Tooth Fringe Projection Technique. MSc. Thesis, Yarmouk University, 2017. (Supervisor Dr. Sami Al-Hamdan)

Profile measurement is the process of extracting a 3D surface map of an object by collecting and analyzing some data about the object. 3D shape measurement is widely used in many applications and fields such as gaming industry, medical diagnosis, industrial manufacturing and many others. Fringe pattern projection techniques (FPP) are non-contact profile measurement methods that are widely used nowadays. This research presents mainly speed enhancement methods for the recently studied intensity based saw-tooth technique developed by Al-Hiary [A practical implementation of intensity based saw-tooth fringe projection technique for surface profilometry measurement, master thesis, Yarmouk University, 2015]. In this work, the MATLAB code developed by Al-Hiary is converted to C code and then parallel versions are developed and tested on multi-core Intel based PCs. The intensity based saw-tooth profile measurement technique consists of two major steps: First projecting saw-tooth fringe patterns on a background flat surface alone and with object under test placed. Second, the reflected images from the first step are captured, enhanced, and then analyzed to extract the 3D map of the object. In this work, serial and parallel codes are developed for the various computations used in the intensity based saw-tooth technique. The serial code developed here was about 3.5x faster than its counter MATLAB code written by Al-Hiary. The different computations in the intensity-based technique are highly parallelizable; that is data dependency is low which allowed us to divide the captured images into equal parts that are assigned to different cores or threads in the computing system. The parallel code was tested on dual and four core PCs. Hyper threading technology available on these systems is also tested. Without hyper threading, two cores and four cores systems were capable of providing speedups of 1.8x and 3.7x, respectively. When hyper-threading is used (i.e. 4 logical cores and 8 logical cores), the same systems were capable of providing 3.4x and 6.2x speedups on the two-core and four-core systems respectively. It is important to clarify that the current algorithms for saw-tooth FPP are reasonably good for object with smooth surface variations, however, for more complex surfaces, the algorithm has to be modified and improved.

Keywords: Profilometry, Fringe Analysis, 3D modeling, Shape Measurement, Fringe Projection.

CHAPTER 1

Introduction

In this chapter, we will discuss 3D shape measurement and parallel processing. The first section will be about 3D shape measurement applications and techniques. The second section will be about parallel computer types, parallel computer memory architecture, multi-core system and parallel processing techniques.

1.1 3D shape measurement

3D shape measurement is a technology used for collecting data about an object to construct a three-dimensional model for it. It is very useful and used in many applications in different fields. For example, it is used for body shape measurement, volume measurement, reverse engineering, etc [1,2]. In the following paragraphs, some of the fields that utilize 3D shape measurement will be presented.

1.1.1 Applications of 3D modeling

Medical field: 3D measurement technologies are useful in medical applications. This can be found in dentistry because of the high accuracy, fast and non-contact benefits. A 3D teeth module of a patient is shown in Figure 1.1.

This technology helps to reduce costs and storage needs and gives an easy way to access the data of the patients with a local area network or online [3].

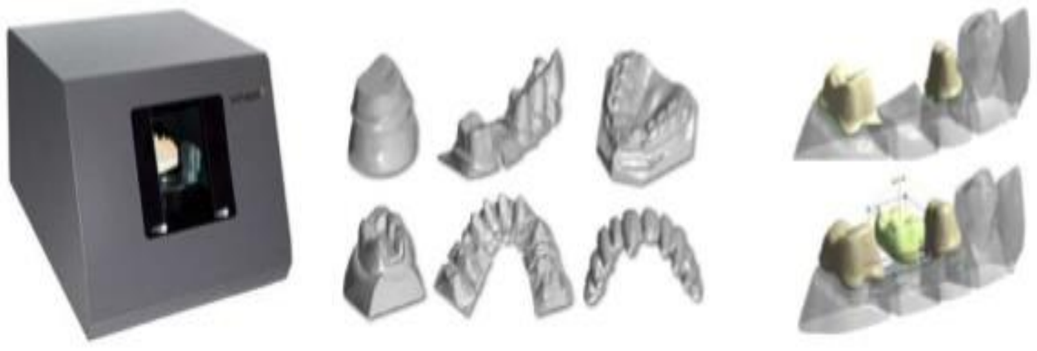


Figure 1.1: 3D Shape D-200 Dental 3D Scanner (Denmark) (left); sample measured data (center); 3Shape Dental Designer software examples (right) [3].

***Cultural heritage*:** 3D measurement techniques are also used in cultural heritage. In fact, 3D measurement in this field became very useful and important because of the non-contact role which makes it safe and keep the historical object away from damage. Another reason is that database records can be established for the historical object. This opened the door for the digital museums which can be viewed from a large number of people anywhere using a 3D digital view of the object. Figure 1.2 shows a real example of 3D modeling of a dinosaur skeleton and an Indian elephant [4].

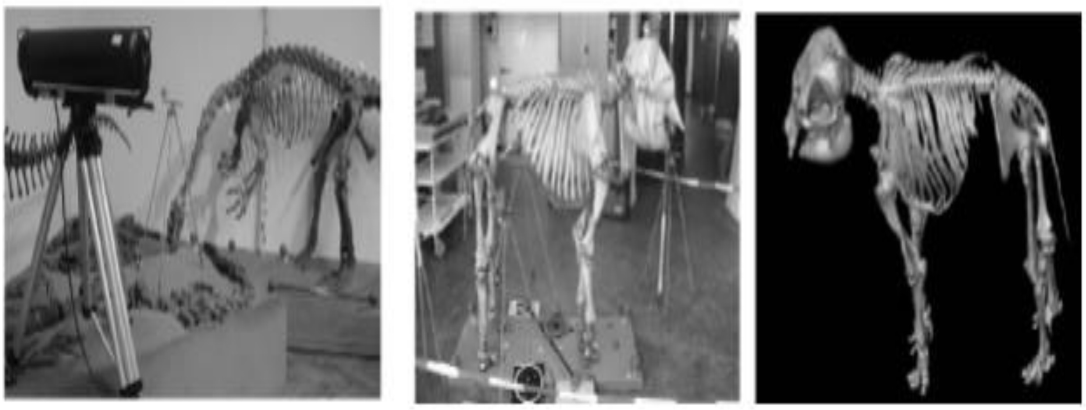


Figure 1.2: Left: dinosaur skeleton. Center and right: Indian elephant skeleton [4].

Fashion: OptiTex Company developed a few applications to create perfect clothing fit.

OptiTex applications scan the body of the customer and using 3D measurement algorithms they create an avatar for it. Using 3D visualization techniques, the customers will be able to try clothes online on their avatars and decide which one is perfect and suitable for them. Figure 1.3 shows the process of the software [5].

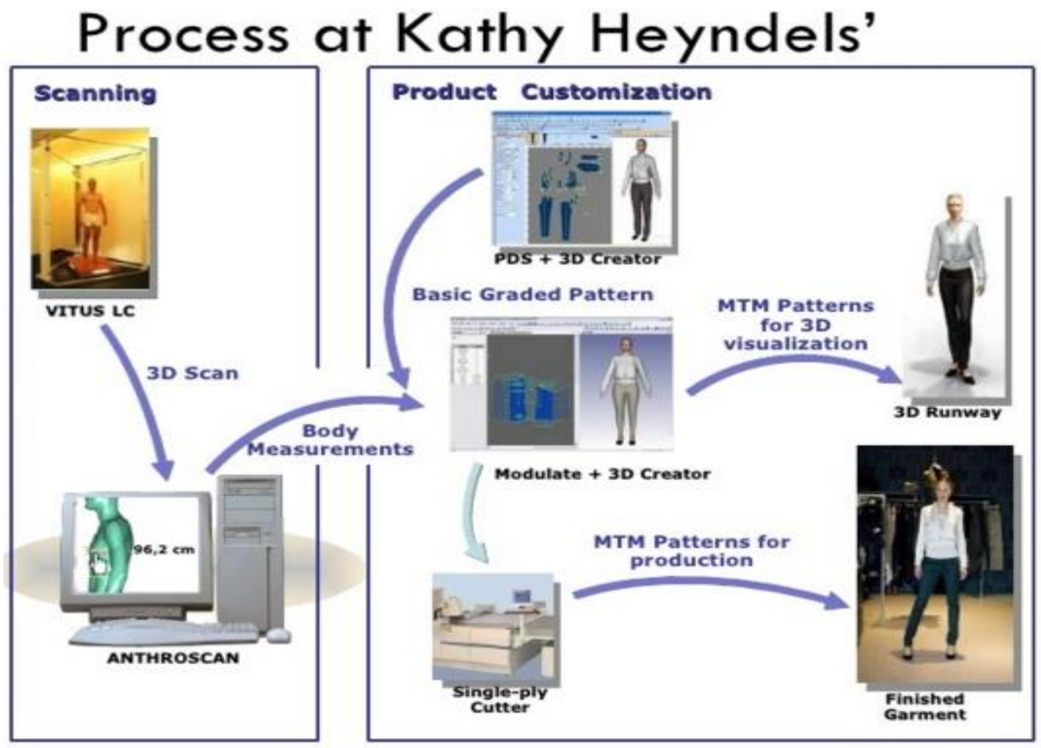


Figure 1.3: Kathy Heyndls' process [5].

Crime scenes documentation:

3D modeling is also used in crime investigation field. It is used to acquire a simulated crime scene. In [6,7] the authors discussed the use of a laser structured light and video pairs for the documentation of 3D crime scenes. HDS3000 laser scanner was used to help in data acquisition. A dummy was used to simulate the victim as shown in Figure 1.4 [8]. After modeling the crime scene, the police investigators can go back to the crime scene and investigates it several times with no fear of losing crime evidences since they have been modeled and digitally saved at the first time the police arrived.

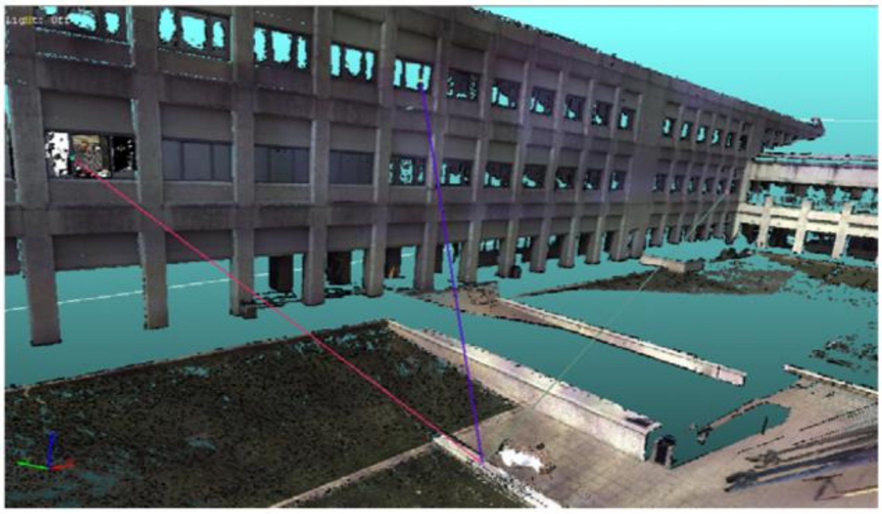


Figure 1.4: 3D model of the crime scene [8].

1.1.2 3D shape measurement techniques

Many different techniques are used in 3D shape measurement. These techniques are divided into two methods: contact and non-contact. The non-contact methods are also divided into passive and active.

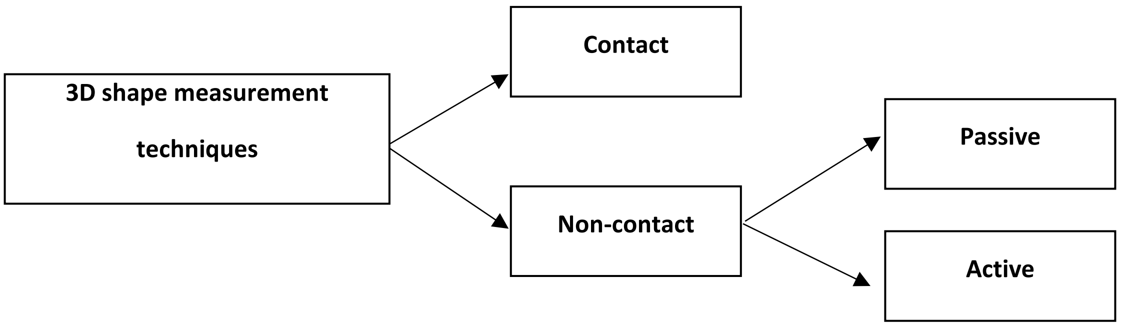


Figure 1.5: Classifications of 3D shape measurement techniques.

In contact methods, a metallic sensor is passed over the object to collect single points relative to each other and use it to calculate the 3D shape of this object.

The disadvantages of this method are that it requires contact with the object, thus, this contacting might damage the object or modify it. Also, these methods are slow and not very accurate when compared with non-contact methods [9].

Non-contact methods are more commonly used for 3D shape measurement. In these methods, there is no physical contact with the object. There are many non-contact techniques used for 3D shape measurement. Laser beam, light, x-ray, fringe patterns are some of the most popular techniques used. These methods are faster and more accurate than contact methods [9].

Non-contact 3D shape measurement methods are also classified into two main categories: passive and active techniques.

Passive techniques: These techniques do not use any kind of radiation and are usually used with dynamic objects which vary with time. Depth from defocus (DFD) and stereo vision are examples of passive techniques. These techniques merely rely on the ambient radiation, passive techniques might fail to provide high accuracy measurement results if the surface texture is not varying drastically from one point to another [10].

Stereo vision:

Stereo vision is a passive method technique. In stereo vision two cameras are used instead of one, see Figure 1.6. These two cameras are similar to human binocular vision. The depth of the point is calculated by comparing the two images captured by these cameras. The advantages of stereo vision system are that it is simple and not expensive.

The problem with this system is the identification of common points within the image pair [15].

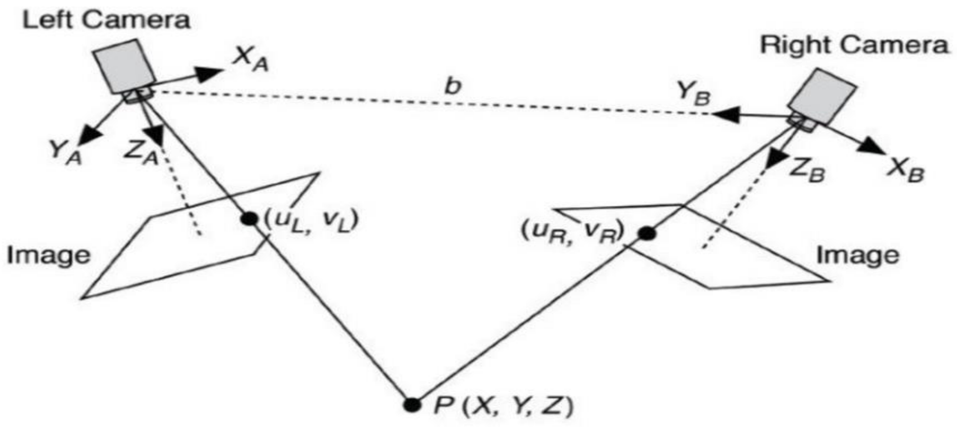


Figure 1.6: Principles of stereo vision system [15].

Photogrammetry:

Photogrammetry is the science of making measurement from photographs. Using photogrammetry, we can build 3D modules from digital images. The main steps of photogrammetry system are camera orientation and calibration, image point measurement, surface generation and finally texture mapping. Photogrammetry is used in many fields like city modeling, medical imaging and heritage [14]. Stereophotogrammetry is a special case of photogrammetry. A real example of using it is shown in Figure 1.7 [16], it is the Royal castle of Sweden module.

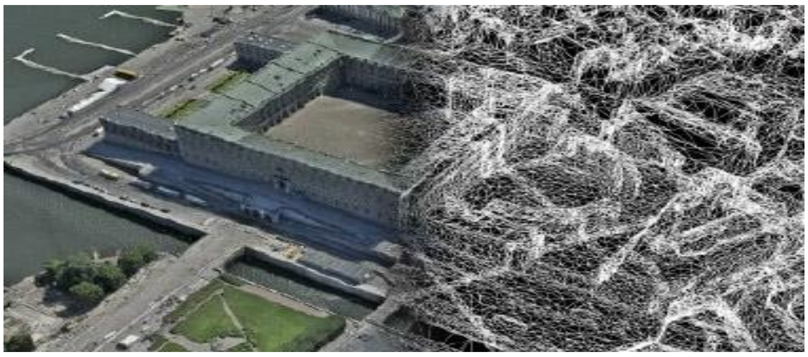


Figure 1.7: The Royal castle of Sweden module [16].

Active techniques: To solve the passive techniques limitations, active techniques have been developed. In active techniques, a kind of radiation is casted onto the object surface. 3D information of the object can be extracted by analyzing the interaction between the radiation and the object surface. Time of flight (TOF), laser triangulation and structured light are examples of active techniques [10].

Laser triangulation:

This method is based on the active triangulation principle [11]. The system setup is shown in Figure 1.8, the system consists of a laser source and image plane. The laser beam is sent to the object using the laser source. The laser beam reflected on a certain point on the image plane. Using the location of the point and geometry the depth of this point is calculated.

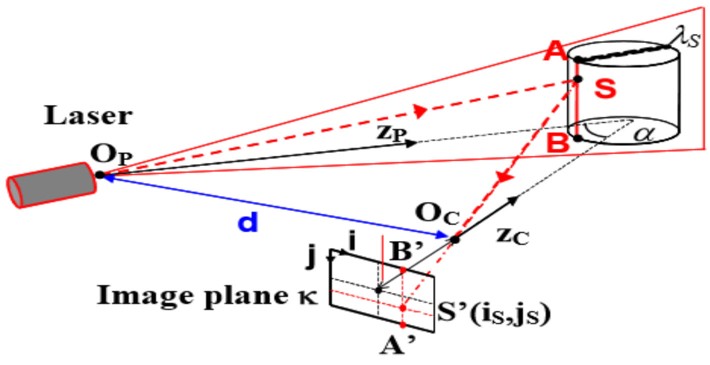


Figure 1.8: Schematics of the triangulation principle [11].

Structured light:

Structured light also shared the active triangulation approach of the laser triangulator. Instead of using a single laser beam to calculate the depth of a single point at a time, the structured light uses a single or multiple patterns to scan all the points at the same time and calculate the depth of them. Figure 1.9 shows the main principles of measurement in structured light systems and Figure 1.10 show examples of fringe projection schemes used [12,13].

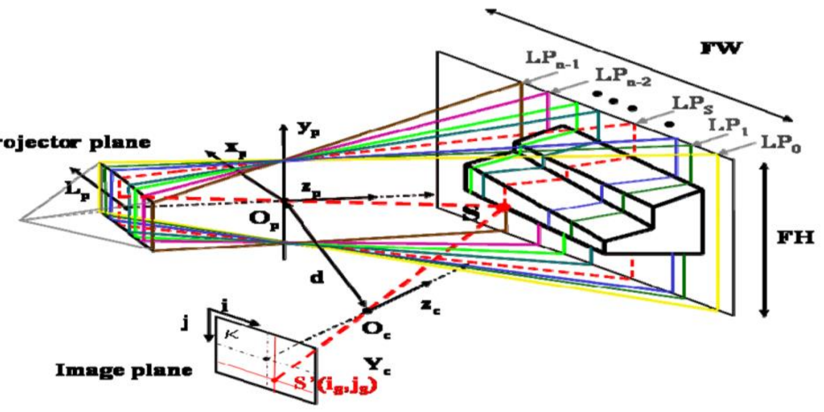


Figure 1.9: Principle of measurement in structured light systems [14].

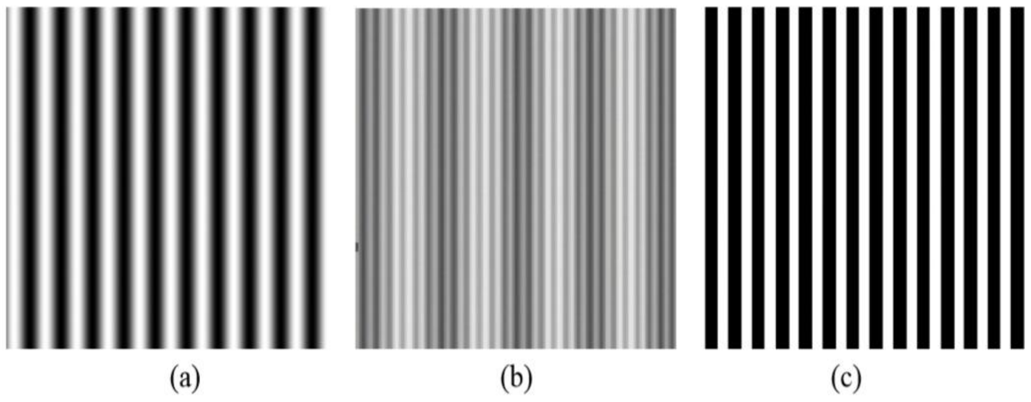


Figure 1.10: Example of fringe projection schemes. (a) Fringe pattern of sinusoidal fringes. (b) Superposition of two sinusoidal fringe patterns at different frequencies. (c) Projection of fringes of rectangular profile [14].

Profilometry:

Fringe Pattern Profilometry (FPP) is one of the most common 3D shape non-contact active measurement methods.

FPP has the advantages of high accuracy, simplicity, flexibility, and the ability to provide high-resolution [4]. Digital fringe projection (DFP) consists of a camera, reference plane and a digital video projector as shown in Figure 1.11 [17].

A computer generates a digital image with fringe patterns and then these patterns are casted onto the reference plane using a projector, after that the same fringe pattern is casted onto the surface of the object under measurement.

The camera captures the deformed fringe images that is caused by the variance of heights of the surface of the object, and send it to the computer. The computer analyzes the two images based on different algorithms (particularly triangulation) and the information carried by the deformed pattern, the 3D shape of the object can be retrieved [18].

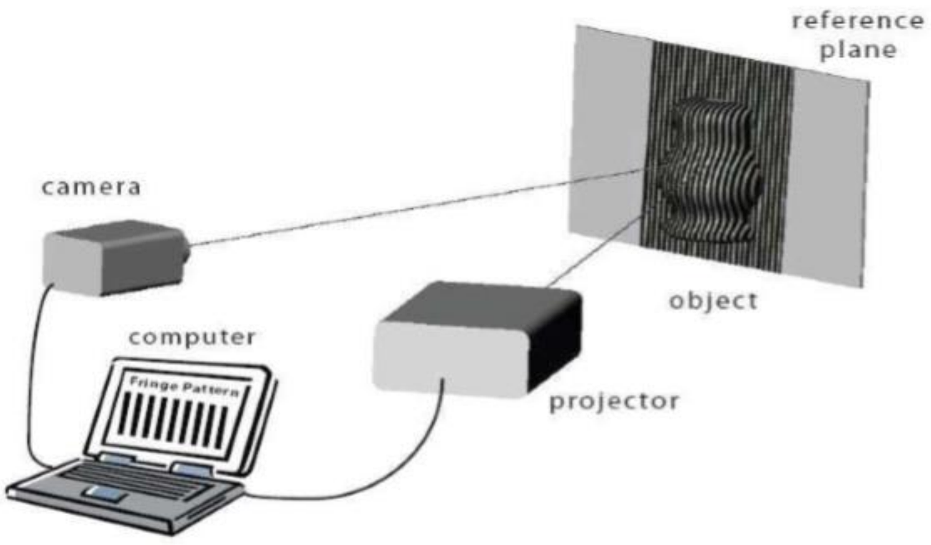


Figure 1.11: Fringe projection profilometry system [17].

After discussing the 3D shape measurement techniques, we made a comparison between the different techniques. The comparison is shown in Table 1.1, both the strength and the weakness of each technique is shown.

Technology	**Strength**	**Weakness**
Laser triangulations	1. Relatively simple. 2. Performance: independent of ambient light. 3. Acquisition rate is high.	1. Safety constraint when using laser. 2. Measurement range volume is limited. 3. Shadow problem. 4. Expensive.
Structured Light	1. Data acquisition rate is high. 2. Measurement volume is intermediate. 3. Performance: depend on the ambient light.	1. Safety constraints, if using the laser. 2. Complex computations. 3. Shadow problem. 4. Cost
Stereo Vision	1. Simple and cheap. 2. High accuracy.	1. Complex computations. 2. Data covering is sparse 3. Limited scene. 4. Data acquisition rate is low.
Photogrammetry	1. Simple and cheap. 2. High accuracy with targets.	1. Complex computations. 2. Limited scenes. 3. Data acquisition rate is low.
Profilometry	1. Real-time measurement. 2. Fast and Inexpensive. 3. Flexible and Robust. 4. Exactness, High resolution.	1. Computation demanding. 2. Performance: is affected by ambient light.

Table 1.1: Comparison of optical range imaging techniques [1,15].

1.2 Parallel processing

The number of transistors used to produce a processor is doubled every 20 months according to Moore law [19]. The clock speed of processors has also been increasing until it reach a maximum that cannot be increased. With more transistors and higher clock speeds, processors get faster but with more power consumption and hence higher processor temperatures.

In recent years, it is rare to find a processor with more than 3.5GHZ speed. The development of computers in recent years changed from complex, bigger and faster computer to develop multiprocessor computers and even mobiles or notebooks. Multi-processor computers also changed the way programs are written due to the power of the distribution of work on a multi-core processor, most programs now are written with parallel code rather than sequential, see Figure 1.12.

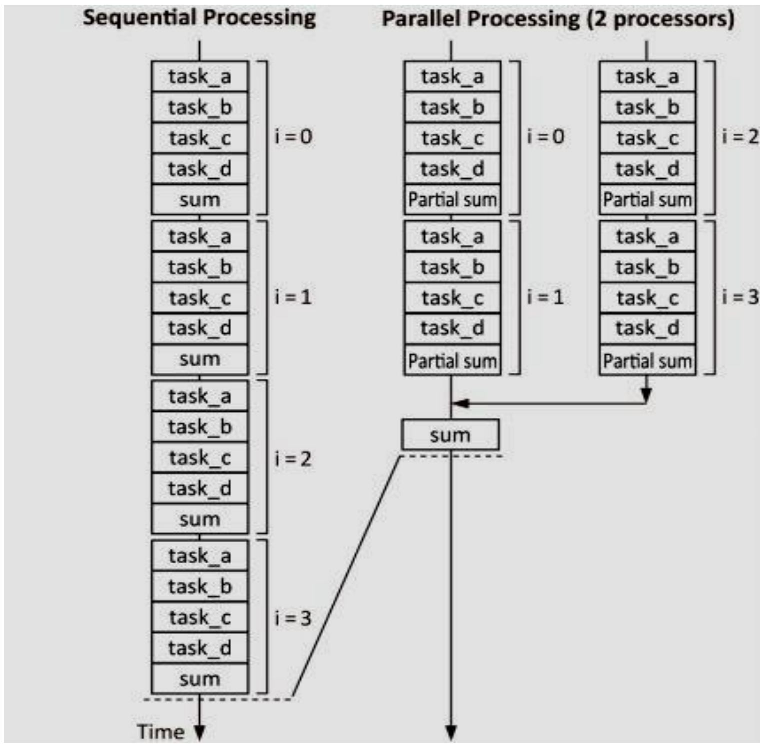


Figure 1.12: sequential processing vs. parallel processing [20].

Parallel processing is the processing of programs using multi-processor computers to speed up running time. Parallel processing is different from concurrent processing. In concurrent processing, different computations can be executing during overlapping time period using for example time slices, like running internet browser and Microsoft word at the same time without waiting one of them to finish. Concurrent processing can be done on a one core single processor.

1.2.1 Parallel computers:

Flynn's Taxonomy [21] is one of the most famous computer classifications. According to Flynn computers are classified into 4 types SISD, SIMD, MISD and MIMD [21]. Single Instruction, Single Data (SISD) is the oldest type of computers where the CPU run only one instruction and use one data stream as input in one clock cycle, see Figure 1.13. Single processor computers are example of the SISD computers.

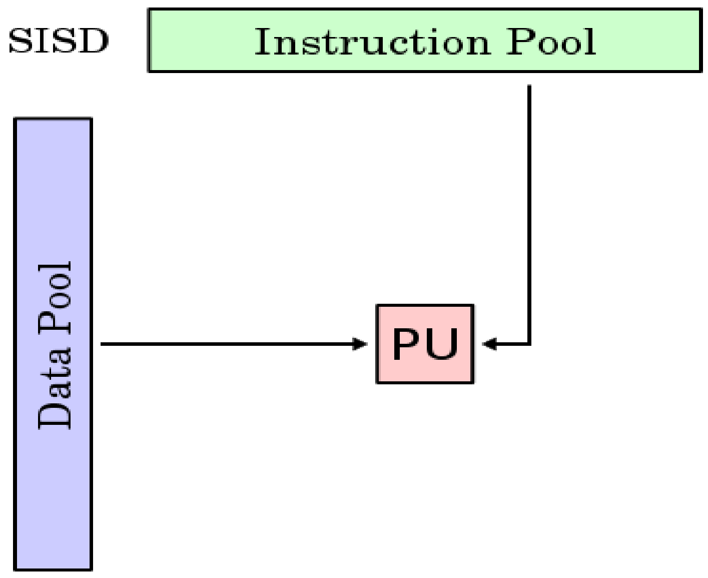


Figure 1.13: SISD computers [21].

In Single Instruction Multiple Data (SIMD) computers, all processor units execute the same one instruction but for different set of data in one clock cycle, see Figure 1.14. Most of recent devices which come with graphic units is SIMD type.

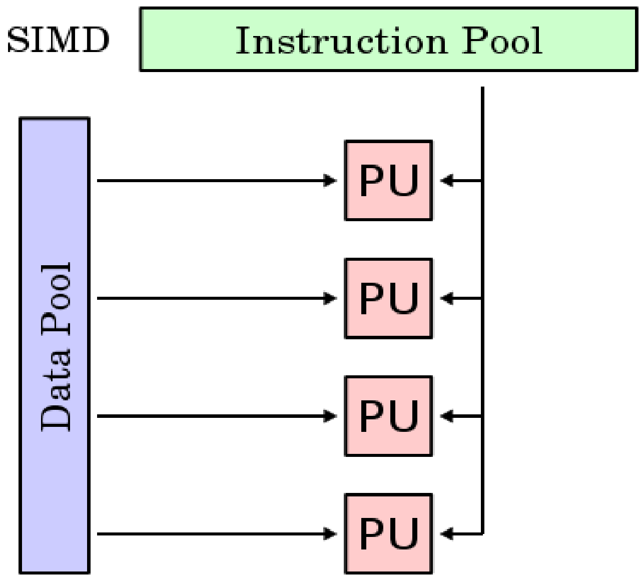


Figure 1.14: SIMD computers [21].

In Multiple Instruction Single Data (MISD), the processing unit executes different instructions but for the same set of data in one clock cycle as shown in Figure 1.15. Few devices use this type of computers.

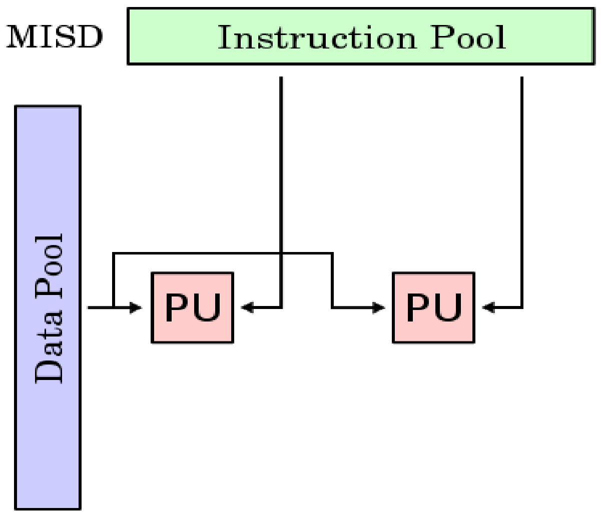


Figure 1.15: MISD computers [21].

In Multiple Instruction Multiple Data (MIMD), any processor can execute any instruction on any set of data in one clock cycle as shown in Figure 1.16. Most of recent computers are of type MIMD, super computers and multi-core PCs are an example of MIMD type.

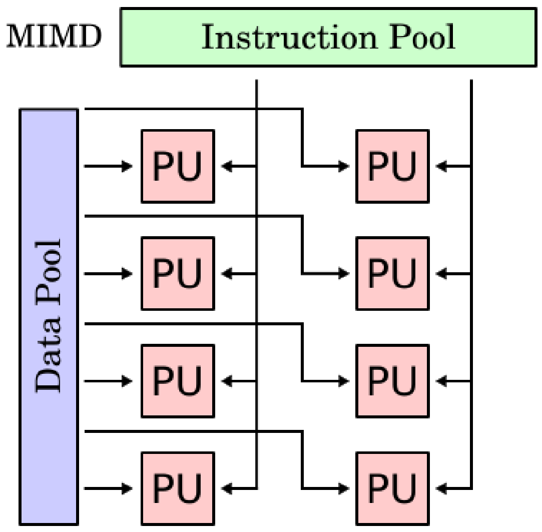


Figure 1.16: MIMD computers [21].

1.2.2 Multi-core processors

A multi-core processor is a processor that combines two or more execution cores within a single processor package. The advantages of using multi-core are higher throughput with low frequency hence low temperatures and improving the performance of the work. A single-core chip and a multi-core chip are shown in Figure 1.17 and Figure 1.18 respectively [22].

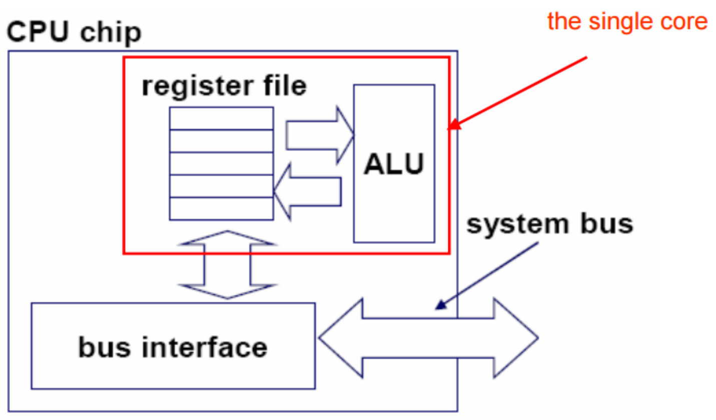


Figure 1.17: Single-core CPU chip [22].

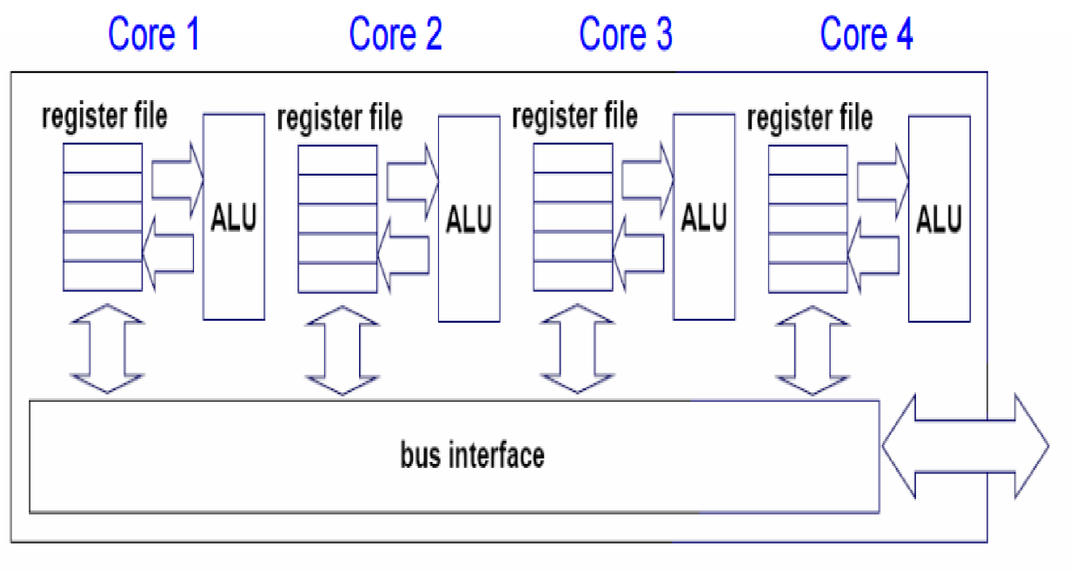


Figure 1.18: Multi-core CPU chip [22].

1.2.3 Hyper threading

Hyper threading technology enables single core to appear as a multiple logical processor to serve multiple threads. The logical processors share the same physical execution resources but have separate register files. Figure 1.19 and Figure 1.20 shows a multi-processor with and without hyper-threading technology respectively. In Figure 1.19 the processor has two physical cores, but in Figure 1.20 the processor has 4 logical cores (2 physical cores with hyper threading) [23].

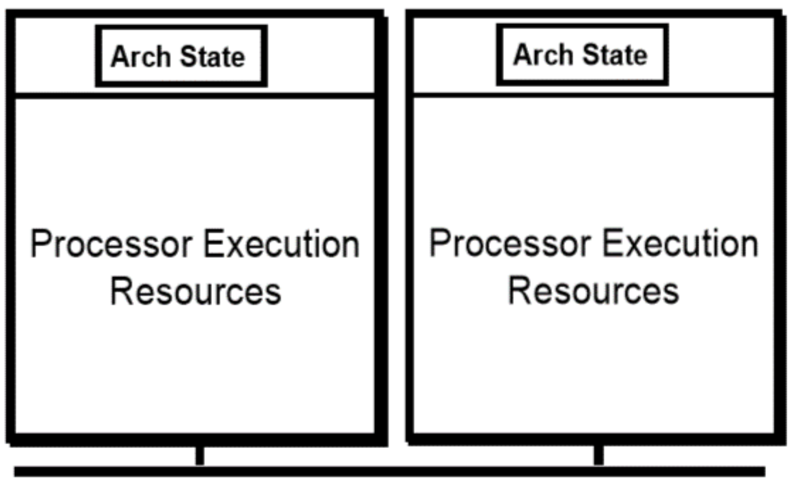


Figure 1.19: Processors without Hyper-Threading Technology [23].

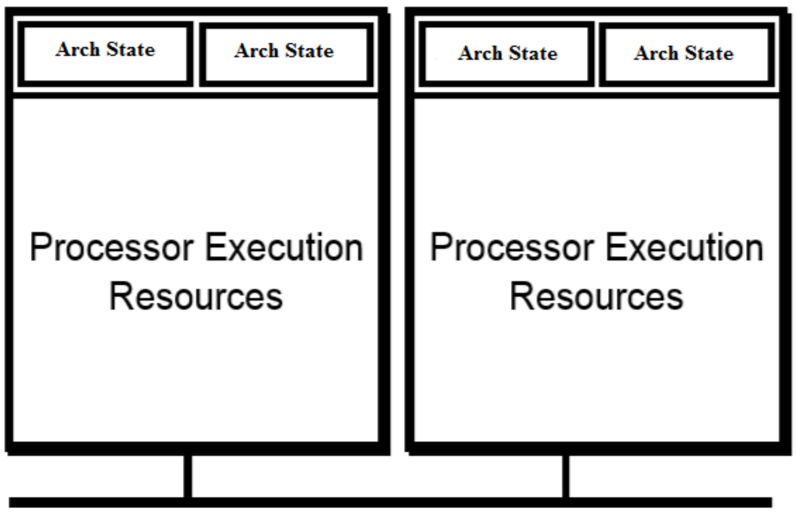


Figure 1.20: Processors with Hyper-Threading Technology.

In a processor without hyper threading there will be some resources not used and set idle when the processer is using one of the resources, but when using hyper threading the other threads can use the other resources at the same time. This will make the processor do more work at the same time and give better performance, Figure 1.21 Shows the difference between the process when using hyper threading and not using it.

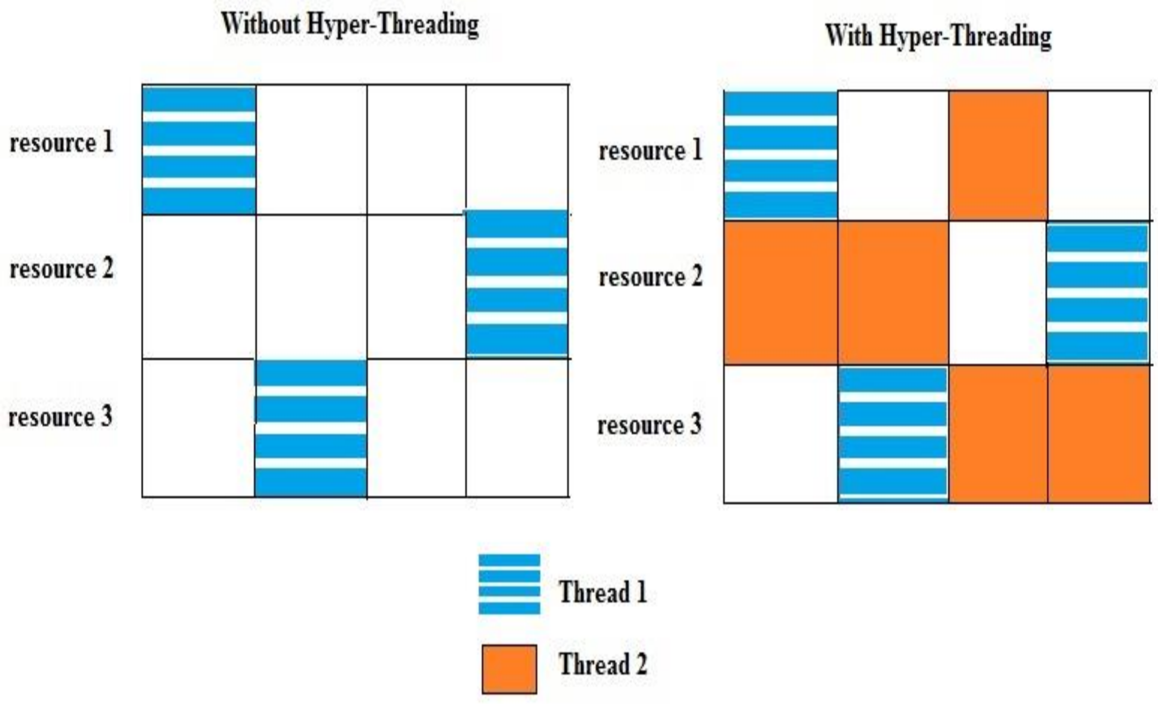


Figure 1.21: The process of the threads with and without hyper threading.

Shared resources consist of different execution units like the integer units, load-store units and floating points units. The efficiency of using hyper threading increases when the shared resources increase. Using shared resources on a small amount of die space reduce the computer power needed.

A comparison on the performance between the case of using hyper threading technology and the case where it is not used is shown in Figure 1.22 [23]. The processor used here is Intel Xenon.

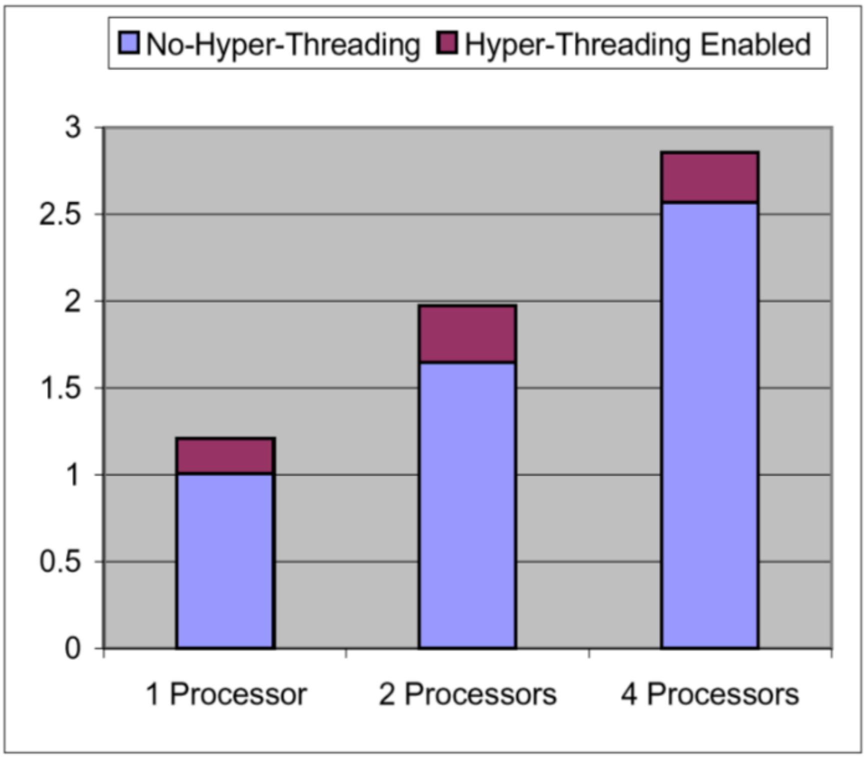


Figure 1.22: Performance increases from Hyper-Threading Technology [23].

Recently most of Intel processors are multi-core processors. Core-i series is one of the latest Intel product of processors. Many different types of core-i series produced with different number of cores such as dual, quad and octa cores. Table 1.2 shows some of these processors and their specifications.

Processor number	**Cache**	**Number of cores / number of threads**
Intel Core i7-7700	8 MB	4/8
Intel Core i7-7500U	4 MB	2/4
Intel Core i7-6770HQ	6 MB	2/4
Intel Core i5-670	4 MB	4/8
Intel Core i3-540	4 MB	2/4
Intel Core i7-5960X	20 MB	8/16

Table 1.2: Different Intel processors with different number of cores [24].

1.2.4 Parallel computers memory architectures

Parallel computers memory architectures are generally divided into two types: shared memory and distributed memory. In shared memory, all processors can access all memory as a global address space. It is classified as uniform memory access (UMA) and non-uniform memory access (NUMA), see Figures 1.23 and 1.24.

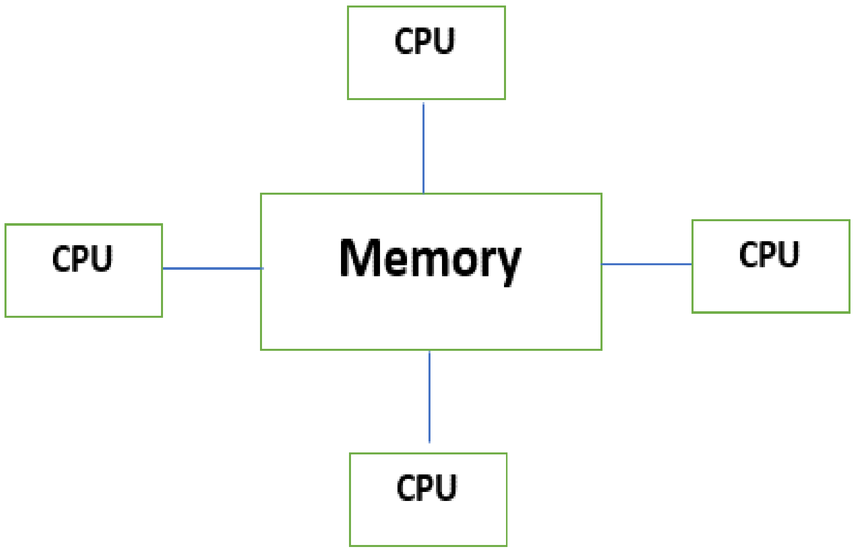


Figure 1.23: UMA Shred memory.

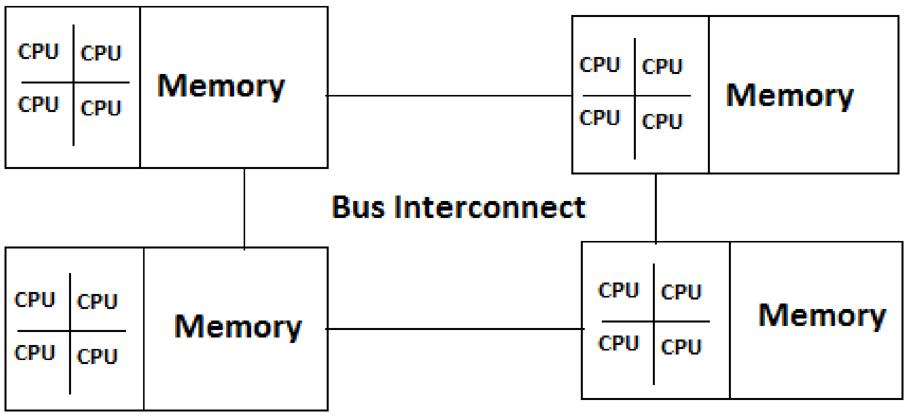


Figure 1.24: NUMA shared memory.

In distributed memory architecture, each processor has its own local memory. Any change in the local memory will not affect the memories of other processors. The advantages of the distributed memory architecture are that the memory increases with the number of processor and that each processor can access the memory with no worry of affecting other processors. On the other hand, the disadvantages of the distributed memory architecture are the difficulty of mapping existing data structures and that the programmers will have more responsibility of data communications between processors. Figure 1.25 shows the architecture of the distributed memory.

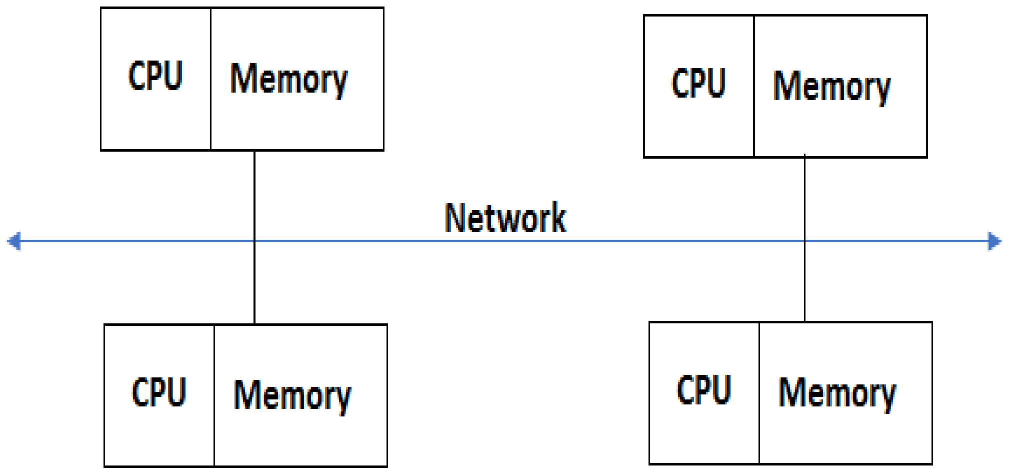


Figure 1.25: Distributed memory architecture.

1.2.5 Parallelization techniques

A multi-processor system contains more than one processor working in parallel on a special motherboard which has several CPU sockets. A multi-core system contains more than one execution core on the same CPU chip, a certain subset of the CPU's components is duplicated, so that they can work in parallel.

In parallel processing, we need a multi core/processor device and distribute the tasks or the data of the program to execute them at the same time on different cores or processors in order to reduce the execution time and power consumption. This can be done using data parallelism or task parallelism, see Figure 1.26.

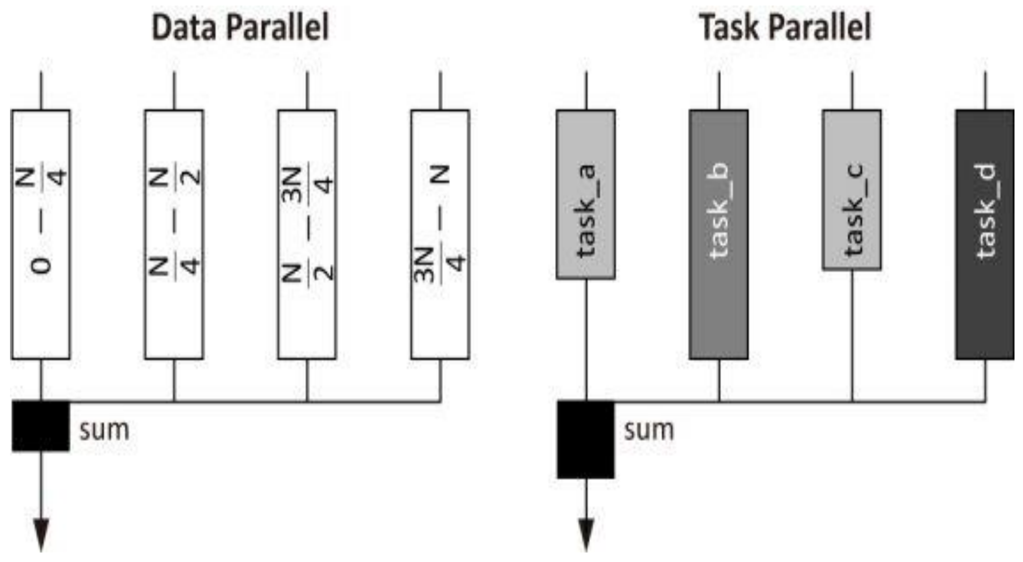


Figure 1.26: Data parallelism vs. Task parallelism [20].

Data parallelism is where we focus on distributing the data in a way that each processor will do the same task but for a different piece of the original data. Many image processing algorithms can be done using data parallelism by splitting the image into smaller images and assigning them to different processors to process the same task simultaneously.

Task parallelism is where we focus on distributing the tasks to run on different processors to process the same or different data at the same time. Some image processing algorithms can be done using task parallelism. For example, by distributing the tasks of a video stream consisting of many frames, where each frame has to undergo two image processing tasks, then by using two processors in a pipeline architecture, we can make sure that each of the two processors is executing one task per frame at the same time that the other processor is executing the second task on the previous frame. If the two tasks are equally balanced, then almost 2x speedup can be achieved.

1.2.6 Parallel programming tools

In the early times of parallel programming, the programmer had to do the parallelization manually, He had to manually divide the work and distribute it to the available processing unit and also manage the threads to use more than one processing unit. Recently, many frameworks and techniques are developed to help the programmer to do this work. Loop unrolling, affine loop transformations or loop tiling are examples of parallelization techniques [25, 26, 27]. Cilk++, OpenMP and OpenCL are examples of frameworks used for parallelization. A comparison between different frameworks execution time for an image of volume 256×256×189 registration is shown in Figure 1.27 [28].

In this research the framework that will be used is Cilk Plus. Cilk Plus is a commercial version of Cilk language and it is developed by Intel and now is known as Intel Cilk Plus. Cilk plus is developed as an extension of the C++ language with three main keywords, to allow the programmer to do the parallelization work and manage the tasks and synchronization between them. Data race problems happen in the case of accessing global variables without locks so hyper-object is provided by Cilk plus to avoid these problems. Cilk plus provides two compilers for code generation: MSVC for Windows and GCC for Linux. This framework also comes with a race detector and a performance analyzer to detect race condition and to calculate the speed up time [29,30].

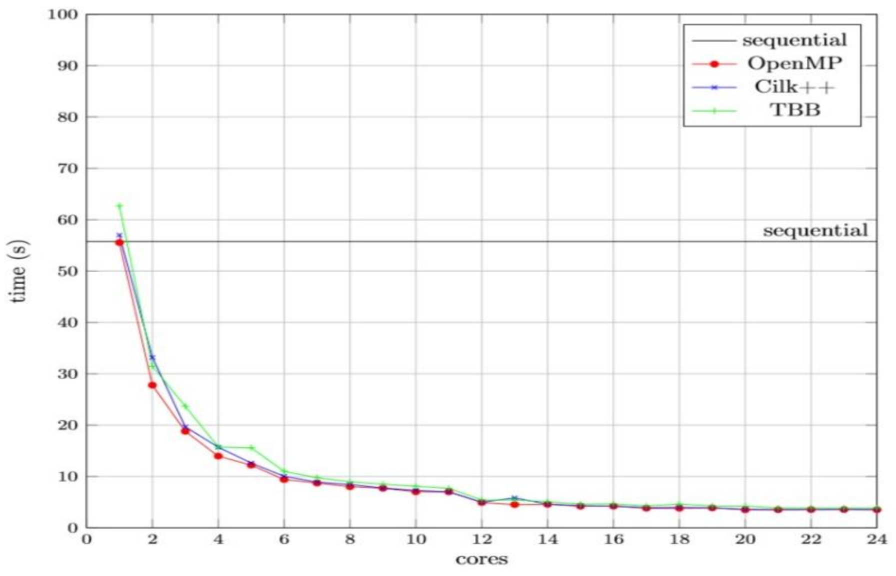


Figure 1.27: Execution times of all parallelization frameworks for the implementation on up to 24 cores for a volume of 256×256×189 voxels compared with the sequential implementation [28].

1.3 Contribution of the study

The saw-tooth fringe projection technique was not parallelized before, and in this work we will use parallelization and speed enhancement techniques to make it faster by converting the MATLAB code to C/C++ code and by using Cilk Plus framework for parallelization. The purpose of enhancing the speed of this technique is to use it in applications where the execution time of the 3D measurements is very important, such as human-to-computer interaction, computer graphics, healthcare, and the manufacturing industry.

1.4 Literature review

The approach of this thesis is to use the power of parallel processing to minimize the processing time of the saw-tooth fringe pattern 3D measurement method. There is a lot of research on 3D measurement using different methods. In this section, we will discuss some of the related sources to our work. We will also discuss some parallel processing research which is a very important part of this thesis.

1.4.1 3D shape measurement literature review

P. Cao et al [18] proposed a method to extract the 3D shape of an object using triangular fringe patterns and spatial shift estimation. In their system, they used digital video projector, a reference plane and a CCD camera, see Figure 1.28. They casted a special fringe pattern consisting of a frame of images onto the reference plane, and then they removed the reference plane and casted the pattern again onto the object surface. The CCD camera used in this system captured the light reflected from both the reference plane and the object's surface. The fringe pattern which is casted onto the surface of the object will be deformed due to the variance of the height of the surface of the object. Finally, they measured the 3D shape of the object using these two patterns based on triangulation.

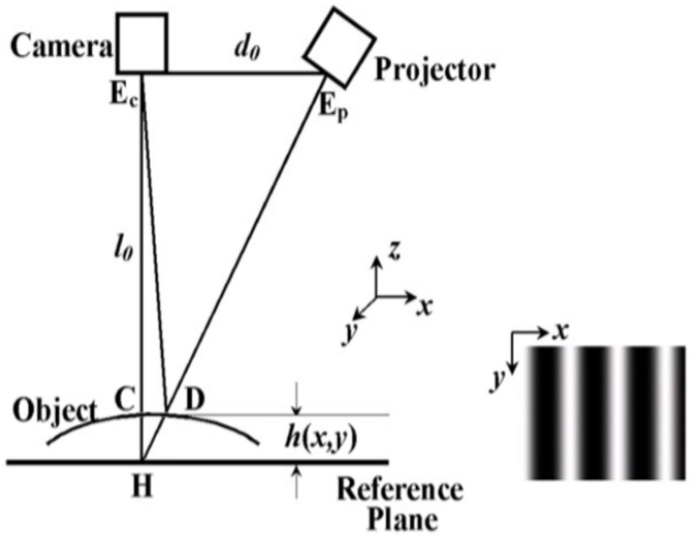


Figure 1.28: Schematic diagram of FPP system [18].

Another method is to use a white plate with hollow ring markers separated with exact distance on the surface, see Figure 1.29. The center position of the markers is automatically calculated. The camera parameters can be standardized using the marker locations. The CCD camera parameters can be determined from the locations of the marker. The 3D position of the plate is calculated by using the acquired parameters of the camera and the separation between markers. The fringe patterns are casted onto the surface at all plate position to extract the phase information. Then the relationship between the phase and the depth data of the pixels can be figured out by calculating the coefficients of a polynomial function depending on the phase and the depth of each pixel for the all plate positions. The transverse X and Y coordinates of each pixel position are also adjusted using the parameters of the camera and the matching depth data. From the internal parameters of the camera and the depth data they determined the crossed X and Y coordinates for each pixel. Using the previous steps, the relationship between phase map and 3D shape data can be determined and it is very simple and flexible way [31].

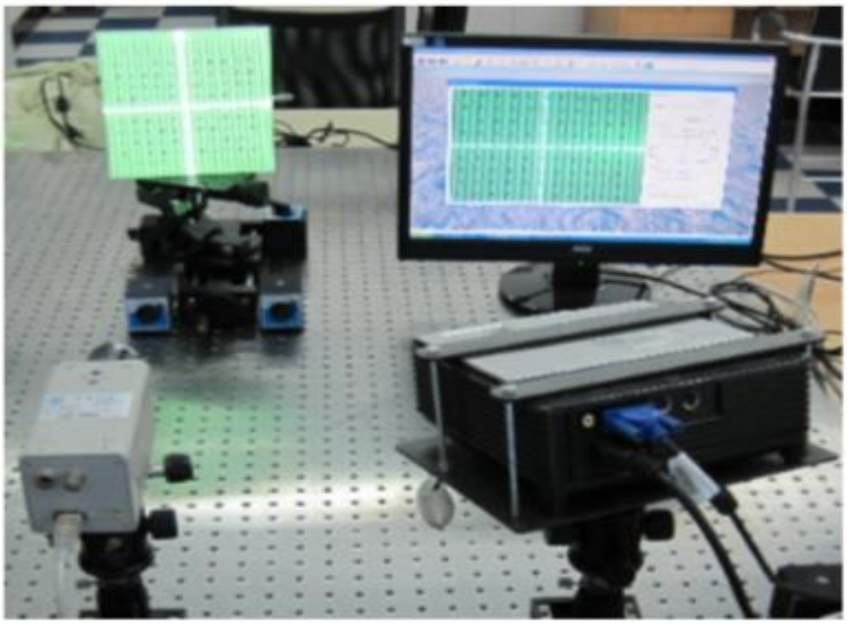


Figure 1.29: The hardware setup of the fringe projection imaging system including a DLP projector, a color 3CCD camera and a personal computer [31].

Lujie Chen et el [32], proposed a method of shape measurement using one frame projected saw-tooth fringe pattern. In this method, they used one frame only, a right-angle triangle saw-tooth fringe pattern. No phase-shifting or Fourier transformation used in this method, a wrapped phase map obtained from a linear translation of the recorded saw-tooth pattern used. To retrieve the unwrapped phase map, they used a quality-guided phase unwrapping algorithm.

The system setup used in this method is simple, see Figure 1.30. A computer generates a saw-tooth fringe pattern, the generated image is projected on the object surface using a LCD projector. In order to focus the projected saw-tooth pattern onto the surface of the object, a camera lens is mounted on the exit pupil of the projector. The CCD camera capture the images of the surface and the object and both images are stored in the computer for the further processing. The angle of projection used in this method is 25 degree. The maximum and minimum intensity used are 230 and 30 respectively [32].

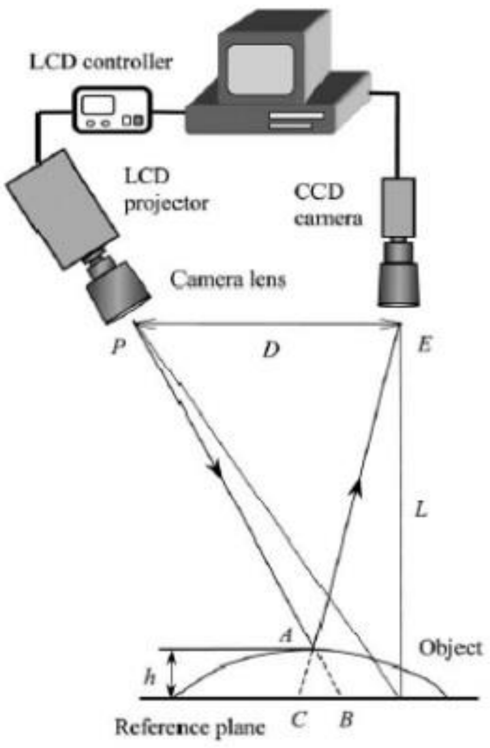


Figure 1.30: Optical geometry of the projection and imaging system [32].

Time of flight (TOF) is a popular method in 3D shape measurement. In this method, we can calculate the distance by calculating the time needed for a light pulse to travel from the source to the surface of the object using the relationship between the speed of light (c = $3x10^8$ m/s) and the time it takes to make the travel. The TOF system is described in Figure 1.31.

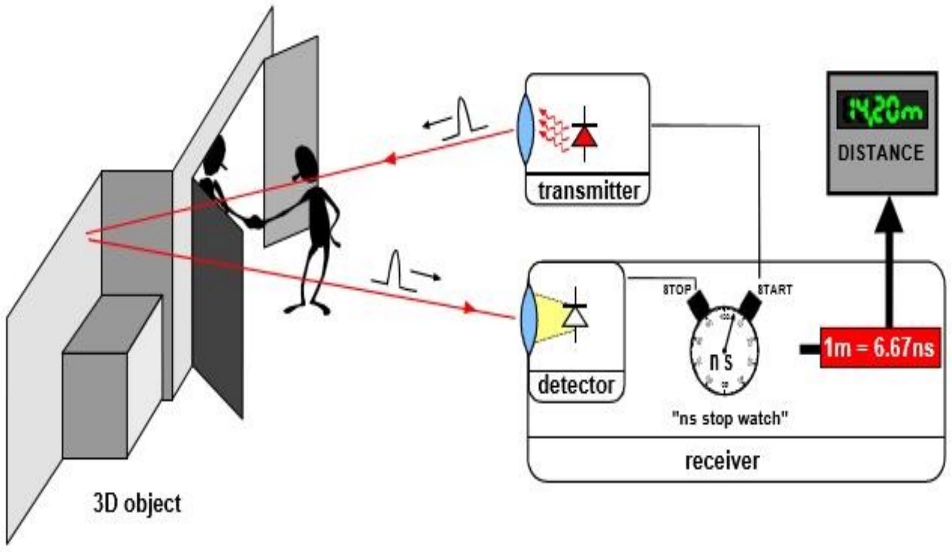


Figure 1.31: Basic principle of an (optical) TOF ranging system [33].

As shown in Figure 1.31, a light source emits a light pulse (usually from a laser source) onto the object and a high accuracy watch is started in synchronous with the light. The light travels to the object and back, at the time that the detector detects the light the watch stops. The watch will show the time of the travel of the light pulse from the source to the object surface. Finally, the distance is measured using the relationship between the speed of the light and the travel time [33]. In this technique, the scanner has to take many points in order to accurate enough data to construct a 3D shape.

Fourier transform profilometry (FTP) is a widely-used method in 3D shape measurement. Xianyu et al [34] proposed a method based on FTP for measuring 3D shapes of objects.

The equipment used in this method are CCD camera, projector with phase shifter and surface as shown in Figure 1.32.

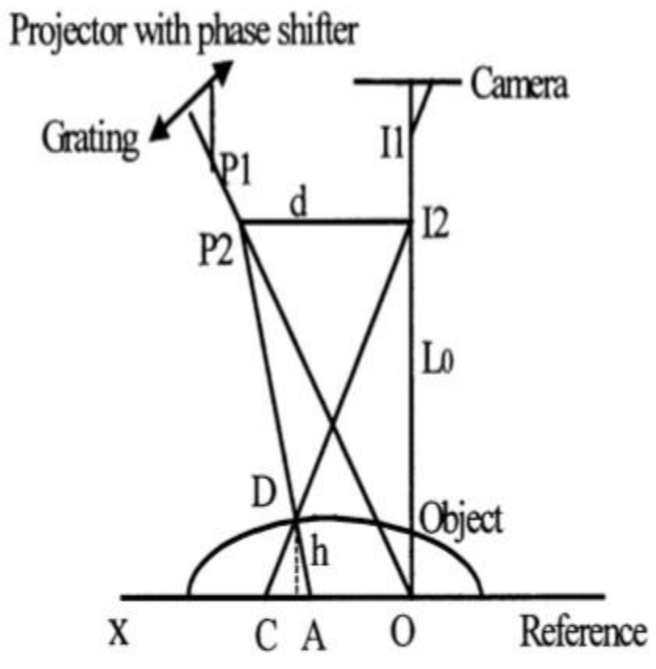


Figure 1.32: Optical geometry [34].

A Ronchi or sinusoidal grating is projected onto a dynamic 3D surface. The CCD camera captured a sequence of deformed fringe image, these images are saved on a disk. The phase-maps of these images are obtained using Fourier transform, filtering and Inverse Fourier transform. By unwrapping these phase maps in 3-D phase space, we can obtain the shape of the dynamic object at different times.

Al-Hiary [1] used saw-tooth fringe pattern to measure the 3D shape of objects. In this system, a camera, a digital projector, and a PC were used. The saw-tooth fringe pattern which is shown in Figure 1.33 was generated by the PC and casted onto both the object and the reference plane using the digital projector.

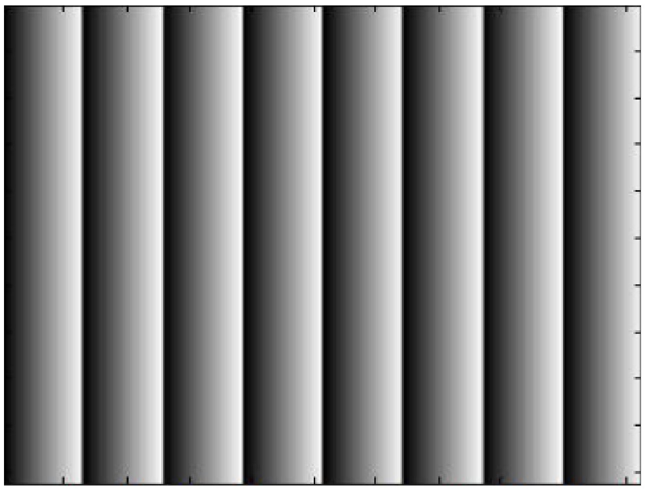


Figure 1.33: Saw tooth fringe patterns [1].

The camera captured the light reflected from both the reference plane and the object surface and sends them to the PC. Using the information carried by the deformed fringe pattern reflected from the object, and using triangulation geometry calculations, the 3D shape of the object is reconstructed. The problem with this method is the averaging of borders pixels at edges between fringe periods that is inherited in digital cameras. Also, this method needs some noise removal techniques which increase the processing time.

1.4.2 Parallel processing literature review

Rye et al [35] used parallel processing to improve the speed performance of the computation of large 1 dimensional discrete fast Fourier transforms (DFFTs). The processor used in the work was multi-core Intel Xeon. In this method, they used Intel Cilk Plus framework for parallelism and a single threaded Intel Math Kernel (MKL) library implementation of discrete fast Fourier transforms.

The work result is a new library called EFFT (Enormous Fast Fourier Transforms). EFFT library performance on one-dimensional DFTs of size 2N for N>20 is 1.5x faster than the MKL and 2.5x faster than the FFTW (Fastest Fourier Transform in the West) [36], this improvement of the speed performance has no effect on the accuracy and need less memory.

Sudipta et al [37] used parallel processing in video feature tracking and matching. In their system, they used Kanade-Lucas-Tomasi (KLT) for feature tracking and Scale Invariant Feature Transform (SIFT) for feature extraction [38,39]. They also used graphic processing unit (GPU) which is very suitable for real-time visions. OpeGL graphic library and CG shading language are used to implement both the KLT and SIFT on GPU. The graphic card used in this system is NVIDIA (7800GTX, 7900GTX). Using this high performance and multi-core GPU instead of the CPU, the results showed a very high improvement in the speed of processing. The results of GPU-based KLT and GPU-based SIFT is 20 and 10 times faster than the CPU implementation respectively, see Figures 1.34 and 1.35.

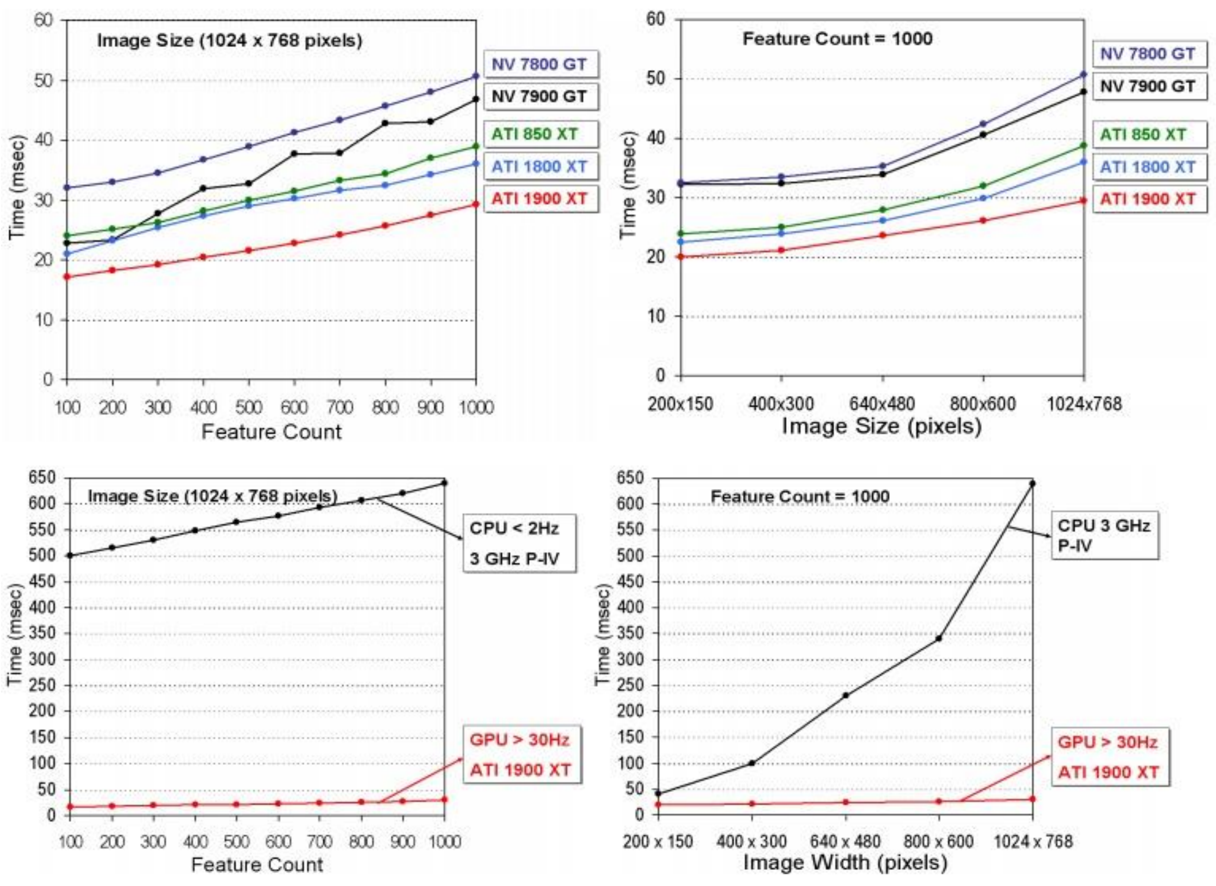


Figure 1.34: A comparison of GPU-KLT timings on various graphics cards (Center) and between GPU and CPU [37].

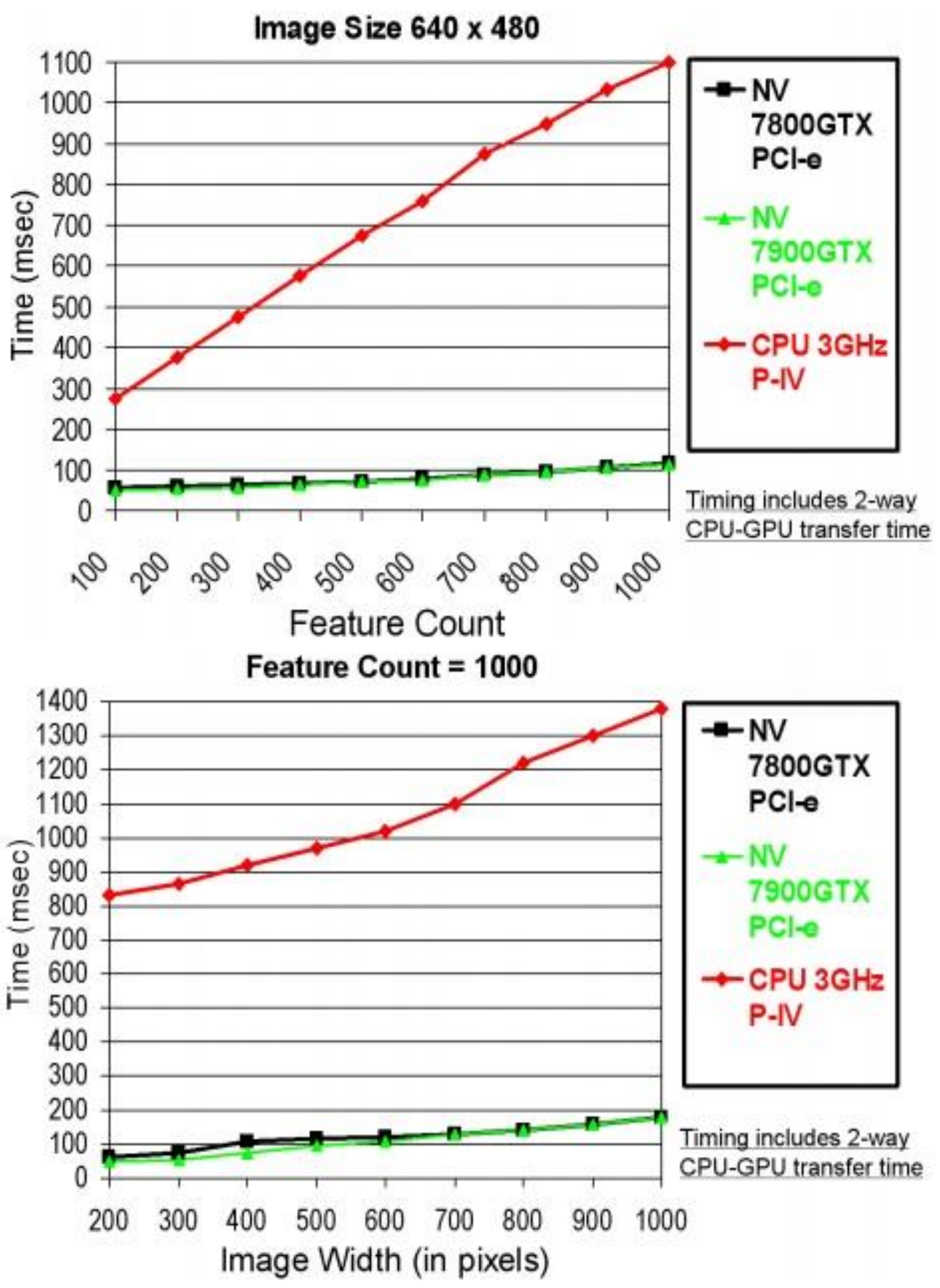


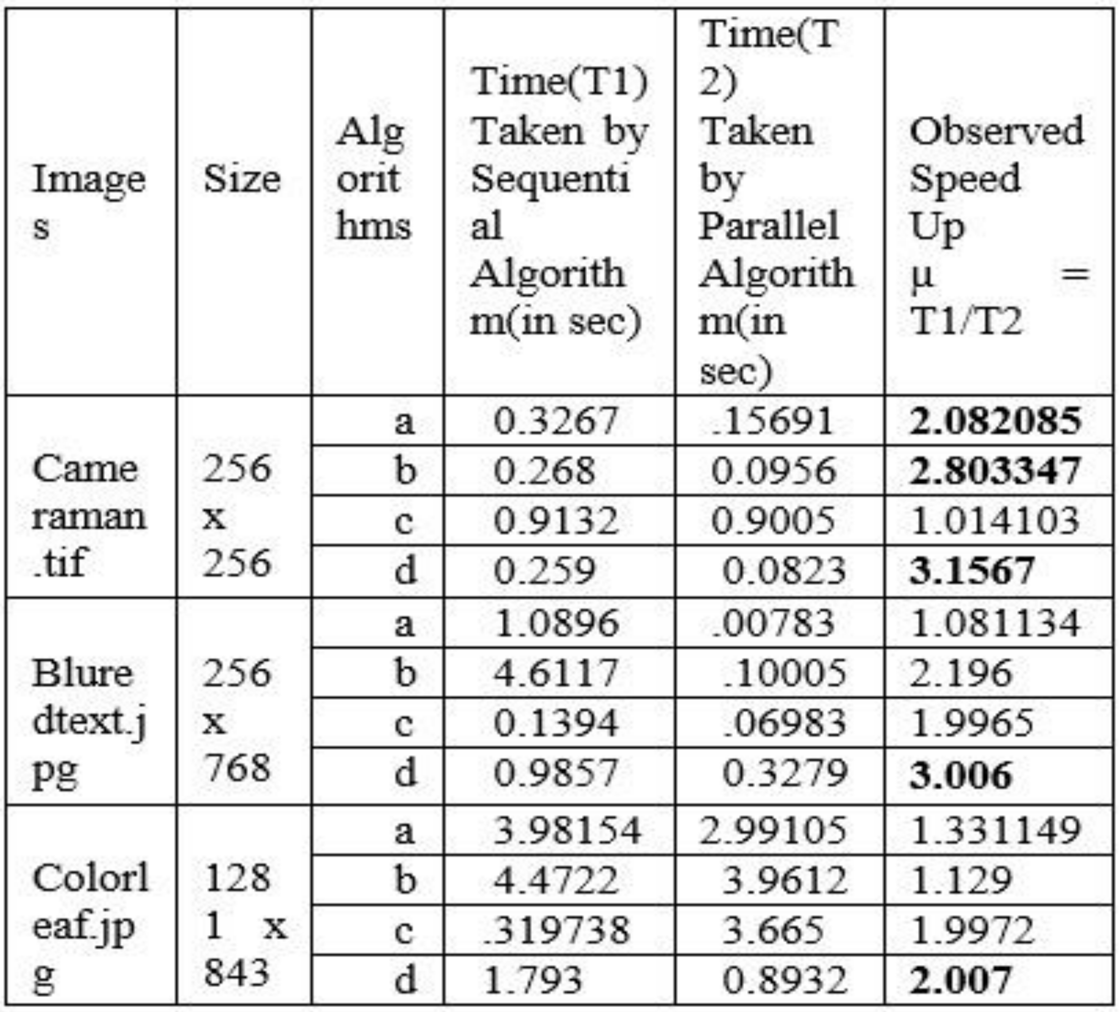
Figure 1.35: GPU-SIFT timings compared with an optimized CPU implementation for a range of image-sizes and feature-counts. GPU-SIFT has a 10-12X speed-up [37].

Sanjay Saxena et el [40], published a paper about image processing tasks using parallel computing. In this paper the authors gone through different sequential algorithms of image processing like histogram equalization, complex noise reduction and global thresholding and calculation of Fourier and regional descriptors of an image. They developed these algorithms to a parallel version to enhance the speed of processing.

The strategy of their work was to divide the original image into four small images and distribute the work to the threads of the processor and process each part of the image at the same time.

The processor of the computer used in this research was core i3-2350M with 2.30 GHz, the size of the RAM and the hard disk was 3GB and 320GB respectively [40].

The result of this research is good. The speed up after the parallelization was about two and half times faster. Table 1.3 shows the results of each algorithms used.



Images	Size	Algorithms	Time(T1) Taken by Sequential Algorithm(in sec)	Time(T2) Taken by Parallel Algorithm(in sec)	Observed Speed Up μ = T1/T2
Cameraman.tif	256 x 256	a	0.3267	.15691	**2.082085**
		b	0.268	0.0956	**2.803347**
		c	0.9132	0.9005	1.014103
		d	0.259	0.0823	**3.1567**
Bluredtext.jpg	256 x 768	a	1.0896	.00783	1.081134
		b	4.6117	.10005	2.196
		c	0.1394	.06983	1.9965
		d	0.9857	0.3279	**3.006**
Colorleaf.jpg	1281 x 843	a	3.98154	2.99105	1.331149
		b	4.4722	3.9612	1.129
		c	.319738	3.665	1.9972
		d	1.793	0.8932	**2.007**

Table 1.3: Performance evaluation table [40].

Parallel processing is used in numeric weather predictions. A study made about the required time for the complex numerical simulation of the weather on multi core processors [41]. The study focused on the serial and parallel computing of the weather forecasting model (WRF).

Figure 1.36 shows the results of the study using a quad core machine. The serial execution took huge time but less time in parallel mode. Different variables made a variation in the speed result like the memory location and RAM speed. As we can see in Figure 1.36 the speed up is almost the same when using 4 or more cores.

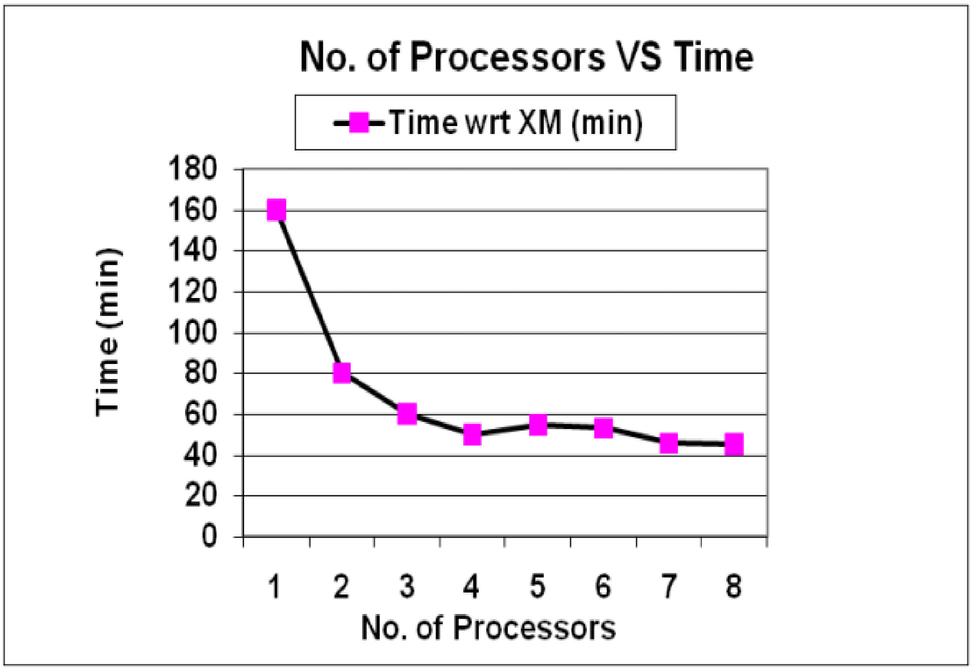
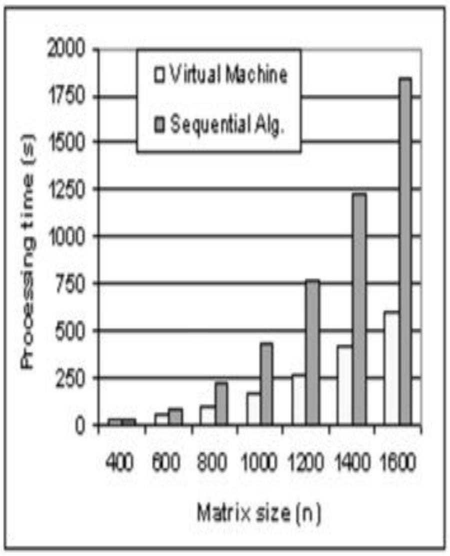


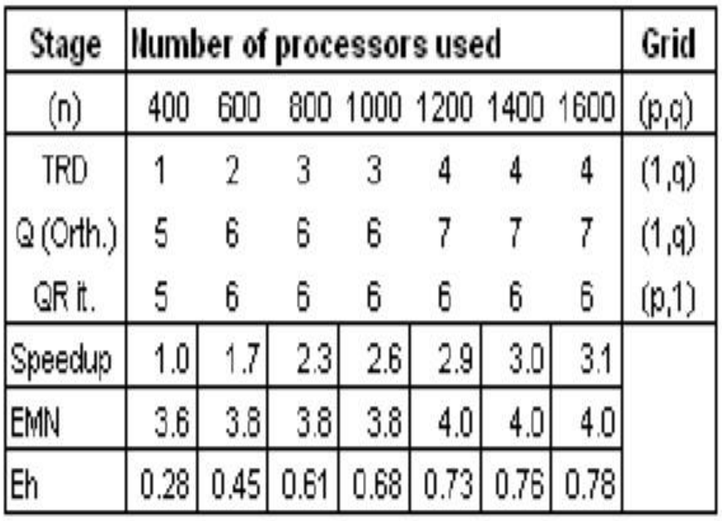
Figure 1.36: WRF model simulation time corresponding to the number of processors by using quad core machine.

Another method of image parallel processing done using a cluster of personal computers [42]. In this research a group of computers in a local area network used to do offline image processing. The computers were connected using low cost network such as 10

Mbits/s Eathernet. The aim of the research was to minimize the processing time of image processing tasks. Figure 1.37 shows the results of using parallel processing method against the sequential method.



Computation time



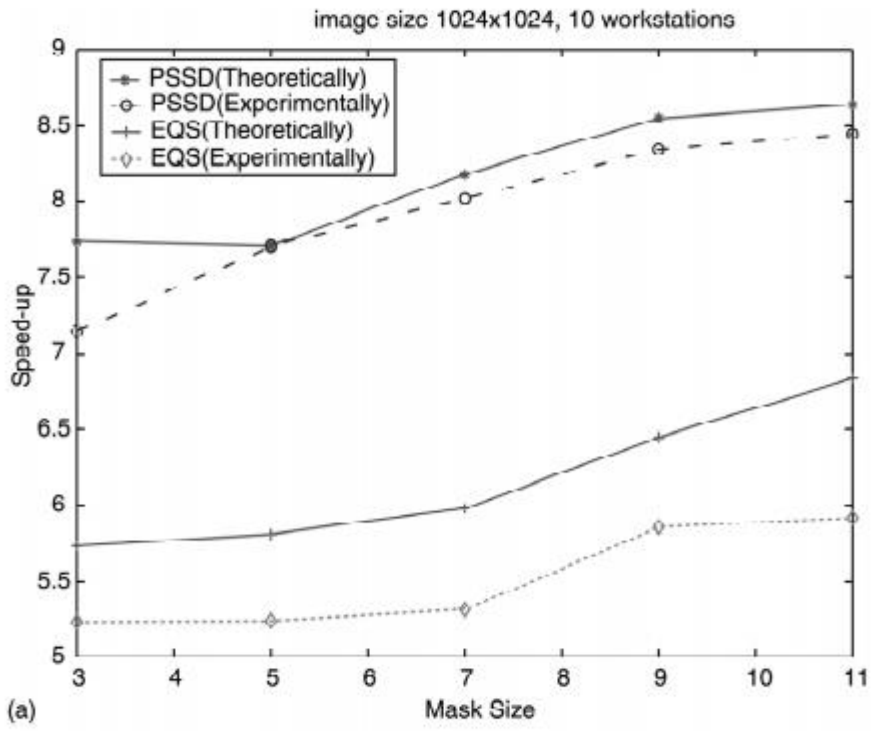
Stage	Number of processors used							Grid
(n)	400	600	800	1000	1200	1400	1600	(p,q)
TRD	1	2	3	3	4	4	4	(1,q)
Q (Orth.)	5	6	6	6	7	7	7	(1,q)
QR it.	5	6	6	6	6	6	6	(p,1)
Speedup	1.0	1.7	2.3	2.6	2.9	3.0	3.1	
EMN	3.6	3.8	3.8	3.8	4.0	4.0	4.0	
Eh	0.28	0.45	0.61	0.68	0.73	0.76	0.78	

Processing results

Figure 1.37: Eigenvector computation in the M2 machine [42].

X. li et el [43], proposed a research about using a network of stations for image processing. The approach was to divide the work and distribute it to the computers of the stations to achieve the minimum processing time.

They used a parallel virtual machine library and two different scheduling and partitioning strategies. The first strategy is the partitioning and scheduling following divisible load theory (PSSD). The second strategy is the traditional equal-partitioning EQS [43]. The experiment done using different number of nodes, different kernel size and different number of work stations. Figure 1.38 shows a comparison of the speeding up between PSSD and EQS strategies.



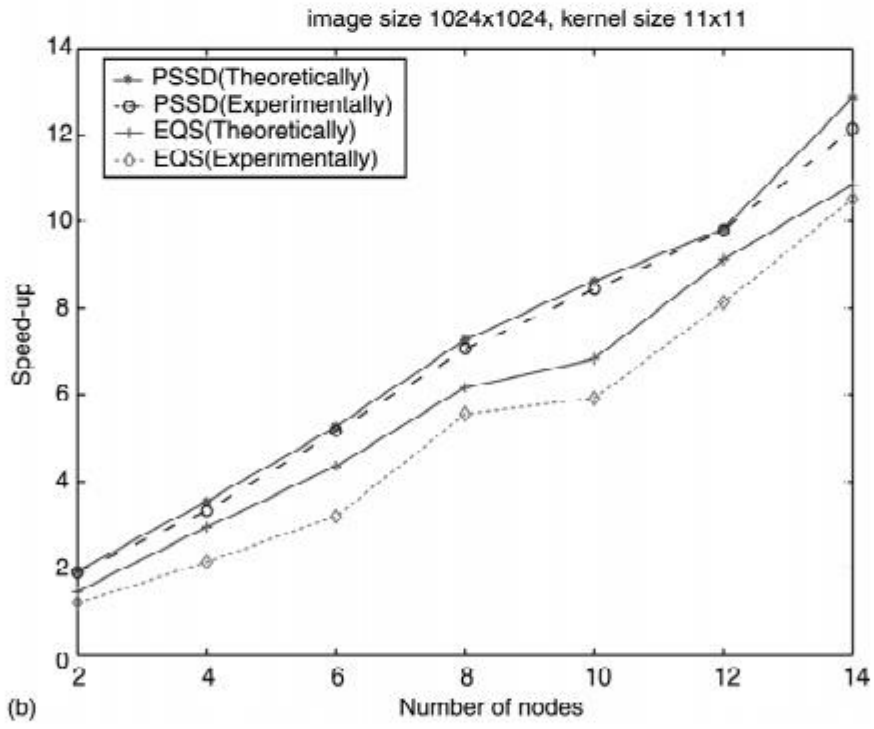


Figure 1.38: Comparison of speed-up between PSSD and EQS. (a) Speed-up versus kernel size; (b) speed-up versus number of nodes.

CHAPTER 2

The Saw-tooth Fringe Projection Technique

In this chapter, we will discuss the saw-tooth 3D profilometry technique. First, we will discuss how the fringe patterns are generated and projected. Then we will talk about the setup of our system and how we captured the images. After that the images enhancement and noise removal methods will be discussed. Finally, we will show the triangular calculations and the height results and the 3D reconstruction of the objects.

2.1 System analysis:

In our system, we use a computer to produce the fringe patterns, and then we use a projector to project the fringes into the reference plane and the object surface. After that we capture the images we need using a digital camera, see Figure 2.1.

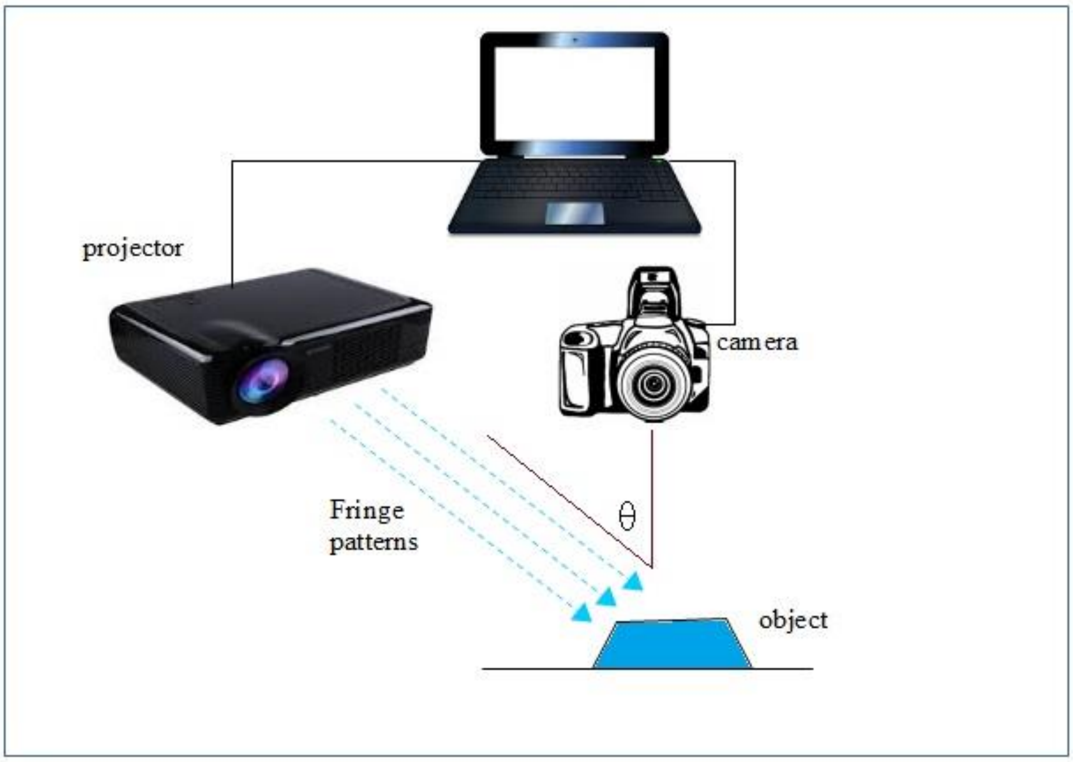


Figure 2.1: System setup.

We capture two images for the reference plane and the object to eliminate the non-linear distortion caused by the projector and the camera. We apply image enhancement and noise removal algorithms to the images, then we calculate the heights using triangular equations. Figure 2.2 show our system main steps of work.

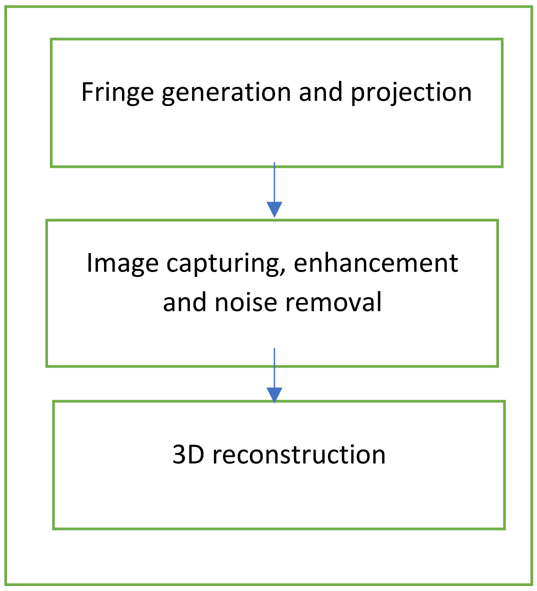

Fringe generation and projection
Image capturing, enhancement and noise removal
3D reconstruction

Figure 2.2: 3D profilometry work steps.

2.2 Fringe generation:

The fringe pattern we use in this system called saw-tooth, it is called by this name because its waves are like the teeth of the saw. AS shown in Figure 2.3, the intensity start increasing from the minimum value to the maximum value for each period, then it goes sharply from the maximum value of the period to the minimum value of the next period. The fringe pattern function is generated by equation 2.1.

$$s(x) = a.\left(\frac{x}{T_0} - floor\left(\frac{x}{T_0}\right)\right) \quad x \geq 0 \qquad \textbf{(2.1)}$$

S(x) is the function of the saw-tooth waveform, floor is a function which maps real numbers. T0 is the period and a is the amplitude. Figure 2.3 show the waves of a saw-tooth function with amplitude=255 and period=16 pixels wide.

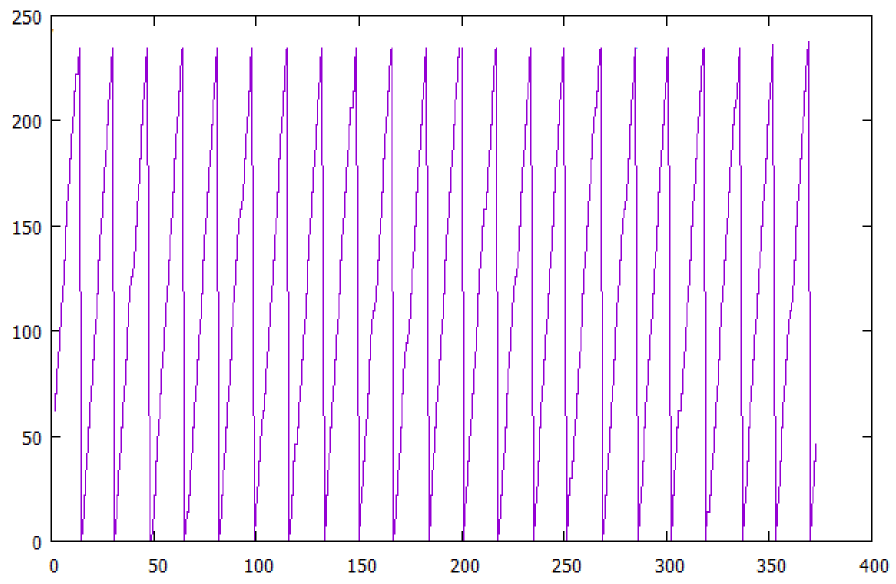


Figure 2.3: Saw-tooth waveform with amplitude=256 and period=16.

To generate the saw-tooth fringe pattern we mix the three colored channels: green, blue and red of the same amplitude and period and then convert it to the gray level as shown in Figure 2.4. Saw-tooth are also simple, monotonic and adjustable for periodic.

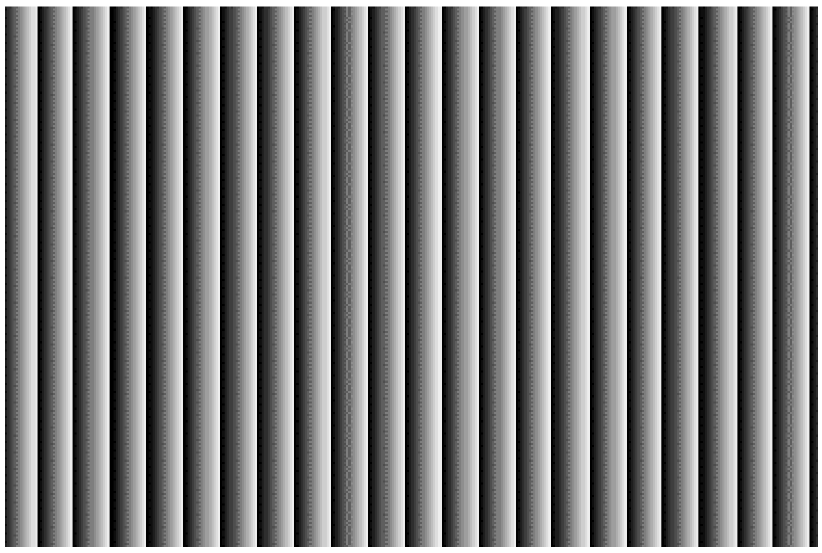


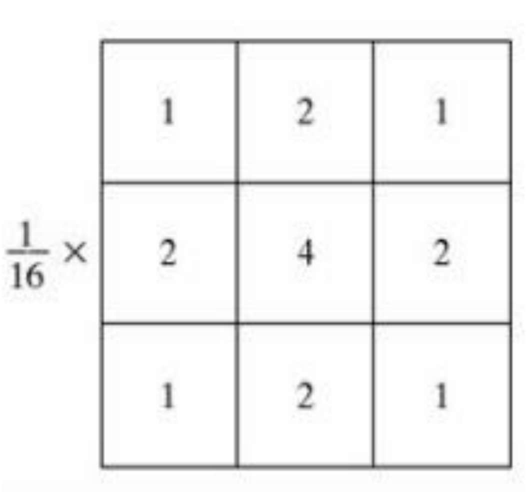
Figure 2.4: Saw-tooth fringe pattern.

2.3 Image processing:

The captured images are affected by different type of noise such as electric noise of the projector and camera, and the non-linear light reflection. In order to reduce noises in images and get a better result we use image enhancement and noise removal methods. Median and mean filters are one of the most common filter used in image processing.

Mean filter: It is a spatial filter, it replaces the center value of group of pixels with the average of the values of all the neighborhood pixels. Neighborhood pixels are chosen using a window or kernel. The window can be any shape, but most of the time it is square, Figure 2.5 shows an example of 3x3 window of a mean filter. If all the weights of the box are equal, then it called box filter. Mean filter smooth the image and reduce the variation in the intensity of neighborhood pixels. Equation (2.2) is used to find the average of the box values.

$$\text{Mean}= \frac{1}{n}\sum_{i=1}^{n} X(i) \qquad \textbf{(2.2)}$$

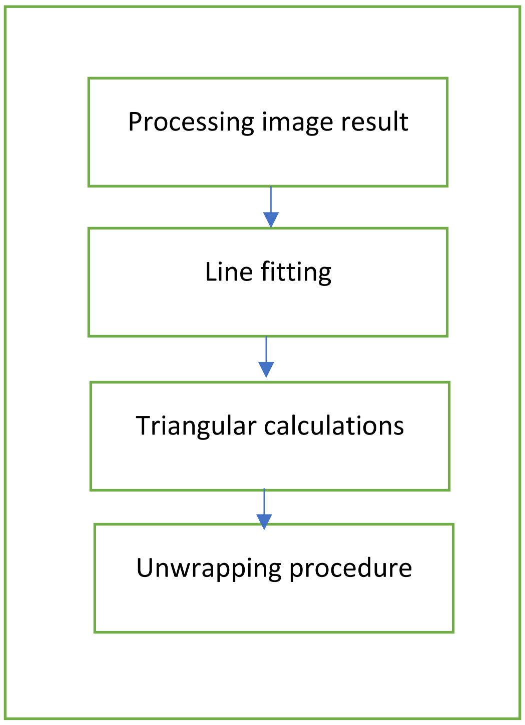


$$\frac{1}{16}\times\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline\end{array}$$

Figure 2.5: 3x3 mean filter example.

2.4 Height Calculations:

Height calculations in this system are calculated by estimating the depth information using the saw-tooth fringe and triangular equations. This method was first proposed and simulated using MATLAB by Hamdan, et el [44] and was practically implemented by Al-Hiari [1]. The steps of height calculations are shown in Figure 2.6.



.

Figure 2.6: Height calculation steps.

2.4.1 Line fitting:

One of the problems caused by the camera is that it tends to average the intensities at the edges, i.e. at fringe period boundaries. The pixels around maximum and minimum are not the exact ones projected by the projector but are changed. These pixels should be corrected otherwise, the calculations will be wrong. To solve this issue, we use line fitting technique. Line fitting is the line of the best fit. It is a linear line which may pass through none of the points, some of the points or all of the points. It represents the best approximation of a data set. Least square method [46] is the most accurate way of finding the line of the best fit of the points. The line fitting equation for a set of points is defined as y= mx+b, m and b are calculated using equation 2. 3 and equation 2. 4.

$$m = \frac{\sum_{i=1}^{n}(x-mean(x))(y-mean(y))}{\sum_{i=1}^{n}(x-mean(x))^2} \tag{2.3}$$

$$b = mean(y) - m * mean(x) \tag{2.4}$$

As we see in Figure 2.7, the change in intensity between two fringes is not sharp because of the intermediate pixels caused by contrast of the intensity and the shadow problem. We use line fitting to eliminate these pixels so the change from maximum of the period and the minimum of the next period is changing from black to white sharply. The dashed lines in the Figure shows the new line, minimum and maximum after applying line fitting algorithm.

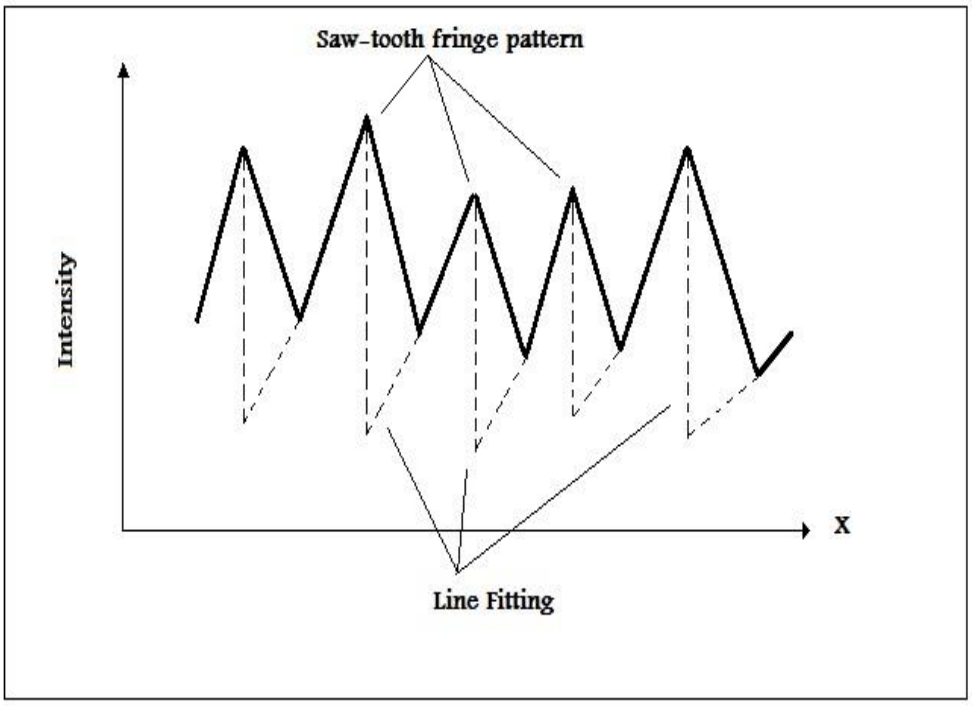


Figure 2.7: Saw-tooth line fitting example.

2.4.2 Triangular calculations:

Each fringe period of the captured images has a light intensity slope A and a period P between the maximum and the minimum as shown in Figure 2.8.

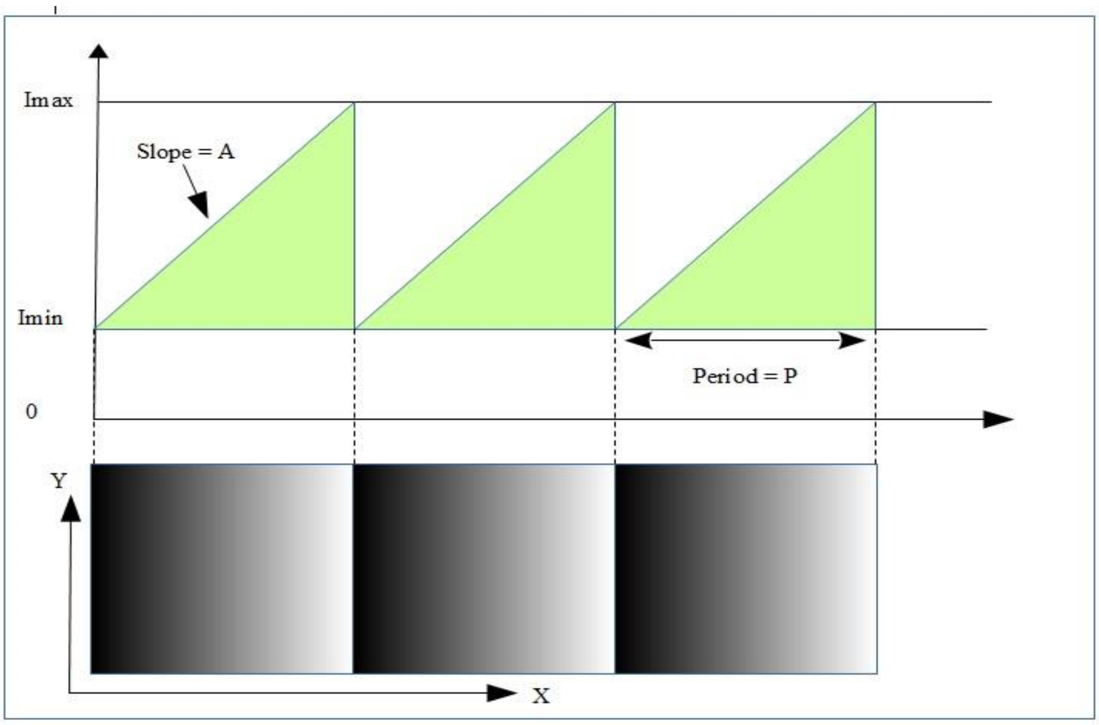


Figure 2.8: Projecting saw-tooth fringe pattern on the object surface.

As we see in Figure 2.8 the intensity is changing along the X-axis only depending on equation 2.5:

$$I(xi, yj) = A(xi - KP) + Imin \qquad (2.5)$$

where (xi, yj) is a point in the reflected image. K is an integer value represents the period number. P is the period of the saw-tooth function. A is the slope and calculated using equation 2.6:

$$A = \frac{Imax - Imin}{P} \qquad (2.6)$$

Assume now we put an object on the flat surface such that the height difference between the point X1 and the point X2 along the X-axis is equal to delta of height (ΔH) as shown in Figure 2.9 [1,45]. The intensity at point X2 is the same as the intensity of X3 if the object does not exist, i.e. If the light ray continues its path until it reaches the flat surface at X3. So, the light intensity at X2 can be written as follows [45].

$$I(X2) = I(X1) + A * (X3 - X1) \qquad (2.7)$$

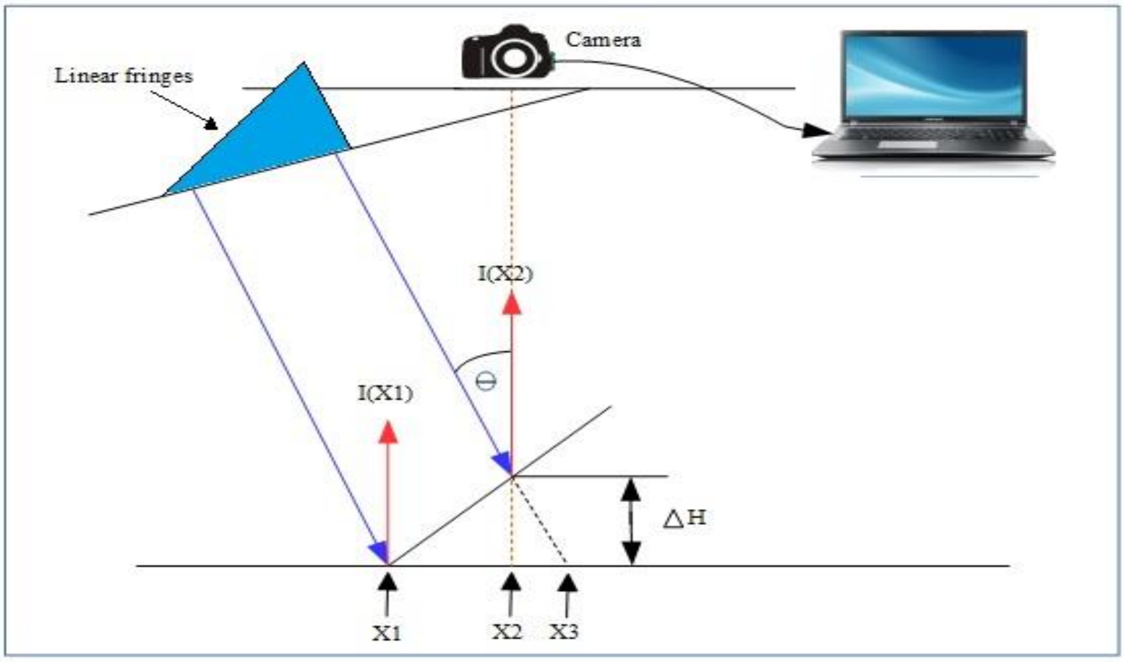


Figure 2.9: Representation of ΔH.

From the previous Figure, we see that:

$$tan(\theta) = \frac{X3 - X2}{\Delta H} \qquad (2.8)$$

which can be written as:

$$X3 = X2 + tan(\theta) * \Delta H \qquad (2.9)$$

From equation 2.7 and equation 2.9 we get:

$$I(X2) = I(X1) + A * [X2 + tan(\theta) * \Delta H - X1] \qquad (2.10)$$

Let $\Delta X = X2 - X1$, $\Delta I = I(X2) - I(X1)$

$$\Delta I = A * [\Delta X + \tan(\theta) * \Delta H] \quad (2.11)$$

Let $\beta = 1/\tan(\theta)$, $\alpha = 1/A$

$$\Delta H = \beta(\alpha \Delta I - \Delta X) \quad (2.12)$$

At each pixel ΔH is calculated by computing the intensity difference of the pixel and the previous pixel so ΔX=1.

2.4.3 Unwrapping procedure:

The unwrapping procedure is a very important step. If we look at Figure 2.10 we will see that X1 and X2 are laying on two different periods.

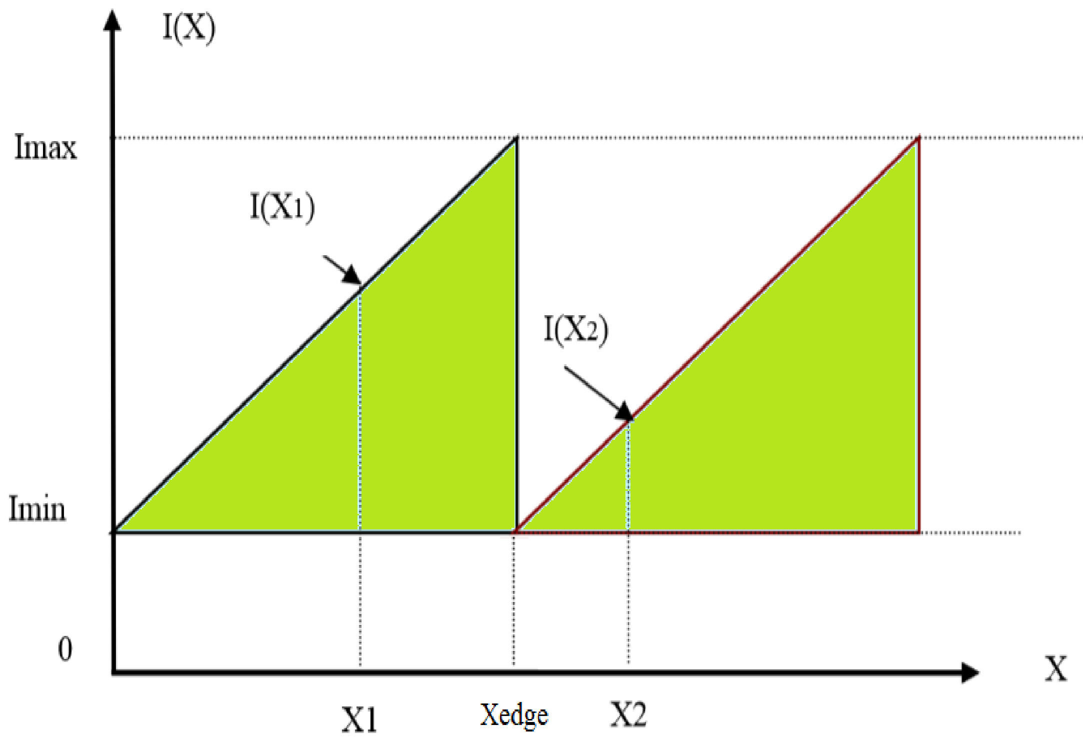


Figure 2.10: Two pints in two different periods.

In this case, we need to make a correction on the measurement of the light intensity difference of the two points. The height difference between X2 and X1 is the sum of the two height differences ΔH1= between X2 and Xedge plus ΔH2 between Xedge and X1.

$$\Delta H = \Delta H1 + \Delta H2 \qquad (2.13)$$

$$\Delta H1 = \beta\,(\alpha\,\Delta I1 - (Xedge - X1)) \qquad (2.14)$$

$$\Delta H2 = \beta\,(\alpha\,\Delta I2 - (X2 - Xedge)) \qquad (2.15)$$

$$\Delta I1 = Imax - I(X1) \qquad (2.16)$$

$$\Delta I2 = I(X2) - Imin \qquad (2.17)$$

From the previous equations:

$$\Delta H = \beta\alpha\,(\Delta I1 + \Delta I2) - \beta\alpha\,(Xedge - X1 + X2 - Xedge) \quad (2.18)$$

Let ΔI(old)= I(x2) – I(X1) and ΔI(new)= ΔI(old) + (Imax-Imin):

$$\Delta H = \beta\alpha\,(\,\Delta I(new) - (X2 - X1)\,) \qquad (2.18)$$

Unwrapping is the process of combining fringe periods such that it looks continuous and all discontinuities are removed, see Figure 2.11.

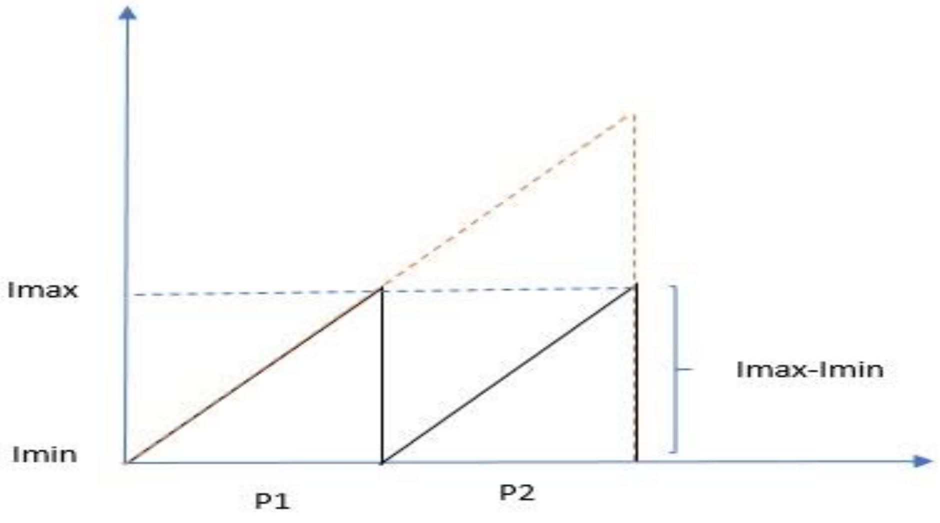


Figure 2.11: Unwrapping procedure.

2.5 3D profilometry:

The procedure of the 3D profilometry shape measurement is to go through the steps shown in Figure 2.12. Height calculation result already described in the previous section.

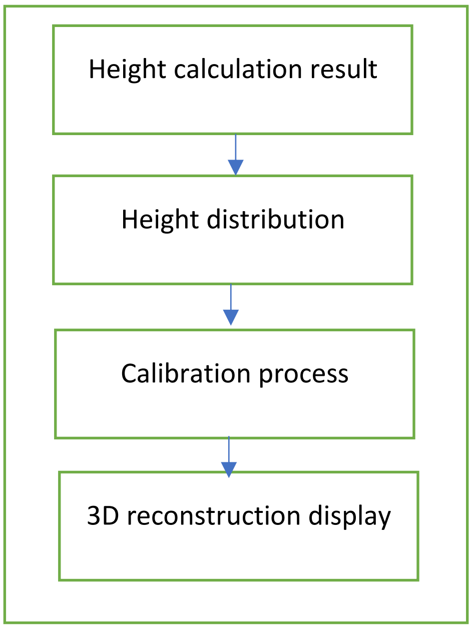


Figure 2.12: Main steps of the 3D profilometry.

2.5.1 Height distribution:

After calculating ΔH in the previous section and doing the unwrapping procedure, the next step is the height distribution. Height distribution is to extract the depth value of each pixel which is the z-axis value of the (xi, yj) pixel. When projecting a fringe pattern on a flat surface, the periods of all fringes will be constant if the projector was at right angle with the surface. However, when the projector is tilted by a certain angle as required by this technique, the fringe periods are no longer of the same period but changes in a linear way.

To compensate for this effect, we do two projection: one on the flat surface with no object, the other with the object placed on top of the flat surface. We compute the height in both cases, then we subtract the flat surface height from the flat with object, see equation 2.19.

$$\text{height distribution} = H(\text{of the object}) - H(\text{of the reference}) \quad (2.19)$$

2.5.2 Calibration process:

Due to the projector and camera propitiates, the measurement are not the same as the real values so we need to use a calibration method. The calibration process is a very important step of the 3D profilometry, it is used to calculate the real measurement of the object under test.

First, we start with the X-axis which represents the length of the object used. We calibrate the X-axis to the original length of the object by multiplying it with the scalar calculated in equation 2.20:

$$\text{X-axis calibration} = \frac{\text{length of the object}}{\text{size of image }(\text{x-axis})} \quad (2.20)$$

The next step is to calibrate Y-axis which represents the width of the object to the original object by multiplying it with the result of equation 2.21:

$$\text{Y-axis calibration} = \frac{\text{width of the object}}{\text{number of lines in the object}} \quad (2.21)$$

The final step is to calibrate the Z-axis which represents the height of the object to the original height of the object by multiplying it with the result of equation 2.23, but before that we have to calculate the average of ΔH using equation 2.22.

$$\text{average of } \Delta H = \frac{\sum \max(\text{unwrapping } \Delta H)}{\text{number of lines}} \quad (2.22)$$

$$\text{Z-axis calibration} = \frac{\text{height of object}}{\text{average of } \Delta H} \quad (2.23)$$

2.5.3 3D reconstruction display:

After finishing the previous steps and calculating the depth for each point, we will have the z coordinate for each (x,y) point. Using the (x,y,z) coordination's we build 3D modules for the object. A special software will be used for displaying the module using different displaying shapes.

2.6 Experimental results:

In our research, we used the setup shown in Figure 2.13. Two wooden boards are used in the setup, the first board is used to set the angel of projecting accurately by drawing lines representing the angles used. The second board is to project the fringe pattern on it, it is colored white which is the best for reflecting the light. We used two angles 10˚ and 15˚. The minimum and maximum intensity is 10 and 240 respectively.

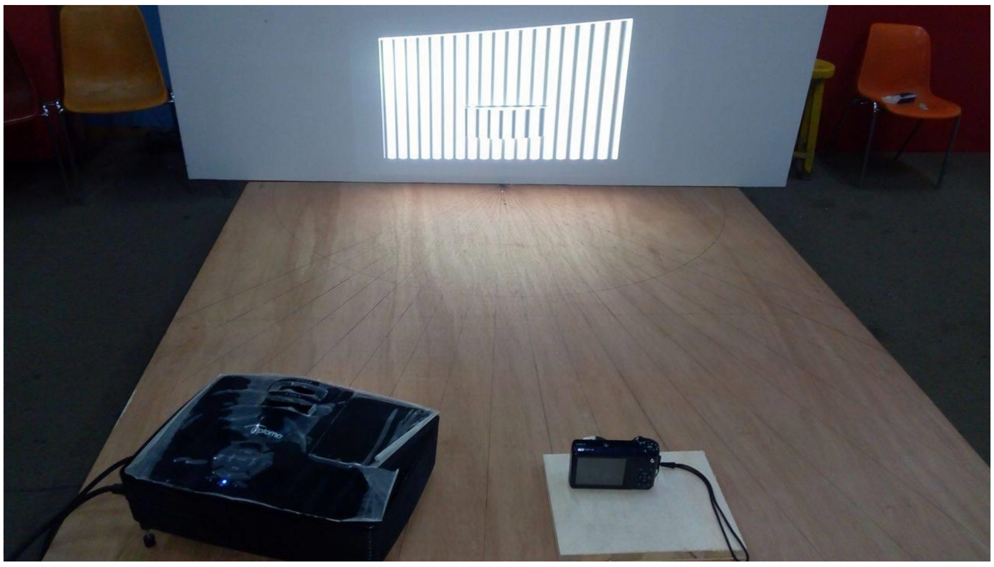


Figure 2.13: System setup

Multiple objects are used in this research. The example that we will discuss in this chapter is a triangle cross section object with length = 15cm and, width= 7cm and height= 2cm, see Figure 2.14 . The period is 16 and the angle is 15˚.

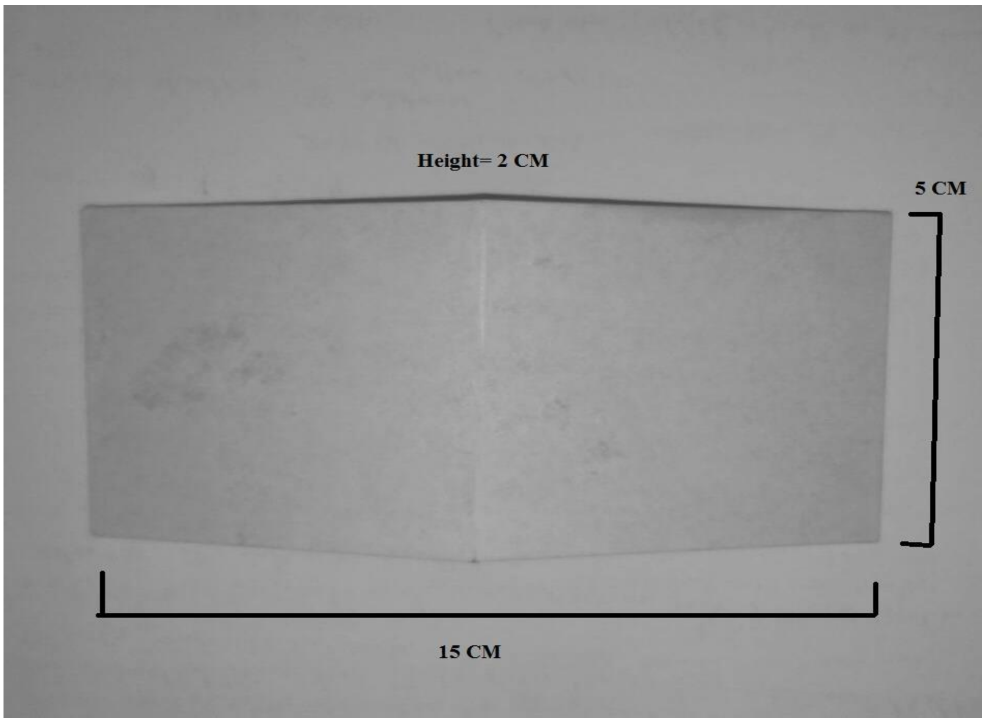


Figure 2.14: Triangle cross section object dimensions.

We take 4 images: First is for the reference plane projected by vertical lines located at the edges of the fringe pattern that will be used. The second image is the same pattern of vertical lines in first image projected with the object under test is placed on top of the flat surface. The third image is a saw-tooth pattern (with the same pattern period in the first two images) projected on the flat surface. The fourth image is as same as the third image but the saw-tooth pattern is projected with the object under the test is placed on the flat surface. The important factor is to keep the setup unchanged (angel of projection, camera location, etc.) during the images capturing.

After capturing the images for the object, reference and edges we store them in the computer the processing is started. First, we apply the method on one line of the image (a row of pixels) then we expand the process in all the other lines to complete the full image. In the next section, we will discuss this example step by step.

2.6.1 Example of 3D profilometry:

In this section, we will go through our algorithm step by step.

Reading and cropping the images: In this step, we read and crop 4 images, one for the reference plane, one for the object and two images for the fringe pattern edges with and without the object. Figure 2.15 and 2.16 shows the four images after reading and cropping.

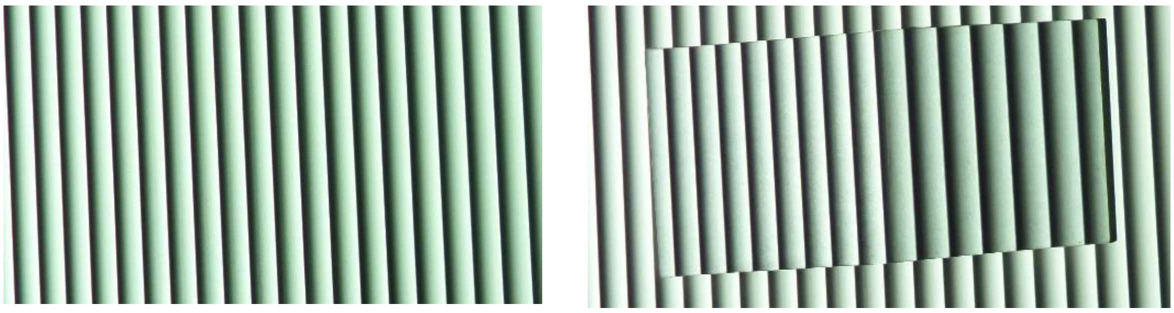


Figure 2.15: Two images for the reference plane and the object.

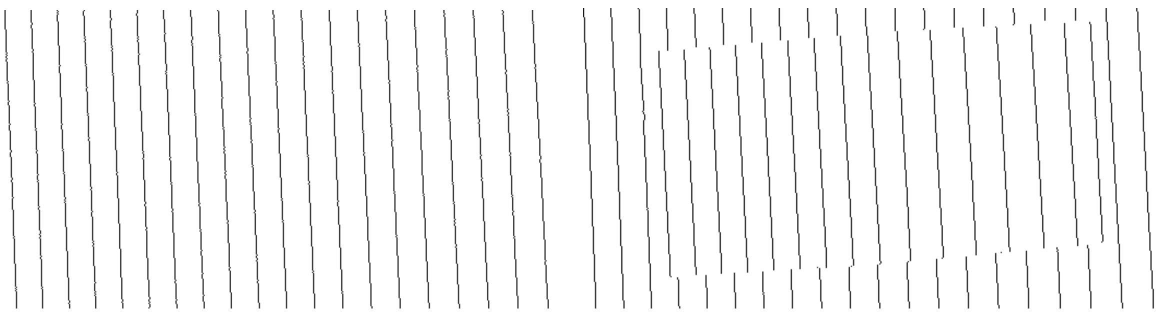


Figure 2.16: Tow images for the edges of the fringe patterns with and without the object.

Converting to gray: In this step, we convert the images from RGB color level into gray color level**.** Figure 2.17 shows the two images after applyi ng color to gray filter.

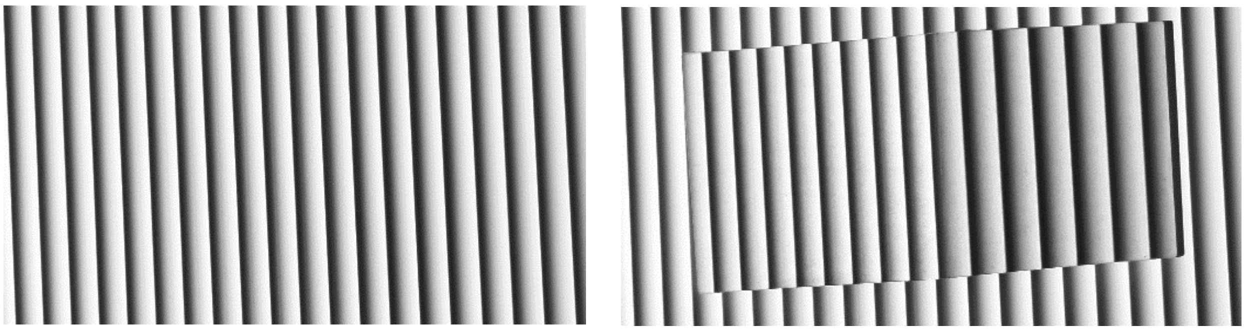


Figure 2.17: Converting the images to gray level.

Noise removal: In this step, we use image enhancement method like median and mean filters to the images to remove the noise caused by the projector or the camera. Figure 2.18 shows the reference and object images after applying the filters.

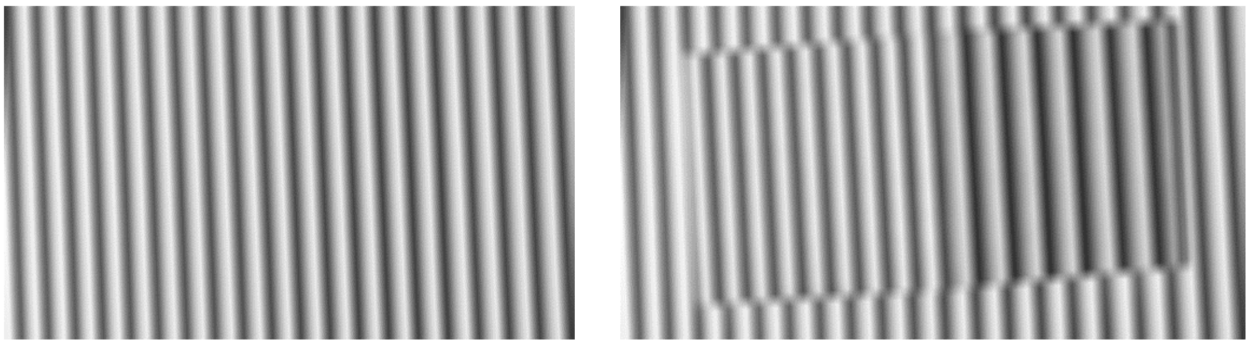


Figure 2.18: Images after the filters.

One line profilometry and average line filter: To demonstrate the algorithm for 1D case a single line profilometry, we choose the middle line of both images as an example, see Figure 2.19. Average filter is used to give a better result and to reduce noise.

Figure 2.20 shows the line before and after applying the average filter. We found that eleven-degree average filter gives the best results. The average filter doesn't affect the edges because we already captured images for the edges, so we know exactly where the edges are located before applying the average filter.

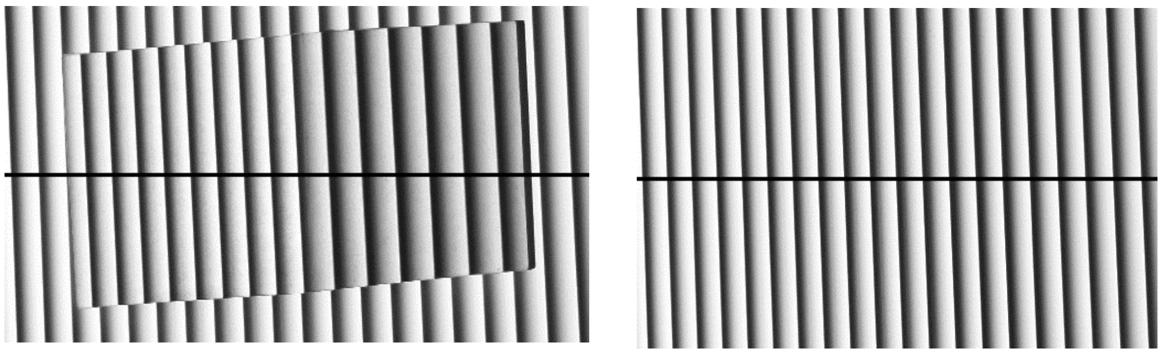


Figure 2.19: The middle line of both images to apply the methods.

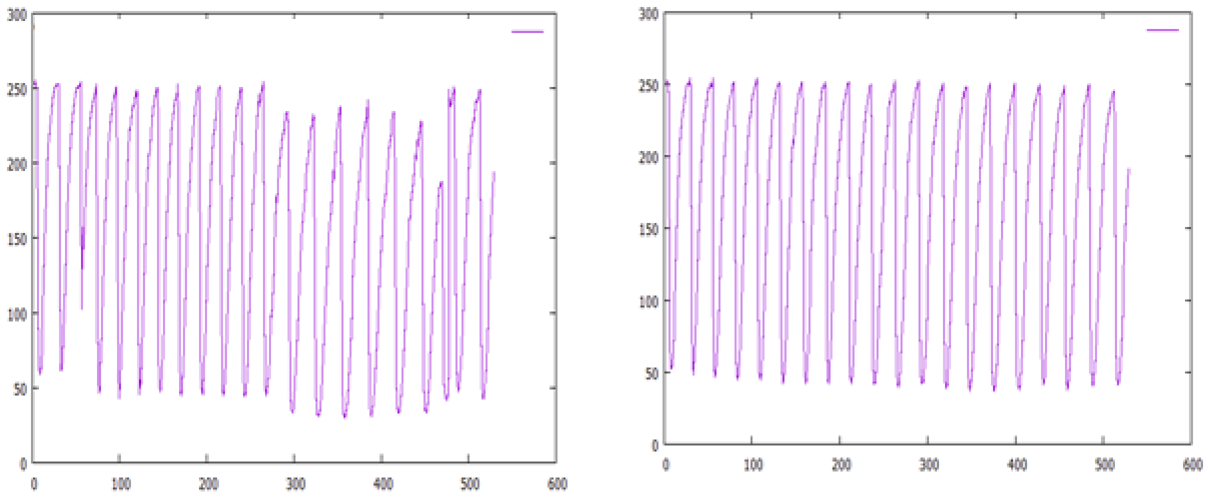


Figure 2.20: The intensity of the line before and after the average filter.

Line fitting: In this step, we apply the line fitting method. Figure 2.21 shows the intensity before and after line fitting.

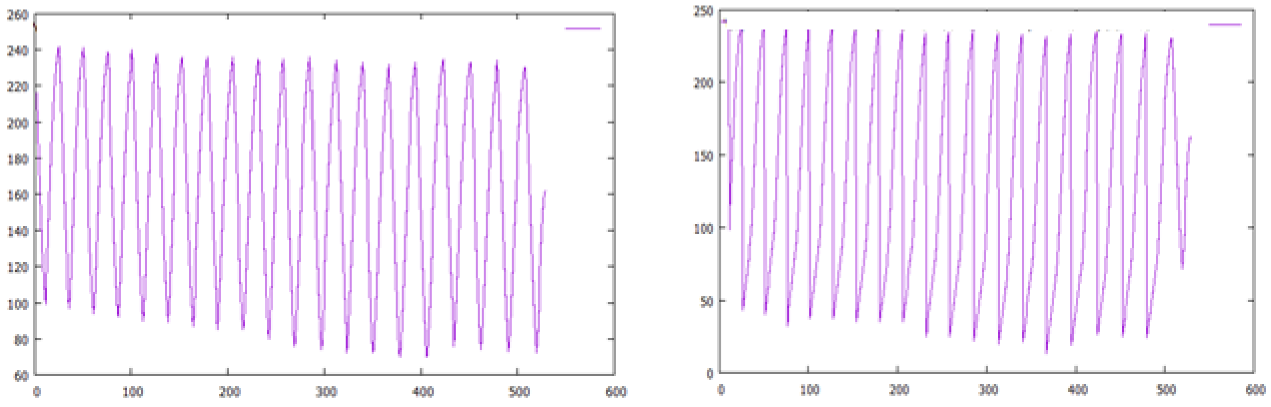


Figure 2.21: The line intensity before and after line fitting.

Height calculations and unwrapping procedure: After finishing the previous steps, we calculate the heights for every point. We apply the unwrapping method (Unwrapping procedure already explained in this chapter) with average filter to the heights and get the results.

Height distribution: After calculating the heights of the object and the reference, we find ΔH for each pixel. Figure 2.22 shows the height distribution for the object under the test.

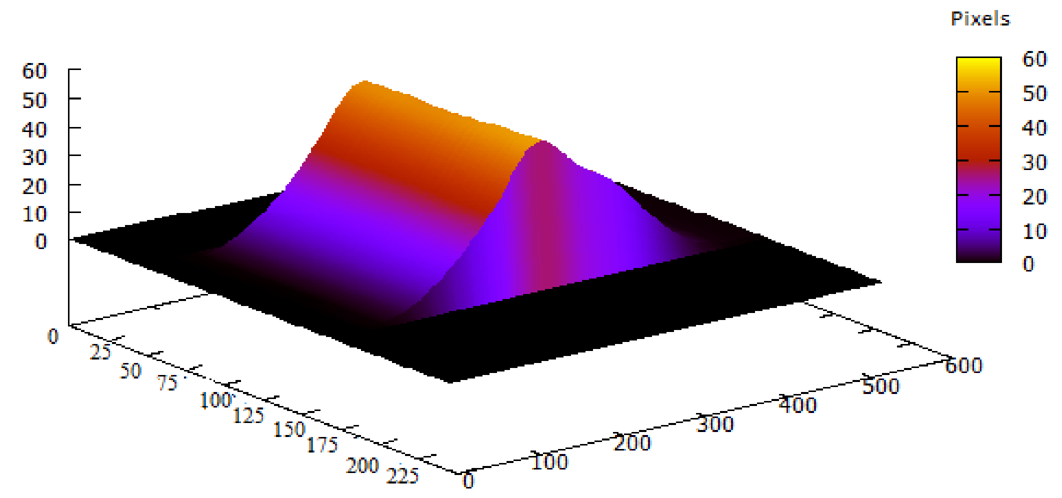


Figure 2.22: The 3D profilometry of the object.

Calibration process: In the calibration step, we use the calibration methods to calibrate the object heights. We use the scaling to change the result from pixels to centimeter. The 3D profilometry of the object after calibration is shown in Figure 2.23.

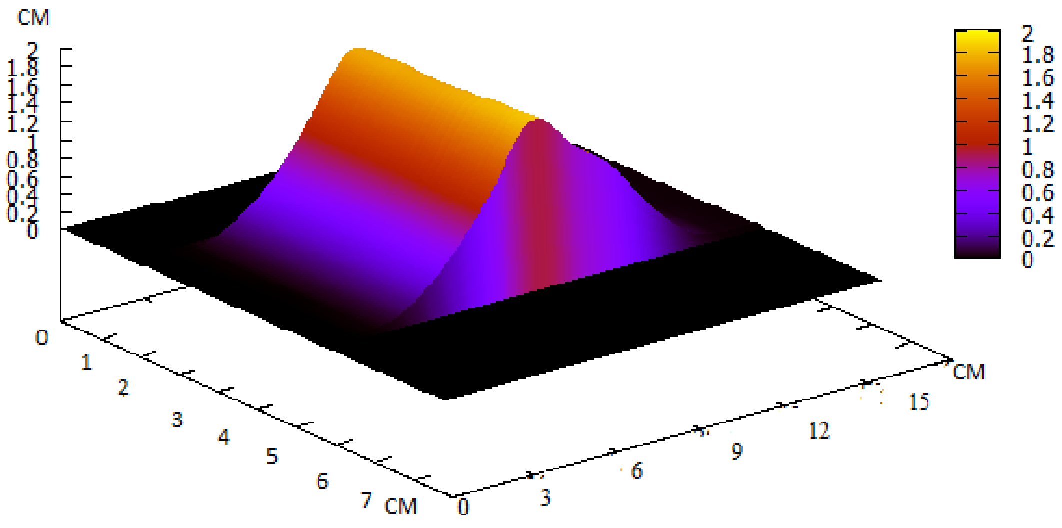


Figure 2.23: The 3D profilometry of the object after calibration.

3D reconstruction display: The 3D module of our object under the test is now ready to display. We use different shapes to display the model, see Figures 2.24.

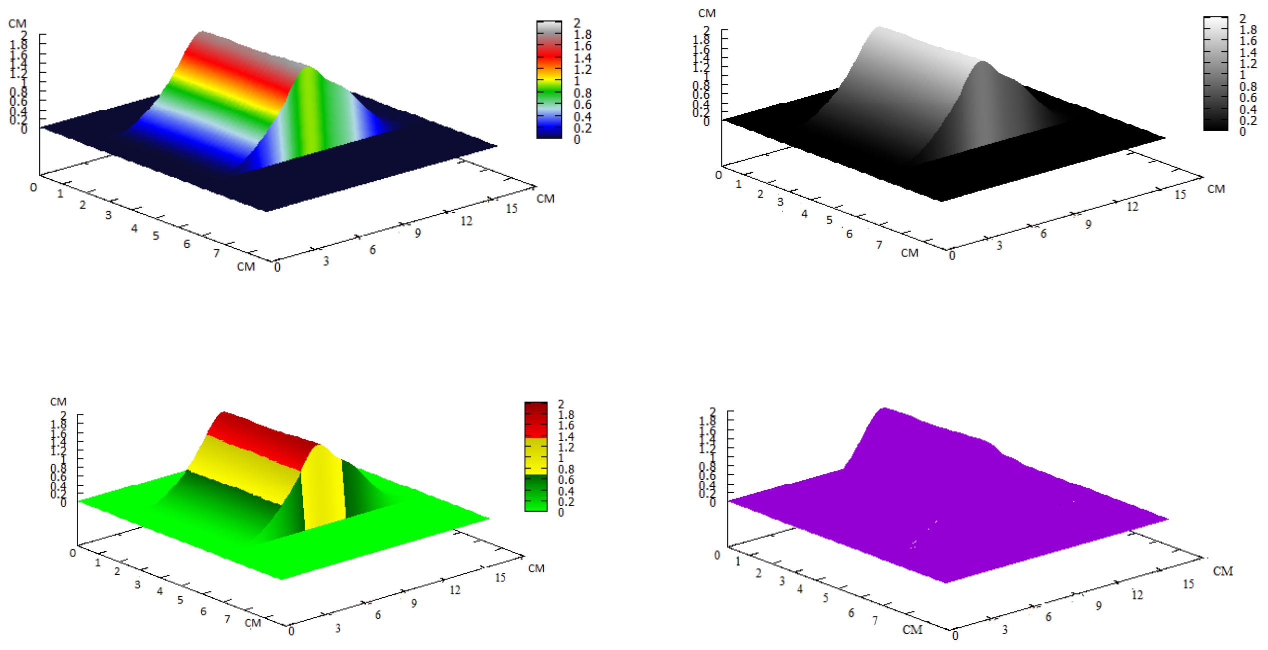
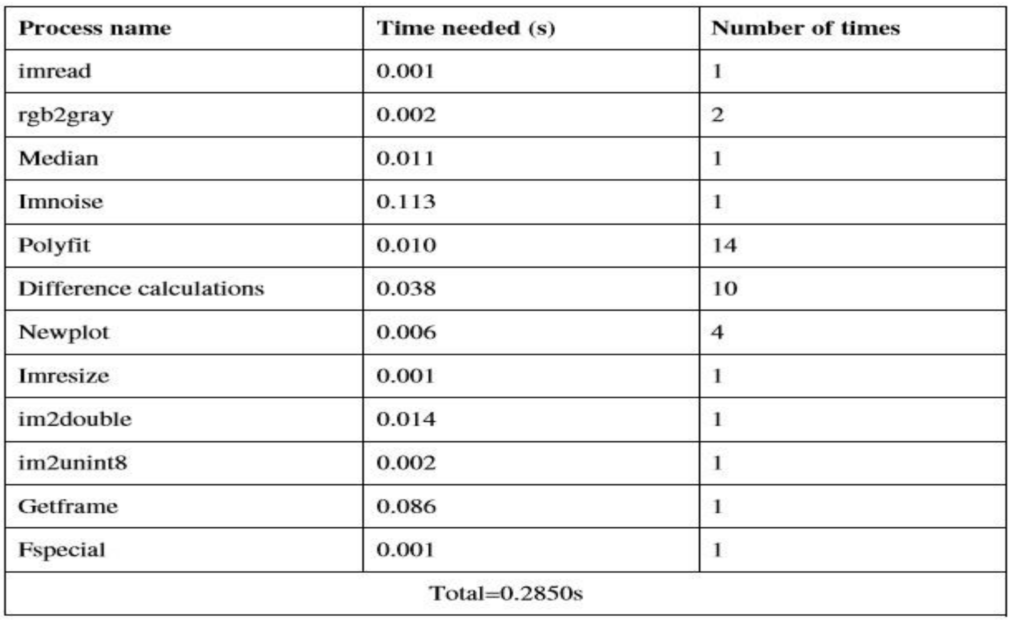


Figure 2.24: 3D displaying of the object

Chapter 3

Speed Enhancement

3D profilometry using Saw-tooth fringe pattern proposed by Al-Hiari [1] using MATLAB code. MATLAB code is slow comparing to c++ language codes. In this research, we will enhance the speed of the algorithm by writing it using c++ language and using the power of parallel processing. Table 3.1 shows the run time results using MATLAB for one dimensional array of the image [1].



Process name	Time needed (s)	Number of times
imread	0.001	1
rgb2gray	0.002	2
Median	0.011	1
Imnoise	0.113	1
Polyfit	0.010	14
Difference calculations	0.038	10
Newplot	0.006	4
Imresize	0.001	1
im2double	0.014	1
im2unint8	0.002	1
Getframe	0.086	1
Fspecial	0.001	1
Total=0.2850s		

Table 3.1: Speed results for one dimensional 3D profilometry using MATLAB [1].

3.1 Saw-tooth 3D profilometry using c++ language:

As we mentioned in the previous section, c++ language is faster than MATLAB. The reason why MATLAB is slower because it is an interpreted language so it's usually slower than compiled languages. Later in this research we will discuss the parallel code we used for enhancing the speed, but we will first start with discussing the sequential code.

Software and tools used for sequential code:

Microsoft Visual Studio 2015: it is an integrated development environment (IDE) developed by Microsoft. It is an open source development application helps the programmers to develop applications or building website and many other uses [47]. It has code editor and supporting many languages such as c, c++, c#, python, etc.

openCV: it is an open source library of programming functions developed by Intel. The main aim of OpenCV (Open Source Computer Vision) is the real-time computer vision. OpenCV has C++, C, MATLAB, Python and Java interfaces and supports Windows, Mac os, Linux and Android [48]. In our research openCV will help us with image processing like reading the images or image enhancement.

The steps for writing the code already discussed in the previous chapter with an example. First, we read the images of the reference, edges and object. Then we convert all the images from color to gray level. After that we apply average and median filters to remove the noise and get better intensity values.

After that we apply line fitting method. The next step is to apply the triangular calculations to calculate ΔH of the reference and the object. Then we calculate the heights of the object by subtracting the values of ΔH of the reference from the values of ΔH of the object. The final step is to display the reconstructed shape of the object.

All these steps are done using c++ language and OpenCV libraries. The speed results of the sequential code and the comparison with MATLAB will be discussed in details in chapter 4.

3.2 Parallelization

3.2.1 Parallelization overview

Parallel processing is to run programs using multi-processor computers which help to speed up the running time. Recently, most of computers comes with more than one processor. This changed the way that programs should be written. Most of programs are now developed in a way that the program can run on multi-processors at the same time which is called parallel processing.

In this research, we will use the power of parallel processing to speed up our algorithm run time. In the next section, we will talk about parallel processing techniques and how we will use them in our research. Also, we will discuss an example of the proposed algorithm but using parallelization.

In the next chapter, we will discuss the results of the speed and we will make some comparisons between sequential and parallel codes, also between different number of cores.

There are different techniques of parallel processing. Data parallelism and task parallelism are two of the most common techniques.

Data parallelism: In n data parallelism, the method focus on distributing the data in a way that each processor will do the same task but for a different piece of data.

Task parallelism: In task parallelism, the method focus on distributing the tasks to run on a different processor for the same or different data. In this method, the tasks should be independent from each other or we will get wrong results.

In this research, we will use data parallelism, it is very compatible with image processing because it is easy to split the images into small images and process each image individually without affecting the results. Task parallelism is not compatible with our work because some tasks depends on other tasks and need to wait until these tasks are completed to start running it without affecting the results. As an example if we start running the task of calculating the heights of an object before converting the images from color to gray level then we will get wrong results.

3.2.2 Parallelization procedure

As we mentioned before we will use data parallelism in our research. The way we use data parallelism in our research is by splitting images into smaller images and run each piece of the original image on a single processor. There are different ways to split the images as shown in Figure 3.1.

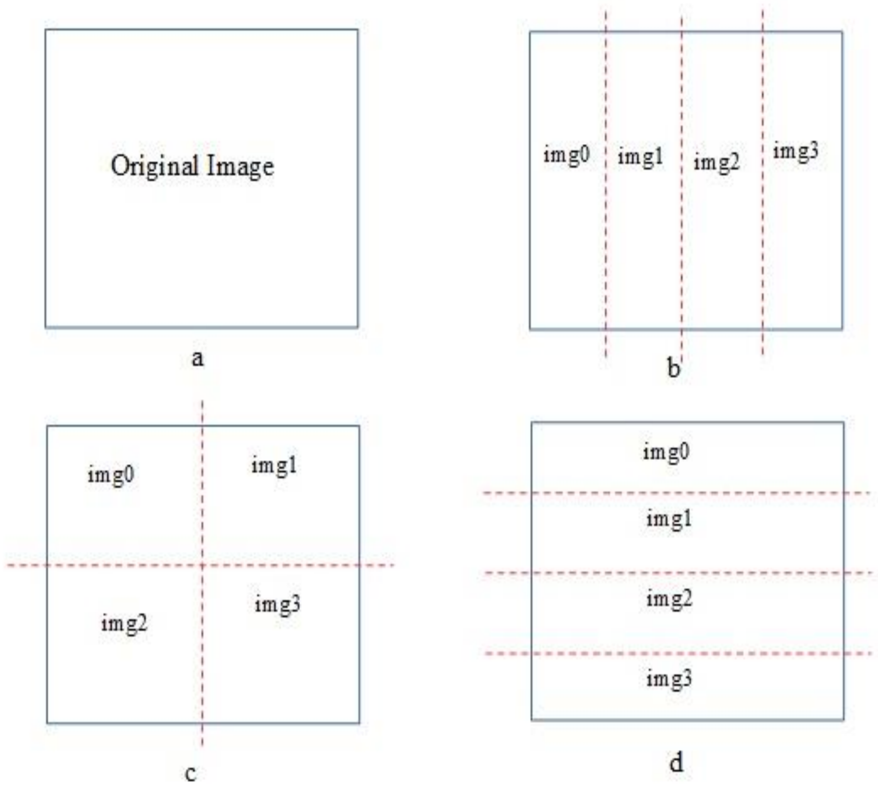


Figure 3.1: Image splitting methods.

The procedure of our work run through each row of image separately (except the average filter), so we have no issues if we cut the image horizontally. In case we cut the image vertically each row will split and this will make issues in the pixels on the boundaries of cutting as shown in Figure 3.2.

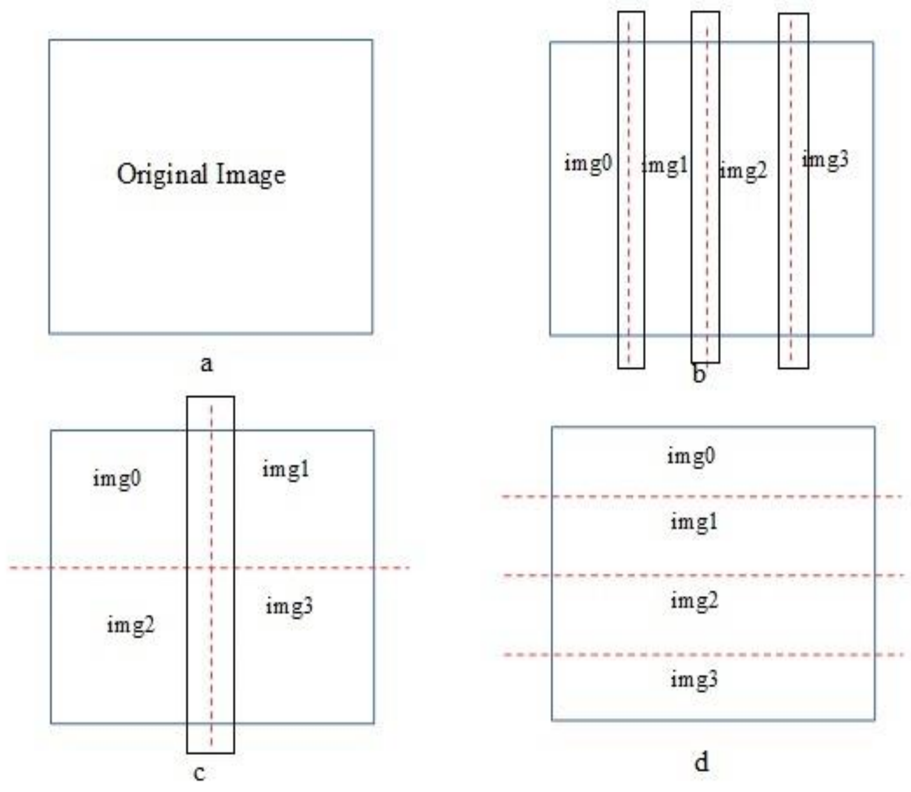


Figure 3.2: Vertical edges issues.

The issues shown in the previous Figure need more running time to fix them. Rows of image appear in contiguous memory addresses, see Figure 3.3. By splitting the image into horizontal images (Figure 3.1 d) and assigning each part to one processor, we make it faster for each processor to access its own data since they are all located in a contiguous block of memory.

The number of slices of the original images depends on the available number of processors, as example if we have two processors the program will slice the original image into two images and so on. Figure 3.4 shows how we slice the image depending on processors number.

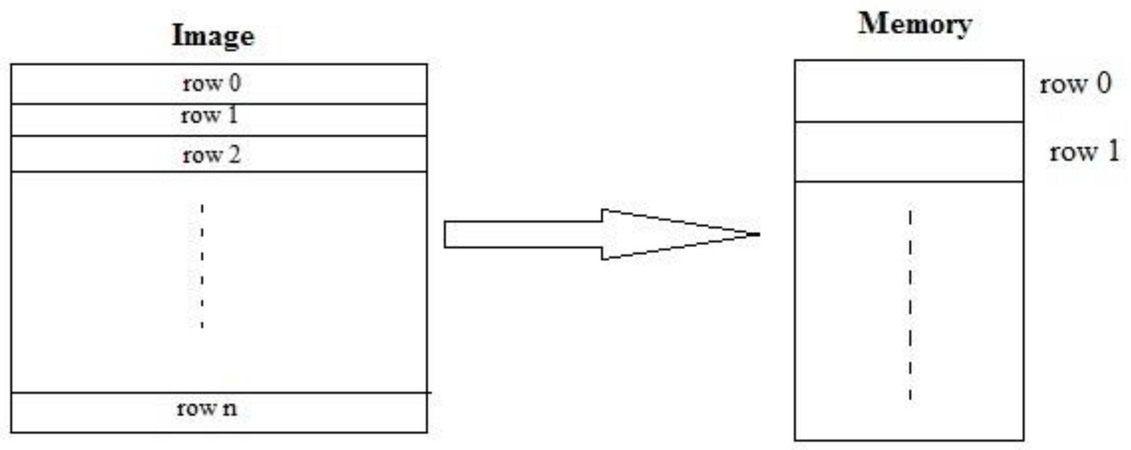


Figure 3.3: How image rows saved on memory.

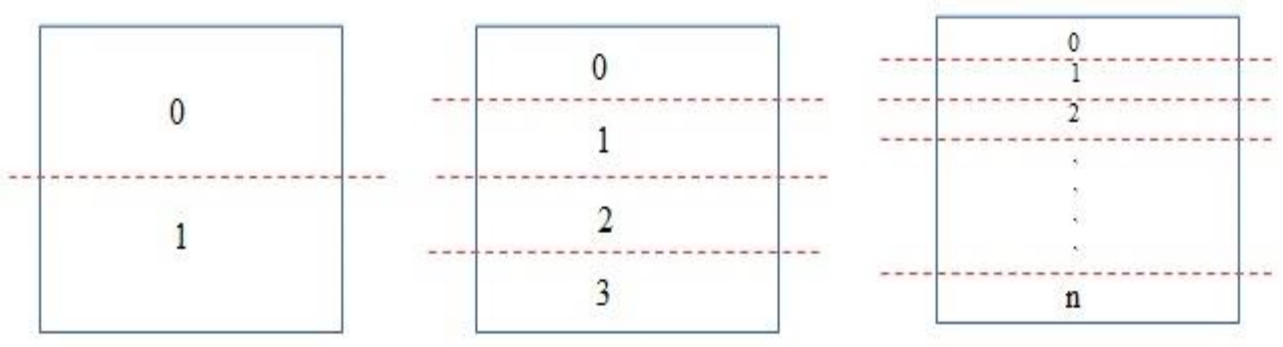


Figure 3.4: Slicing images depending on the processors/cores number.

After slicing the images, we distribute each slice to a single core/thread and run the tasks at the same time. We do all the sequential steps described in the previous chapter for every slice as same as the original image.

If we use a computer with (n) number of threads/cores, then we slice all the images into n number of small images. All the small images of the edges, reference and object will have the same size and the same range of pixels. Each group of small images of the same range will go through all the steps together, see Figure 3.5.

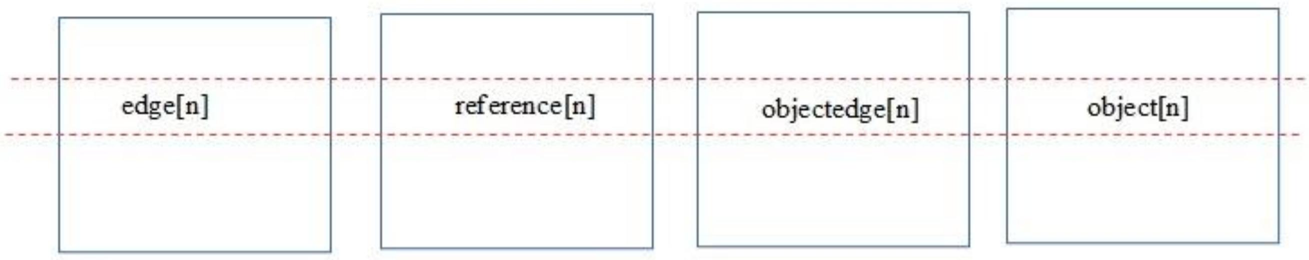


Figure 3.5: A group of small images of the same slice number.

At the end of the sequential steps for each group of slices we will have n number of heights arrays. Each array represents the heights of the object but for the range of pixels the group represents. The final step is to combine these arrays into one array that represents the whole object. See Figure 3.6.

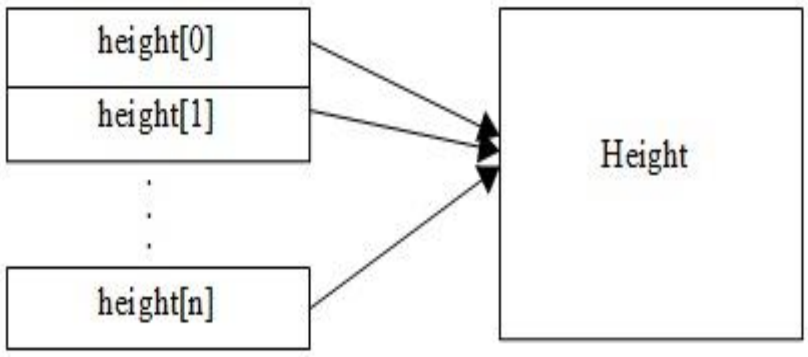


Figure 3.6: Heights combining.

3.2.3 Experimental example

In this section, we will discuss one test case of our work. In this example, the object is a triangle and the processor is Intel core i3-2350M which has 2 cores and 4 threads.

The first step is to read the images, we use 4 images in our system. Object image which is captured after projecting the saw-tooth fringe pattern on the object. The second image is captured after projecting the edges pattern on the object. The last two images are the same of the first and second images but without the object, we project the edges and saw-tooth pattern on the reference plane only. Figure 3.7 shows the 4 images after the reading and cropping process.

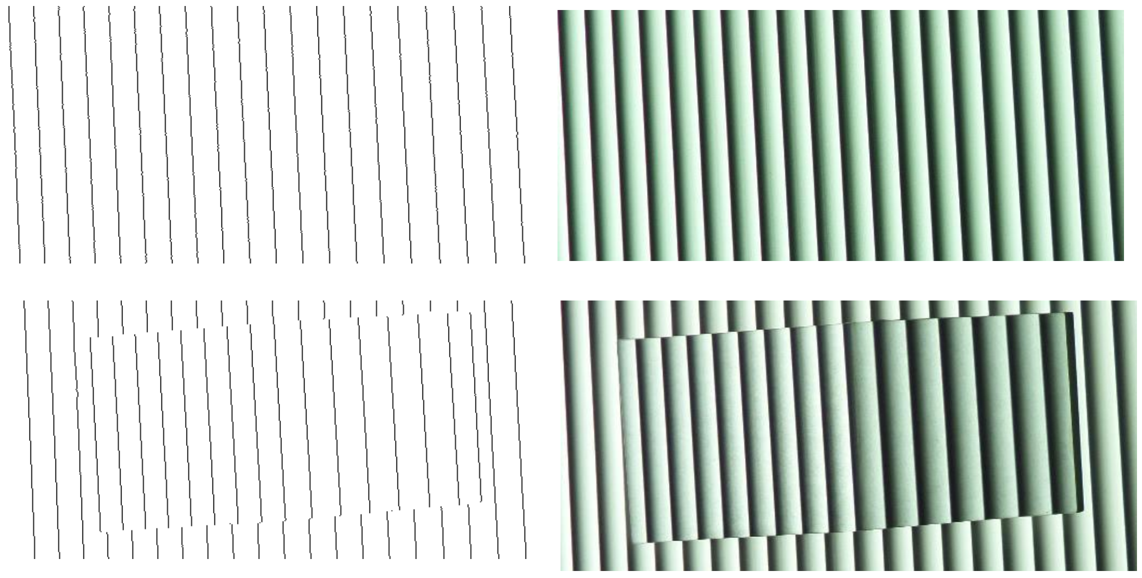


Figure 3.7: The 4 images after reading and cropping in parallel.

The next step is slice the images. Because the number of threads is 4, we first start by slicing all images into 4 slices for each image as shown in Figure 3.8. We use the command Rect to create a rectangular area which will store the slice of the image in it. Then we use push_back function to store the data of the rectangular area of the image.

```
Rect rect = Rect(x, y, smallSize.width, smallSize.height);
smallImages.push_back(Mat(bigImage, rect));
```

The width of the slice will be the same as the width of the original image. The height of the slice will be the result of dividing the original image height on the number of the available cores (threads).

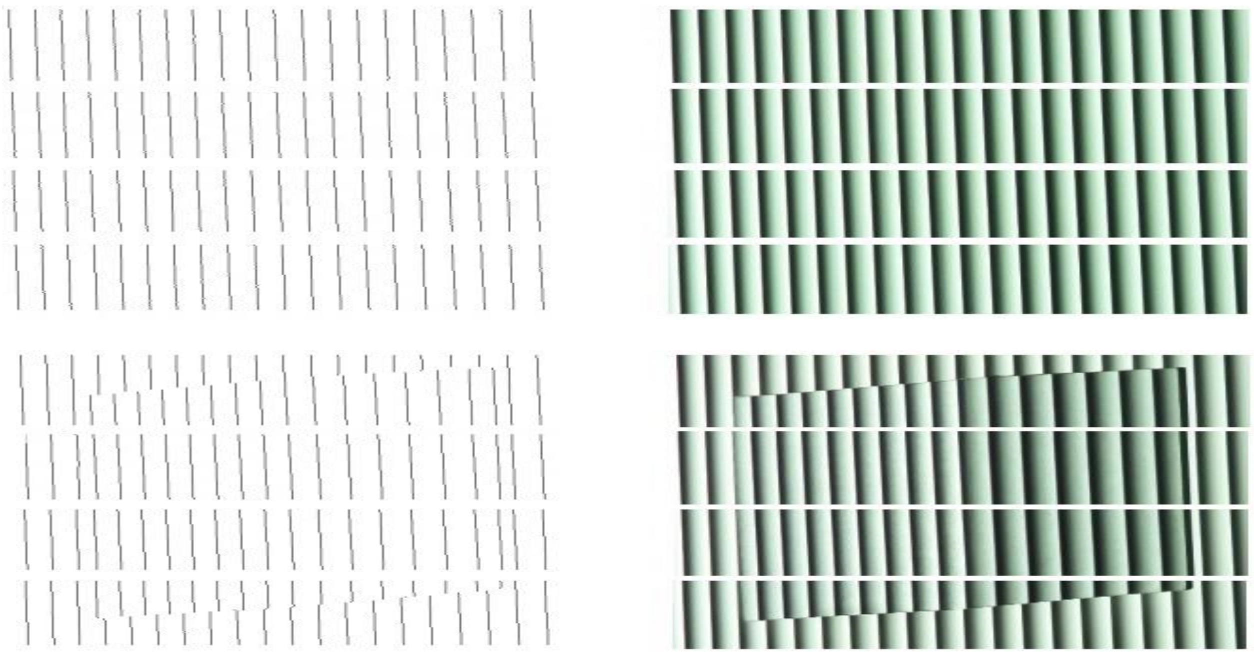


Figure 3.8: Images after slicing.

In order to divide the work on the available cores we use the Cilk Plus keyword **cilk_spawn** before the function we want to parallelize. This keyword tells the processor that you can process all functions that we wrote **cilk_spawn** before it in parallel. Some functions need other functions to end before they started or they will give wrong results, so the keyword **cilk_sync** tells the processor that you should finish all the parallel work before this keyword, then start running the other tasks.

The next step is to convert all the slices of the reference and object images from RGB color to gray level color as shown in Figure 3.9.

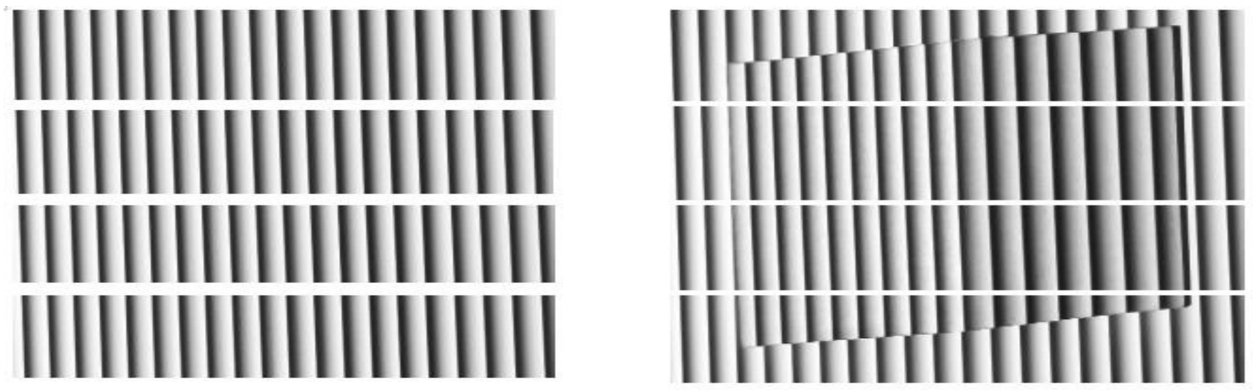


Figure 3.9: Image slices after converting to gray.

After that we apply image filter methods to the image slices of both the reference and the object, see Figure 3.10.

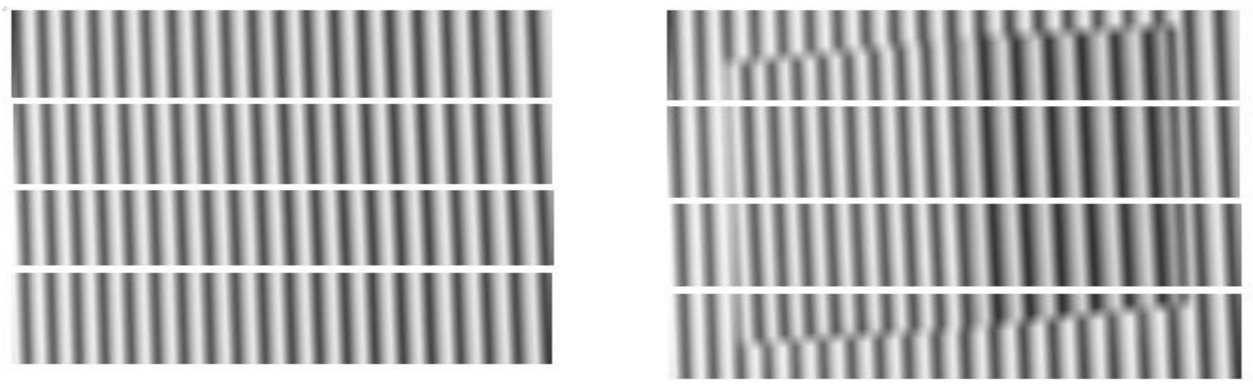


Figure 3.10: Images slices after applying the filters.

After finishing the previous steps and applying the height calculations we will get four arrays of heights. Each array represents a part of the object as shown in Figure 3.11.

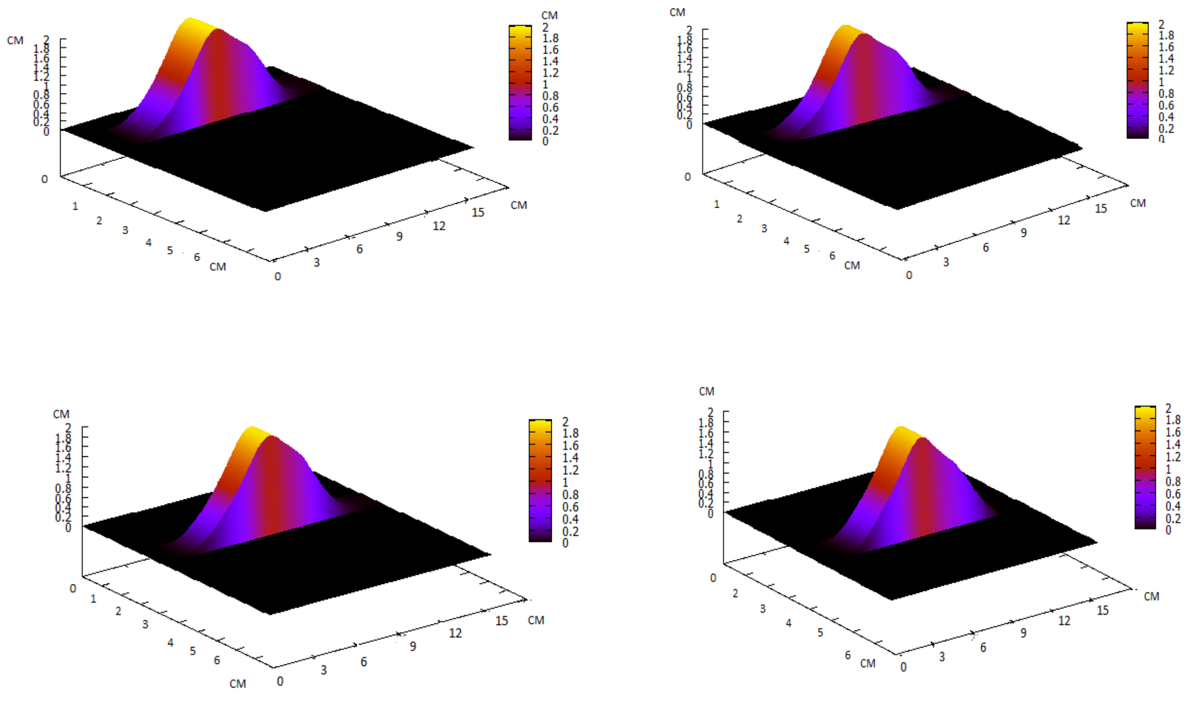


Figure 3.11: Height arrays 3D visualization.

The final step is to combine these height arrays into one array of all the object. Figure 3.12 shows the object module after combining all the height arrays.

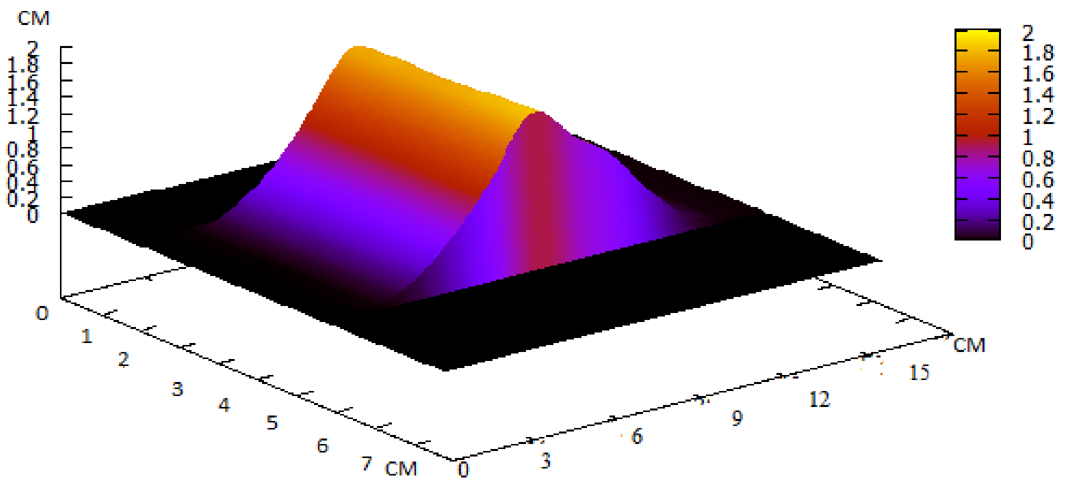


Figure 3.12: The 3D profilometry of the object after combining all the height arrays.

CHAPTER 4

Experimental Results

In this chapter, we will discuss the results of our research. First, we start with the 3D profilometry results. In this section, we will talk about the system setup and requirement, also we will show the specification of the equipment used and we will show the objects used and the 3D modules results.

The next section is about the serial c++ code speed results and comparison between our method and other methods. Finally, we will discuss the parallel code and compare the results with the serial code.

4.1 3D profilometry results:

In chapter 3 we discussed the saw-tooth fringe pattern and how the algorithm works. We also show an example of our work. In this section, we will specify the system requirements and show the objects used and the results.

4.1.1 System requirements:

In our system, we used a projector to project the fringe patterns on the object. The used projector is Optoma HD141X, the specifications of the projector are shown in table 4.1. The camera used to capture the images is Samsung wb800f. The specifications of the camera are shown in table 4.2. The computer used is Dell Inspiron 14z (N411z), the specifications of the computer are shown in table 4.3.

Optoma HD141X	
Display Technology	DLP
Native Resolution	1080p 1920 x 1080
Brightness	3000 ANSI Lumens
Contrast	23,000:1
Throw Ratio	1.48 - 1.62:1
Zoom Type	1.1x Manual
Projection Lens	F/2.5~2.67; f=21.9~24mm
Projection Screen Size	41.8" - 300"Diagonal 16:9
Displayable Colors	1073.4 Million

Table 4.1: Projector specifications.

Samsung wb800f	
Sensor Type	1/2.3″ (Approx. 7.76mm) BSI CMOS
Effective Pixel	Approx. 16.3 Mega pixels
Lens focal Length	Samsung LENS 21x Zoom Lens f = 4.1 ~ 86.1mm (35mm film equivalent: 23 ~ 483mm)
Optical Zoom	21x Optical Zoom
Digital Zoom	4X digital zoom

Table 4.2: Camera specifications.

Dell Inspiron 14z (N411z)	
RAM	4.00 GB
Processor	Intel® core™ i3-2350M CPU @ 2.30GHz
System type	64-bit system
Storage	500 GB

Table 4.3: Computer specifications.

The operating system is Windows 10. Microsoft Visual Studio 2015 is used for writing the code with openCV 3.1 libraries for images reading and processing. Gnuplot 5.0 used for 3D visualization.

4.1.2 3D visualization:

In chapter 3 we showed an example of our algorithm. The object of the example was a triangular object. The objects used in the experimental results are shown in Figure 4.1. Table 4.4 shows different 3D models of different objects after applying the algorithm on them.

The modules shown in the table visualized using different shapes by Gnuplot software. The objects shown are triangle, curved paper and car roof. The first row of a table is a 3D model of simulated car roof object. The other rows of the table show models of a real objects.

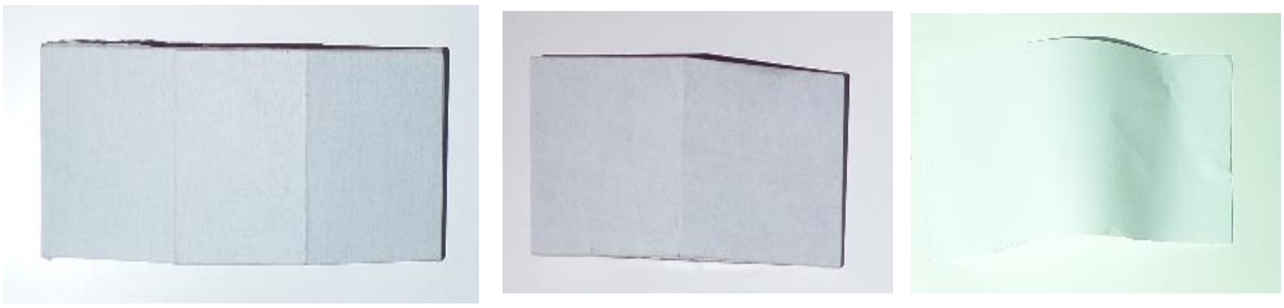


Figure 4.1: Objects used in the research.

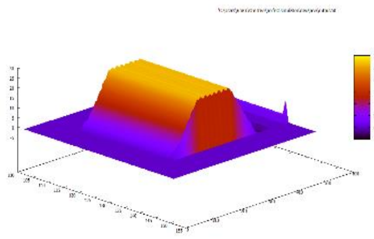
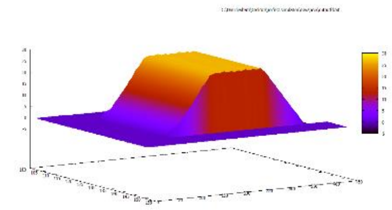
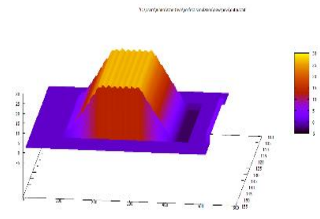
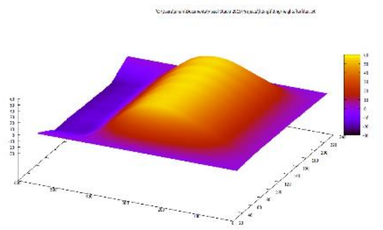
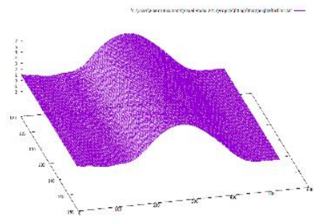
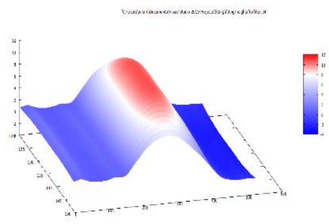
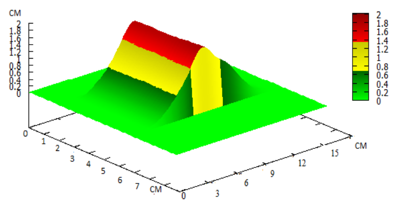
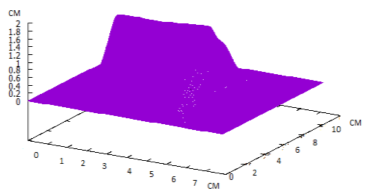
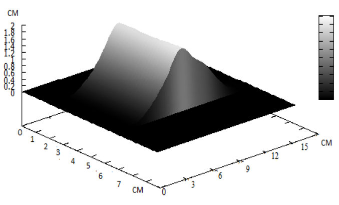
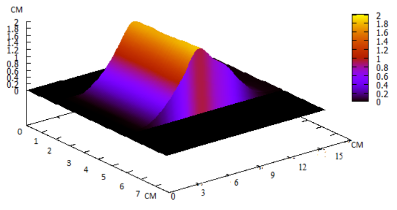
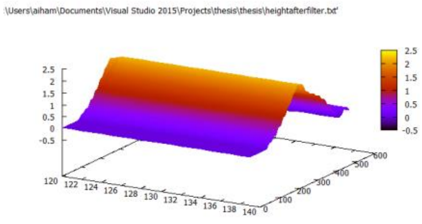
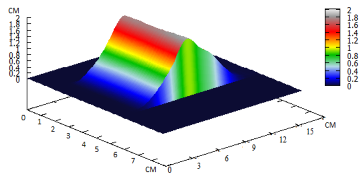
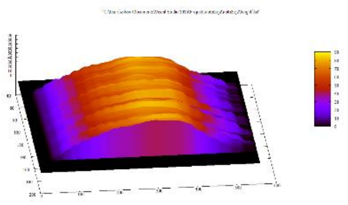
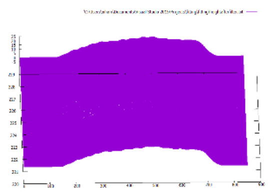
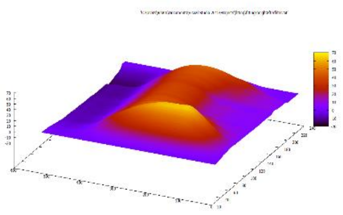
		
		
		
		
		

Table 4.4: 3D visualization Figures of different objects, first row represents the simulated car roof object, second row represents the curved paper, the third and fourth rows represents the triangular object and the last row represents the practical car roof object..

4.2 Sequential code speed results:

3D profilometry is very useful in many fields like education, medical, entertainments, etc. In some of the 3D profilometry applications the speed is very important, as example in real time applications we need a very fast algorithm. In this section, we will discuss the speed of the saw-tooth algorithm using the c++ language. We will also make a comparison with other algorithms and with the same algorithm but using MATLAB code.

4.2.1 One dimensional shape measurement profilometry based on Fourier transform:

A 3D profilometry based on Fourier transform approach speed results on one line of the image done by Al-Hiari [1]. The steps of this approach are to read the image, convert the image to gray, applying fast Fourier transform (FFT) and doing the other calculations needed. Total time needed for one dimensional shape measurement: **0.6780s**

4.2.2 One dimensional shape measurement profilometry based on saw-tooth fringe pattern algorithm using MATLAB:

Al-Hyari [1] proposed the saw-toot algorithm using MATLAB. The steps of the work are to read the images and convert them to gray, applying noise removal and image enhancement filters, line fitting and applying the height calculations. Total time needed for one dimensional shape measurement: **0.2850s**

From the previous results, we find that the saw-tooth 3D profilometry is faster than the Fourier transform 3D profilometry algorithm. MATLAB is slower than c++ language, in the next section we will discuss the result of using C++ code instead of MATLAB.

4.2.3 One dimensional shape measurement profilometry based on saw-tooth fringe pattern algorithm using C++:

The steps and the speed result of the proposed algorithm using C++ language for one line of the image are reading the images, convert them to gray, applying image enhancement methods, period calculations, line fitting and height calculations. Total time needed for one dimensional shape measurement: 0.08s

As we can see from the previous sections, programming the algorithm using C++ instead of MATLAB enhance the speed and gives better results. Figure 4.2 shows a comparison between the three methods.

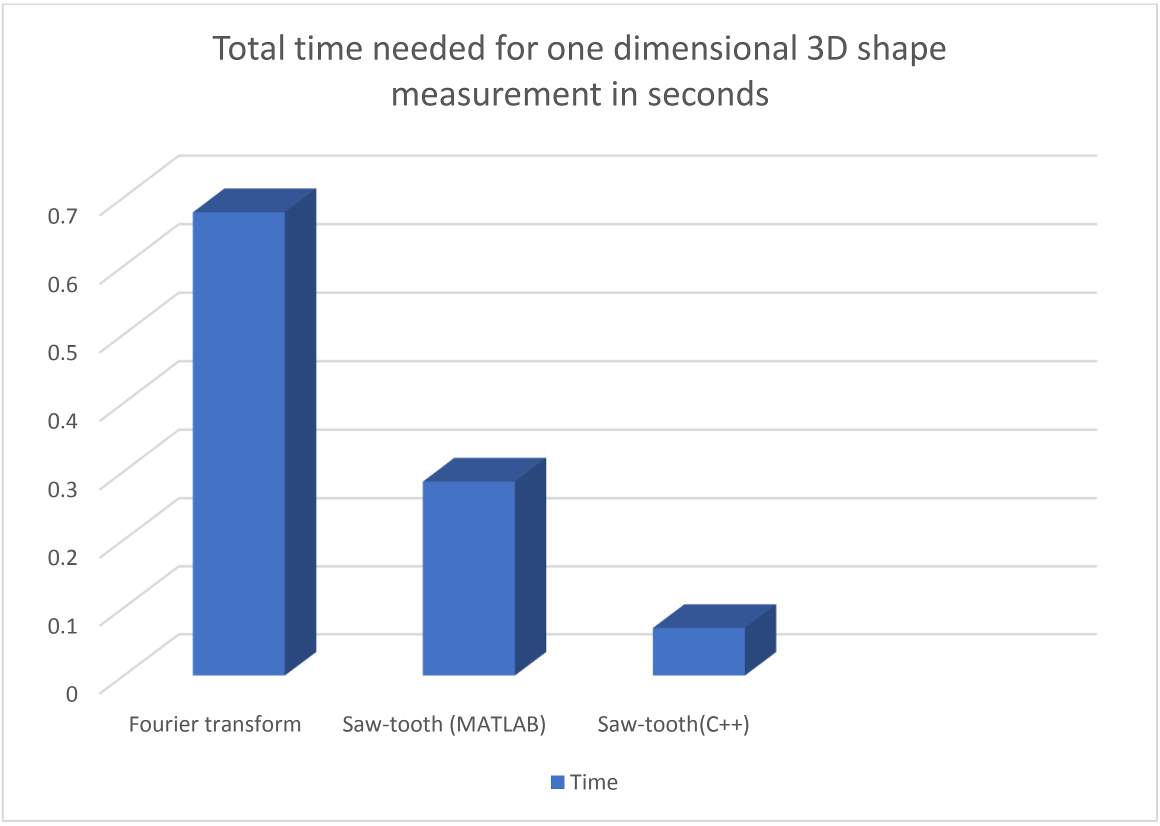


Figure 4.2: A comparison between the three methods.

4.3 Parallel code speed results

The main purpose of the research is to speed up the running time of the algorithm. As we mentioned in the previous section, writing the code using C++ language enhances the speed. In this section, we will discuss the power of parallel processing and how parallelization will enhance the algorithm speed.

Computers these days comes with more than one processor. We will use this feature to speed up the algorithm by distributing the work on the processor to run on the same time which reduce the running time.

Some processors (like Intel core-i series) are designed with possible hyperthreading allowing the threads to operate on a single core with almost zero switching overhead, these are called logical cores and they are not effective as the physical cores because it shares resources with other logical cores operating on the same physical core, so having multiple logical cores can't match the performance of having more physical cores. In this research, we tested the parallel code of the algorithm on 2 and 4 physical cores and on 4 and 8 logical cores (2 and 4 cores bit with hyper threading).

4.3.1 The result of using 2 physical cores:

For this test, we used core i3-2350M . The result will be the average of 10 iterations because of the variation of the results. The variation in the result is because the computer is running windows and other tasks which affect the speed results.

Table 4.5 shows the results of 10 iterations and the average of them when using 2 physical cores.

Iteration	serial	parallel	Speedup
1	0.814	0.404	2.01485
2	0.811	0.453	1.79029
3	0.78	0.464	1.68103
4	0.749	0.409	1.8313
5	0.807	0.464	1.73922
6	0.769	0.417	1.84412
7	0.784	0.403	1.94541
8	0.777	0.403	1.92804
9	0.807	0.464	1.73922
10	0.768	0.449	1.53908
Average	**0.7866 sec**	**0.433 sec**	**1.805256**

Table 4.5: The results of 10 iterations and the average of them when using 2 physical cores in seconds.

As we can see from table 4.5, dividing the images into 2 small images and assigning them to 2 processors will speed up the running time by 1.8 which is close to the number of cores used, see Figures 4.3 and 4.4.

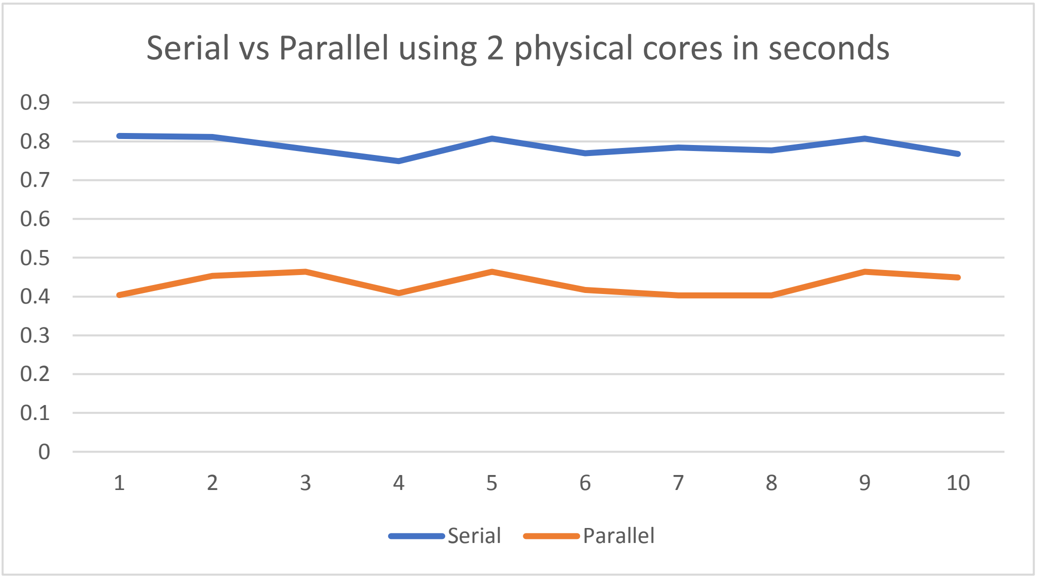


Figure 4.3: Serial vs parallel code using 2 physical code.

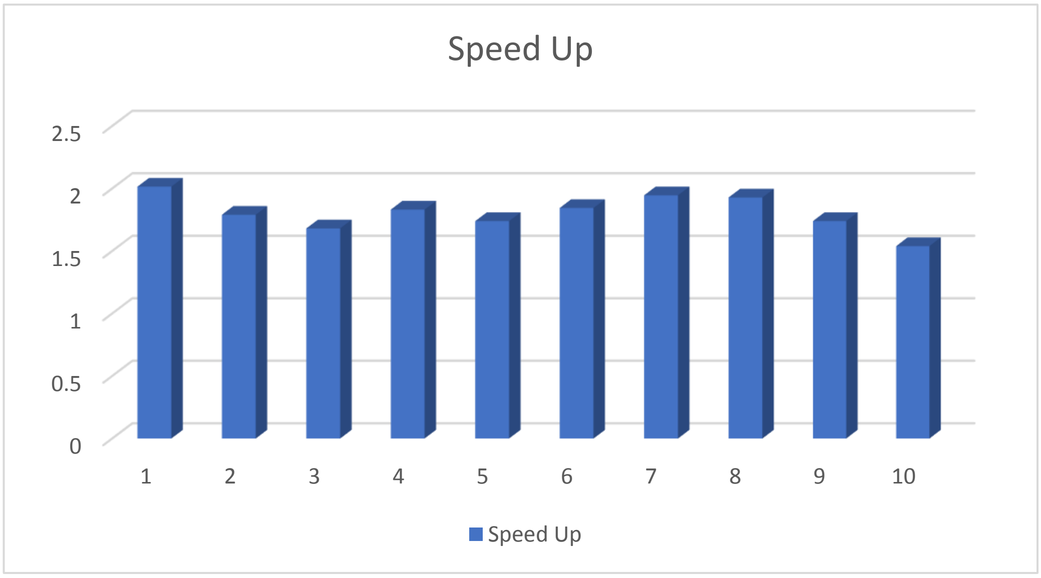


Figure 4.4: The speedup results of using 2 physical cores.

4.3.2 The result of using 4 logical cores (2 physical cores with hyper threading):

For this test, we used core i7-4500U. We divided the images into 4 slices and assigned them to the cores. Table 4.6 shows the results of 10 iterations and the average of them when using 4 logical cores.

Iteration	serial	parallel	Speedup
1	1.14	0.32	3.4625
2	1.133	0.32	3.54062
3	1.017	0.32	3.17812
4	1.109	0.32	3.46562
5	1.035	0.322	3.21429
6	1.131	0.319	3.54545
7	1.105	0.32	3.45313
8	0.995	0.321	3.09969
9	1.095	0.32	3.42188
10	1.114	0.321	3.4704
Average	**1.0874 sec**	**0.3203 sec**	**3.38517**

Table 4.6: The results of 10 iterations and the average of them when using 4 logical cores in seconds.

As we can see from table 4.6, dividing the images into 4 small images and assigning them to 4 logical processors will speed up the running time by 3.4 which is close to the number of cores used but it gives a good result, see Figure 4.5 and 4.6.

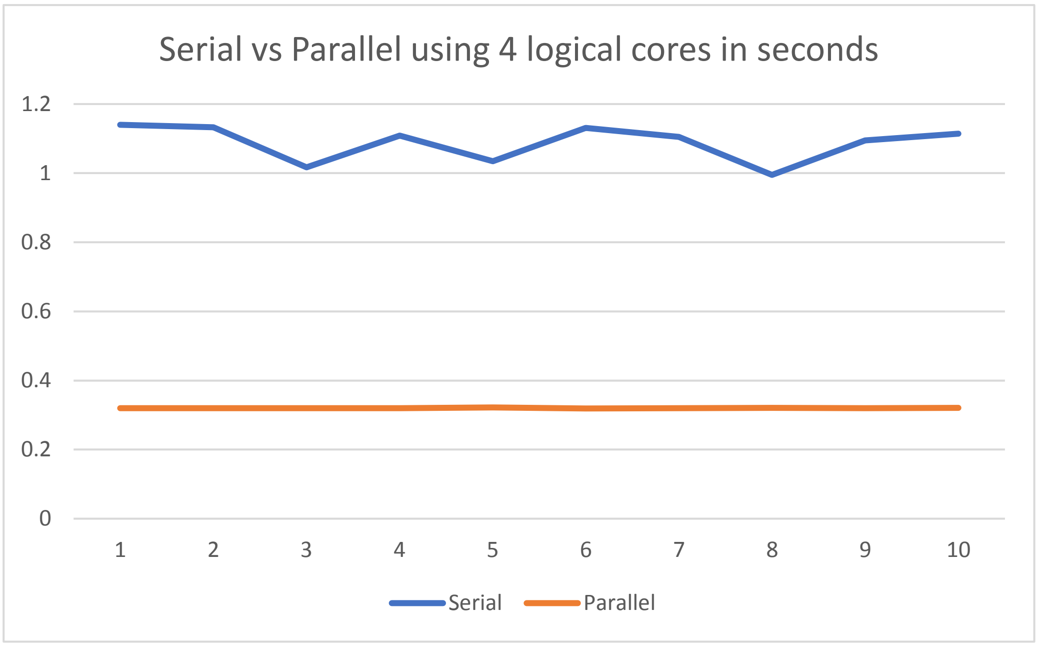


Figure 4.5: Serial vs parallel code using 4 logical cores.

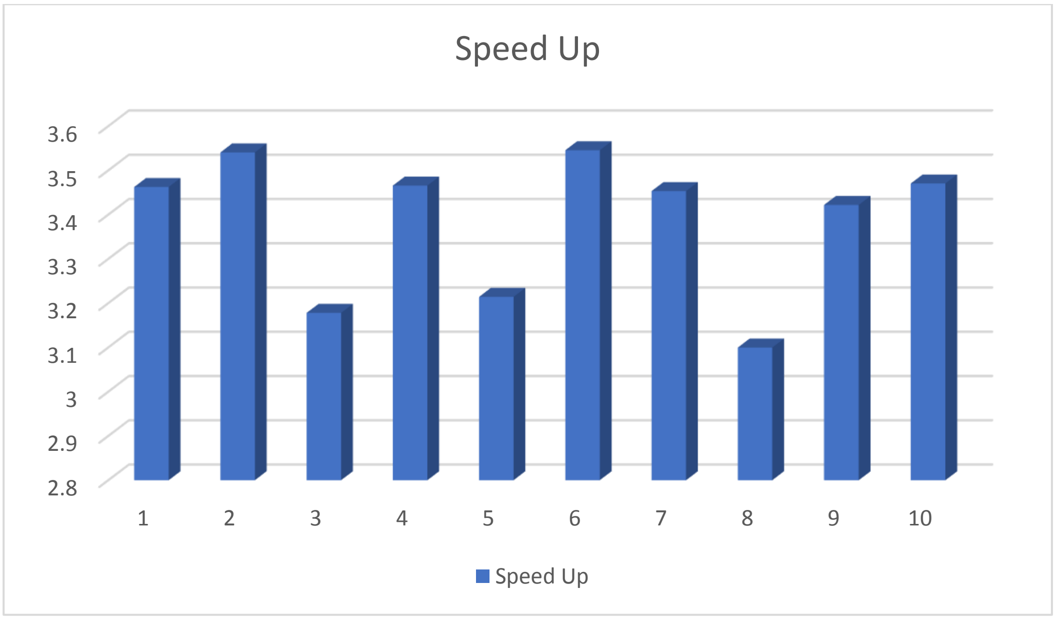


Figure 4.6: The speedup results of using 4 logical cores.

4.3.3 The result of using 4 physical cores

For this test, we used Core i7-4710HQ. We divided the images into 4 slices and assigned them to the cores. Table 4.7 shows the results of 10 iterations and the average of them when using 4 physical cores.

Iteration	serial	parallel	Speedup
1	0.679	0.181	3.75138
2	0.58	0.167	3.47305
3	0.681	0.216	3.15278
4	0.609	0.153	3.98039
5	0.881	0.204	4.31863
6	0.839	0.223	3.76233
7	0.867	0.208	4.16827
8	0.774	0.21	3.68571
9	0.815	0.183	4.45355
10	0.6	0.252	2.38095
Average	**0.7325 sec**	**0.1997 sec**	**3.713**

Table 4.7: The results of 10 iterations and the average of them when using 4 physical cores in seconds.

As we can see from table 4.7, dividing the images into 4 small images and assigning them on 4 physical cores will speed up the running time by 3.713 which is close to the number of cores used, see Figure 4.7 and 4.8.

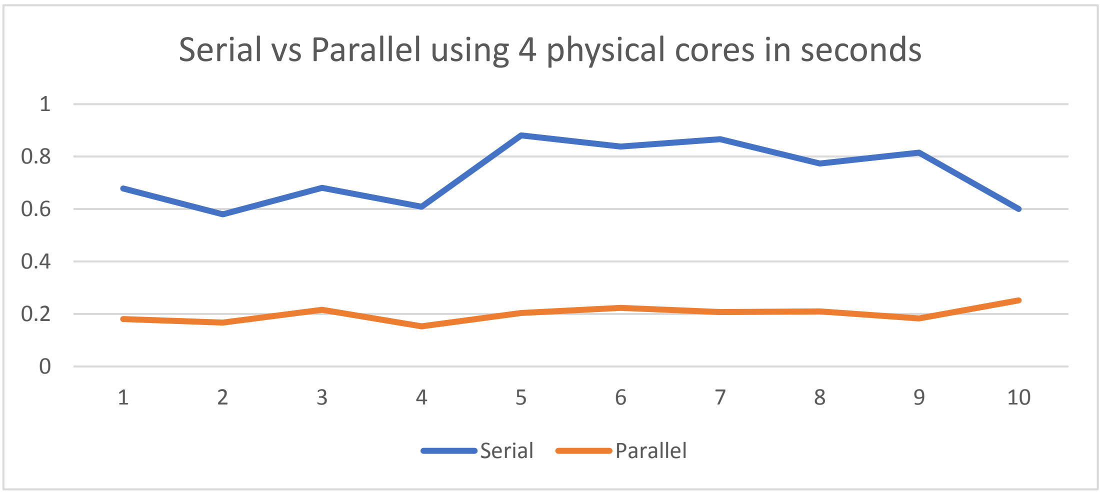


Figure 4.7: Serial vs parallel code using 4 physical cores.

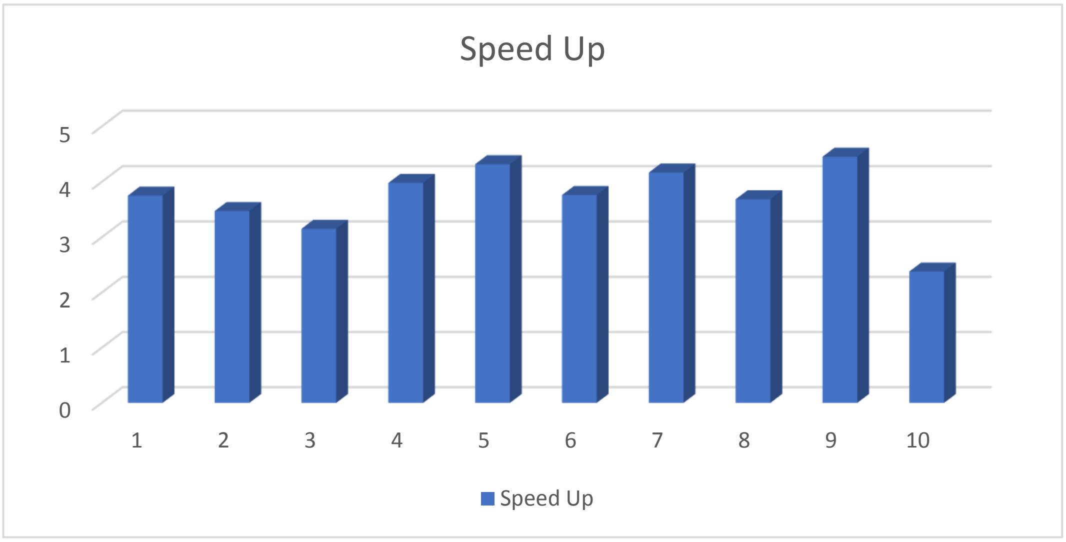


Figure 4.8: The speedup results of using 4 physical cores.

If we make a comparison between a processor with 4 logical cores (2 physical cores with hyper threading) with 4 physical cores as shown in Figure 4.9, we will notice that the logical cores have a less performance than the physical cores, but still have a good performance.

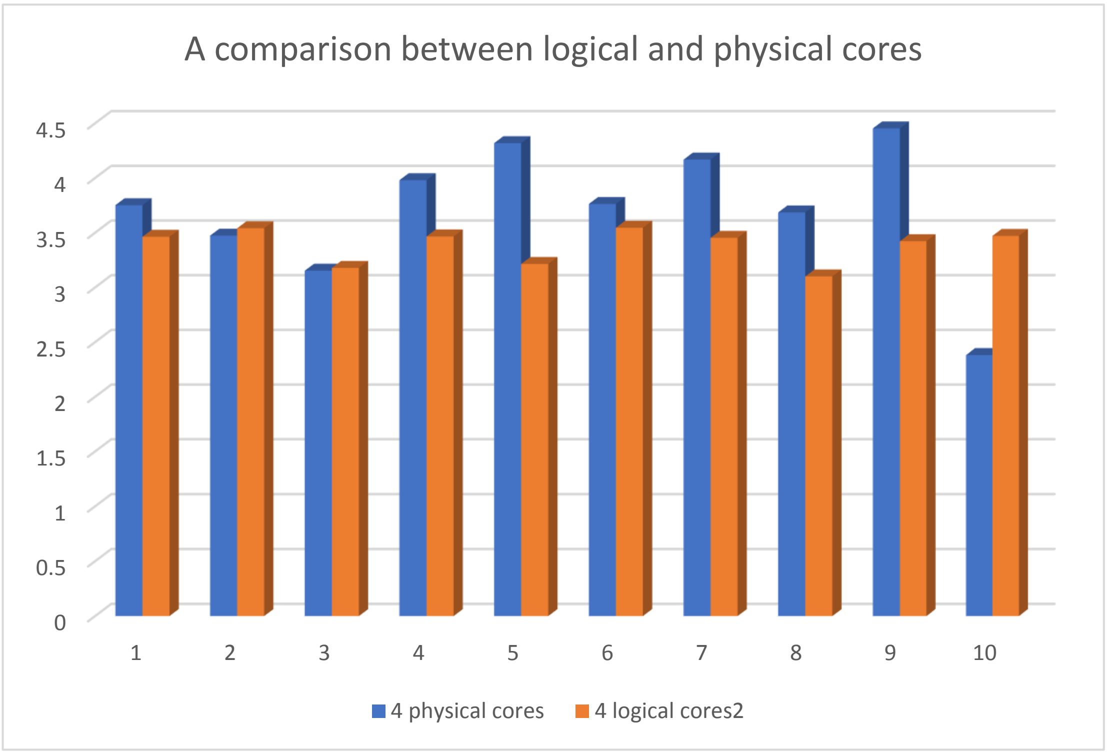


Figure 4.9: A comparison between logical and physical cores.

4.3.4 The result of using 8 logical cores (4 physical cores with hyper threading):

For this test, we used Core i7-4710HQ. We divided the images into 8 slices and assigned them to the 8 logical cores. The result will be the average of 10 iterations because of the variation of the results. Table 4.8 shows the results of 10 iterations and the average of them when using 8 logical cores.

Iteration	**serial**	**parallel**	**Speedup**
1	0.916	0.145	6.3173
2	1.187	0.187	6.348
3	1.207	0.194	6.2212
4	1.208	0.194	6.2268
5	1.207	0.189	6.3862
6	0.905	0.142	6.3732
7	0.917	0.141	6.5036
8	0.925	0.143	6.4685
9	1.199	0.198	6.0606
10	0.82	0.143	5.73427
Average	**1.049 sec**	**0.1676 sec**	**6.264**

Table 4.8: The results of 10 iterations and the average of them when using 8 logical cores in seconds.

As we can see from table 4.8, dividing the images into 8 small images and assigning them to 8 logical cores will speed up the running time by 6.264 which is close to the number of logical cores used, see Figure 4.10 and 4.11

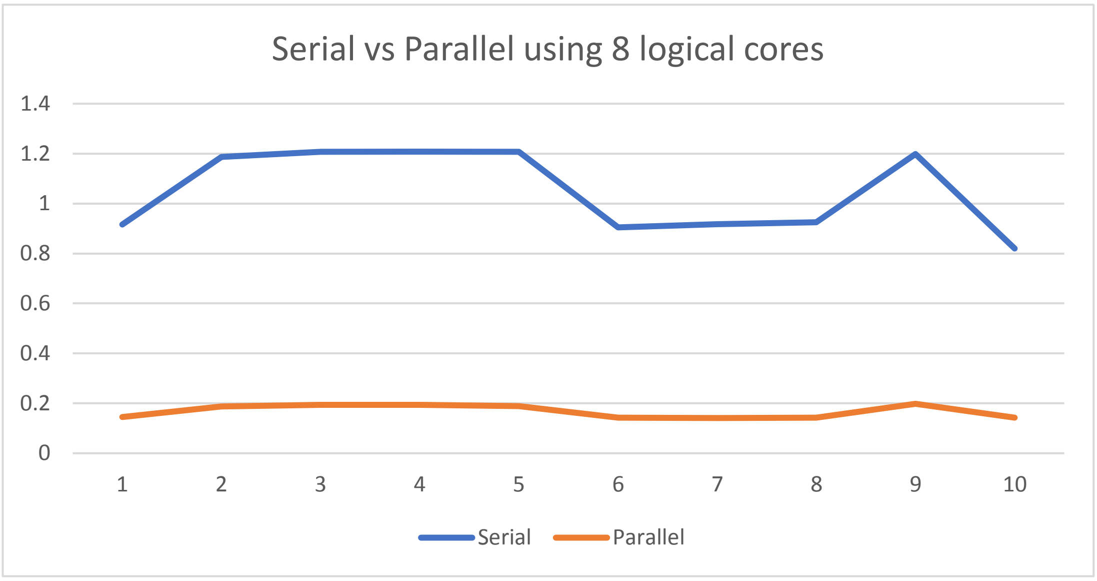


Figure 4.10: Serial vs parallel code using 8 logical cores.

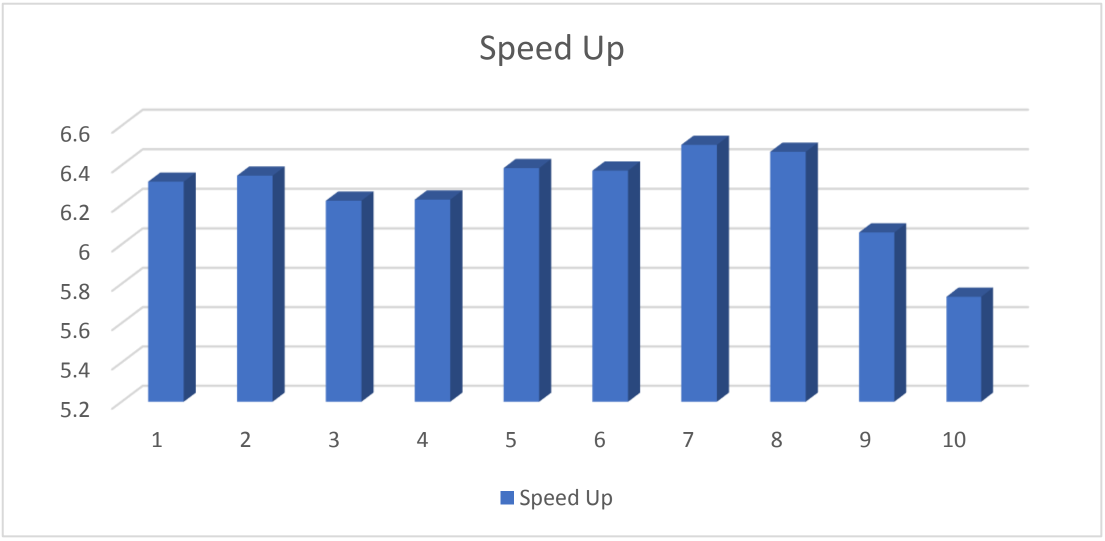


Figure 4.11: The speedup results of using 8 logical cores.

4.3.5 A comparison in speedup between different cores

From the previous tables and Figures we can see that the algorithm using parallelization speed up the running time almost N times when using N physical cores. In the case of using logical cores the performance is less than the same physical cores but still the results are close, see Figure 4.12.

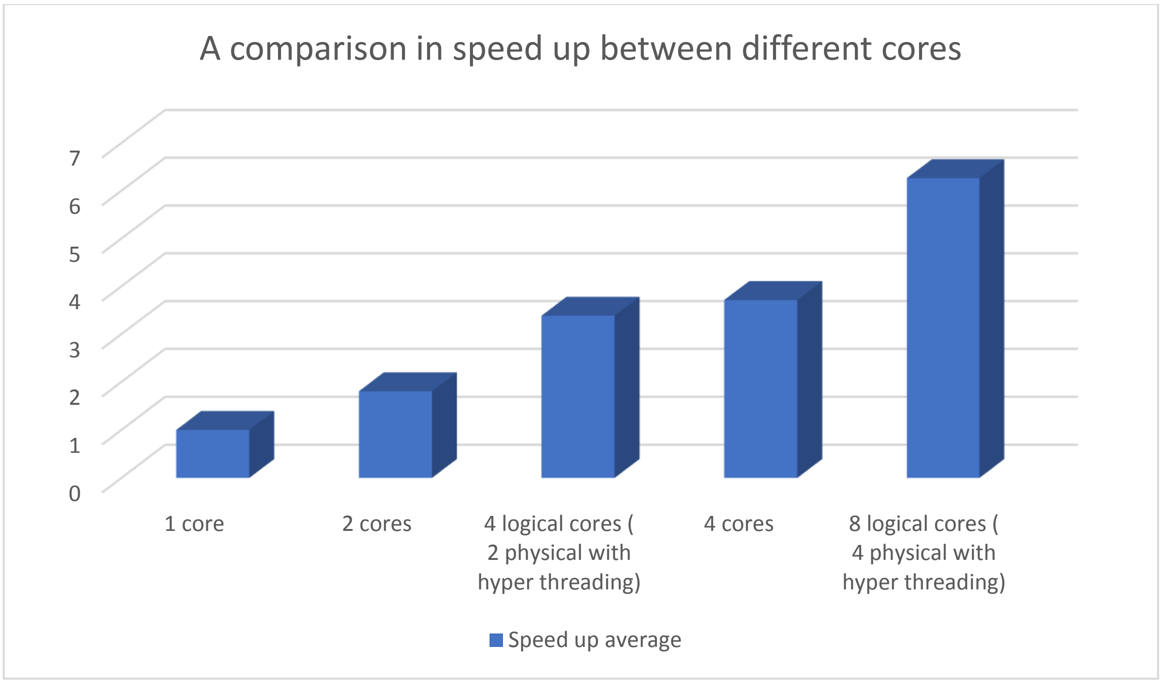


Figure 4.12: A comparison in speed up between different cores.

CHAPTER 5

Discussion, Conclusion and Future work

5.1 Discussion

In this section, the issues we faced during this are discussed and possible solutions to improve the work are presented.

The main problem of the 3D shape measurement was the setup we used because we had low resources. The setup we made consists of simple things like wooden boards and simple tools to avoid any relocation of the projector or the camera during images capturing. We tried to achieve that as possible as we can, but still not perfect. We can improve the setup by using a better equipment that makes the camera and projector fixed and achieves the required angle we need. Figure 5.1 and Figure 5.2 shows an example of using a good setup for the system.

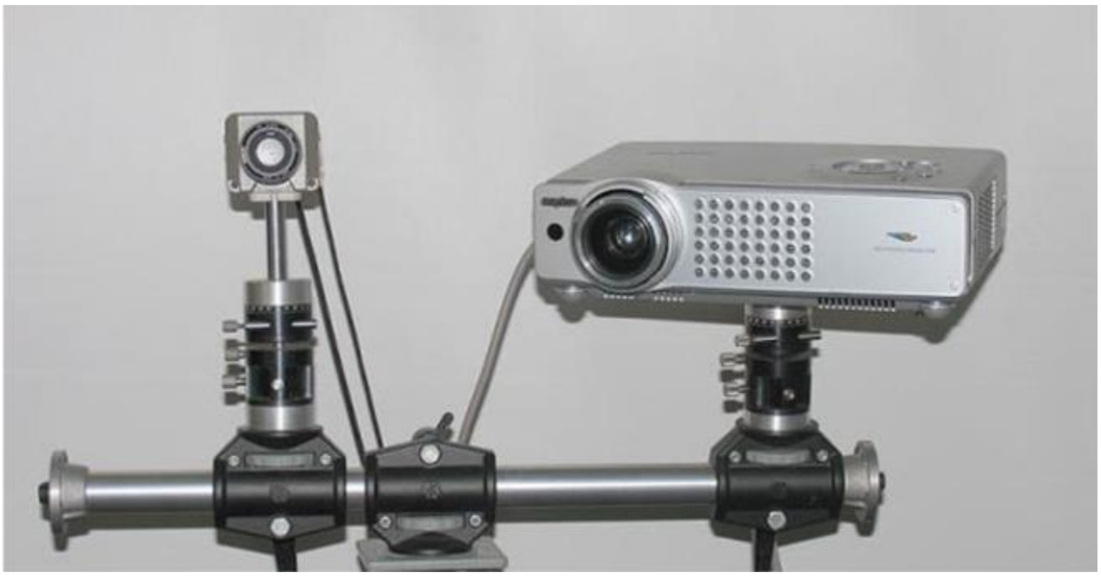


Figure 5.1: Example of system setup [13].

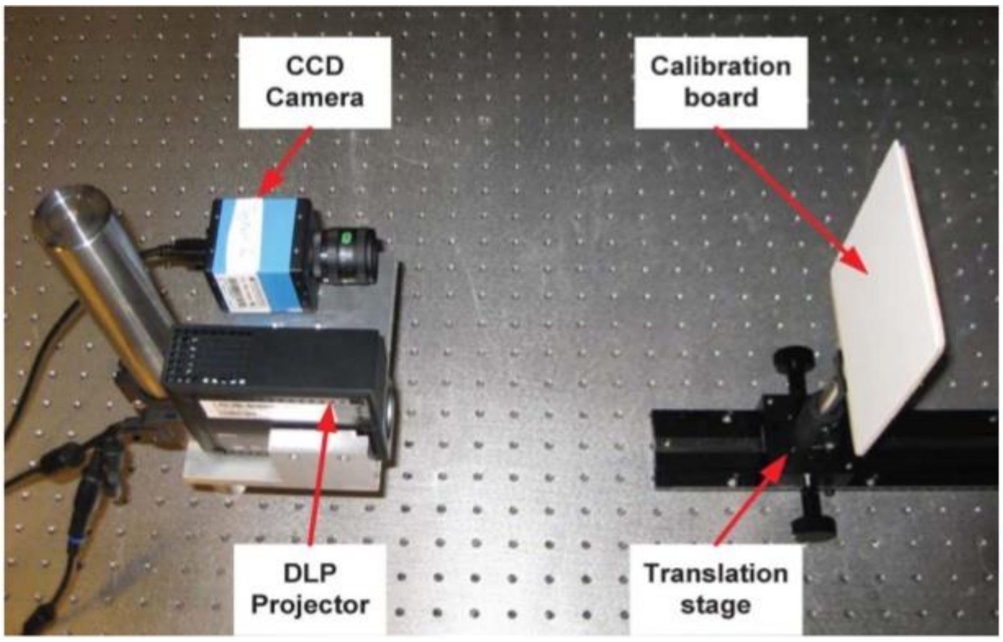


Figure 5.2: Example of system setup [49].

Another way to improve the captured images is to use a better-quality camera and projector. Using a professional camera and a special type of projectors more suitable to our work will reduce the noise and the intensity differences between the images.

Although the speed results are very good and fast we can also get better speed results using the same algorithm. One of the issues of the speed results is that we apply the algorithm on a personal computer which runs the operating system and other tasks at the same time when we run our algorithm on it. So, the time measurement actually are the time needed for running our algorithm and operating system tasks. In order to solve this problem, we can use a dedicated computer system responsible only for running our code.

If we design our system using special setup and we use a dedicated PC for running the program as discussed previously, the system is expected to give better results for 3D shape measurement in terms of quality and speed:

- Using good setup is expected to reduce the noise and intensity variation between the reference and the object images.
- The angle of projection will be accurate and will not change.
- We can use real time camera system to reduce the time of image capturing.
- The reference plane image and the edges of the reference plane image will be captured once only since it will be the same for all of the objects, so the time needed for it will not be measured and the algorithm will be faster.
- The calibration process also will be needed one time only and it will be the same for all of the objects; hence the time of the calibration process will also be ignored which makes the algorithm faster.
- We can use a camera feature to capture the images in a gray level which also reduces the time needed since we don't need to convert the images from color to gray level and that will make the algorithm faster.
- Using a special device to process the algorithm only will make the algorithm faster because it will not run other tasks like the background services of the operating system on the computer.
- We can use a processor that have more cores inside, which will improve the speedup of the system.

5.2 Conclusion

3D shape measurement is a very important field and is used widely. Fringe pattern projection (FPP) is one of the most common methods used for this purpose. Saw-tooth fringe pattern used by Al-Hamdan S. and Al-Hiary S [1] was implemented using MATLAB.

In this research the method is developed using C++ language and parallelization features to get better speed results.

A projector and a camera are used to project the fringe patterns and capture the images. Four images are captured, one for the reference plane without the object, the second image is for the object after placing it on the reference plane. The third and fourth images are for the edges of the reference without the object and with the object respectively.

The images are stored on the computer and then the algorithm is applied and the heights are calculated for the reference plane and for the object using triangular calculations. After that the reference plane heights are subtracted from the object heights to calculate the 3D shape of the object.

The previous steps are implemented using C++ language which gives faster results than MATLAB. The speedup result is about 3.5x faster than the MATLAB code when applied on one dimensional profilometry (calculating height for a cross section in the object under test).

In this research, we used the power of parallelization to improve our speed results. We used Intel Cilk Plus and Intel parallel studio to write the parallel code. The strategy of the code was to divide the images into small images and distribute the work on the available cores of the computer so that each core applies the algorithm on its part of the images.

The parallel code is tested on different number of cores and the speedup is excellent. The parallel code was tested on 2 and 4 physical cores and tested on 4 and 8 logical cores. The speedup results of the parallel code are 1.8x and 3.7x respectively when using 2 and 4 physical cores and 3.4x and 6.2x respectively when using 4 and 8 logical cores.

5.3 Future work

There are many improvements that can be further done on the system to enhance its functionality and get better results, such as:

- Use computers with more number of cores to improve the speed up.
- Improve the speed results using GPU for parallelization.
- Use better projector and camera to capture images.
- Use better setup to make sure that the projector and the camera don't move while capturing different images needed because all images should be taken from the same angle.

References

[1] Al-Hiary S. "*A practical implementation of intensity based saw-tooth fringe projection technique for surface profilometry measurement*", master thesis Yarmouk University, 2015.

[2] D'Apuzzo, Nicola. "*Overview of 3D surface digitization technologies in Europe*". Electronic Imaging 2006. International Society for Optics and Photonics, 2006.

[3] http://www.3shape.com

[4] A. Bellmann, T. Suthau, S. Stoinski, A. Friedri, O. Hellwi, H.-C. Gunga, "*3D Modelling of dinosaurs*". Proc. of the 7th conference on Optical 3-D Measurement Techniques, Vienna, Austria, October 3-5, 2005.

[5] http://optitex.com/the-perfect-fit/

[6] Little, C.; Small, D.; Carlson, J. "*3D Imaging and Modeling for Crime Scene Documentation*". In Proc. ICIP99, Kobe, Japan, 1999; pp. 27-34.

[7] Altsuler, B.; Monson, K. "*Initial progress in the recording of crime scene simulations using 3D laser structured light imagery techniques for law enforcement and forensic applications*". In Proc. SPIE, San Diego, CA, USA, July 1998; 3240, pp. 230-24

[8] Cavagnini, G.L.; Sansoni, G.; Trebesi, M. "*Using 3D range cameras for crime scene documentation and legal medicine*". Proc. SPIE, 3D Imaging Metrology. 2009, 3272, in press.

[9] Wang, Yajun. "*Superfast three-dimensional (3D) shape measurement with binary defocusing techniques and its applications*" (2013).

[10] Han, Xu. "*3D shape measurement based on the phase shifting and stereovision methods*". State University of New York at Stony Brook, 2010.

[11] Blais, F.; Rioux, M.; Beraldin, J.A. "*Practical considerations for the design of a high precision 3D laser scanner system*". Proc. SPIE 1988, 959, 225-246.

[12] Tang, Shouhong, and Yau Y. Hung. "*Fast profilometer for the automatic measurement of 3-D object shapes*". Applied Optics 29.20 (1990): 3012-3018.

[13] Sansoni, G.; Trebesi, M.; Docio, F. "*Fast 3D profilometer based upon the projection of a single fringe pattern and absolute calibration*". Meas. Sci. Tenol. 2006, 17, 1757-1766.

[14] Sansoni, Giovanna, Marco Trebesi, and Franco Docio. "*State-of-the-art and applications of 3D imaging sensors in industry, cultural heritage, medicine, and criminal investigation*" *Sensors* 9.1 (2009): 568-60

[15] Robin C. Colclough "*Stereo conversion & depth determination with mixed 3D graphics*". Viva3D Real-time Stereo Vision ,2016

[16] http://www.saabgroup.com/About-Saab/Newsroom/Image-bank/

[17] Gorthi, Sai Siva, and Pramod Rastogi. "*Fringe projection techniques: whither we are?* ". Optics and lasers in engineering 48. IMAC-REVIEW-2009-001 (2010): 133-140.

[18] Cao, P., Xi, J., Yu, Y., & Guo, Q. "*Digital fringe profilometry based on triangular fringe patterns and spatial shift estimation*". SPIE Sensing Technology+ Applications. International Society for Optics and Photonics, 2014.

[19] Moore, Gordon E. (1965). "*Cramming more components onto integrated circuits*". Electronics Magazine. p. 4. Retrieved 2006-11-11

[20] https://www.fixstars.com/en/opencl/book/OpenCLProgrammingBook/parallel-computingsoftware/

[21] https://computing.llnl.gov/tutorials/parallel_comp/

[22] https://www.cs.cmu.edu/~fp/courses/15213-s06/lectures/27-multicore.pdf

[23] http://www.cs.sfu.ca/~fedorova/Teaing/CMPT886/Spring2007/papers/hyper-threading.pdf

[24] www.Intel.com

[25] Kejariwal, A., Nicolau, A., Banerjee, U., Veidenbaum, A. V., & Polychronopoulos, C. D. "*Cache-aware iteration space partitioning*". Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming. ACM, 2008.

[26] Munick, S.: "*Advanced Compiler Design and Implementation*". Morgan Kaufmann (1997).

[27] Wolfe, M.: "*High Performance Compilers for Parallel Computing*". Addison-Wesley (1996).

[28] Membarth, R., Hannig, F., Tei, J., Körner, M., & Eckert, W. "*Frameworks for multicore architectures: a comprehensive evaluation using 2D/3D image registration*". Architecture of Computing Systems-ARCS 201 Springer Berlin Heidelberg, 201 62-73.

[29] Leiserson, C.: "*The Cilk++ Concurrency Platform*". In: Proceedings of the 46th Annual Design Automation Conference. pp. 522–527. ACM (2009)

[30] https://www.cilkplus.org/

[31] Zhang, Z., Huang, S., Meng, S., Gao, F., & Jiang, X. "*A simple, flexible and automatic 3D calibration method for a phase calculation-based fringe projection imaging system*". Optics express210 (2013): 12218-12227.

[32] Chen, L., Quan, C., Tay, C. J., & Fu, Y. "*Shape measurement using one frame projected saw-tooth fringe pattern*". Optics communications, (2005), 246(4), 275-284.

[33] Lange, Robert. "*3D time-of-flight distance measurement with custom solid-state image sensors in CMOS/CCD-technology*". Diss., Department of Electrical Engineering and Computer Science, University of Siegen (2000).

[34] Su, X., en, W., Zhang, Q., & ao, Y. "*Dynamic 3-D shape measurement method based on FTP*". Optics and Lasers in Engineering 36.1 (2001): 49-64.

[35] Asai, Ryo, and Andrey Vladimirov. "*Intel Cilk Plus for complex parallel algorithms: "Enormous Fast Fourier Transforms" (EFFT) library*". Parallel Computing 48 (2015): 125142.

[36] FFTW Library. http://www.fftw.org/.

[37] Sinha, S. N., Frahm, J. M., Pollefeys, M., & Genc, Y. "*GPU-based video feature tracking and mating*". EDGE, Workshop on Edge Computing Using New Commodity Architectures. Vol. 278. 2006

[38] S. Birfield, "*KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker*". Http://www.ces.clemson.edu/stb/klt, Nov 2005.

[39] Lowe, David G. "*Distinctive image features from scale-invariant key points*". International journal of computer vision 60.2 (2004): 91-110.

[40] Saxena, Sanjay, Neeraj Sharma, and Shiru Sharma. "*Image processing tasks using parallel computing in multi core architecture and its applications in medical imaging*". International Journal of Advanced Resear in Computer and Communication Engineering 2.4 (2013): 2278-102

[41] Maity, Subhendu, et al. "*Role of parallel computing in numerical weather forecasting models*". IJCA Special Issue on International Conference on Computing, Communication and Sensor Network CCSN2012. Vol. 4. 2013.

[42] Barbosa, Jorge, João Tavares, and Armando J. Padilha. "*Parallel image processing system on a cluster of personal computers*". International Conference on Vector and Parallel Processing. Springer Berlin Heidelberg, 2000.

[43] Li, X. L., B. Veeravalli, and C. C. Ko. "*Distributed image processing on a network of workstations*". International Journal of Computers and Applications 25.2 (2003): 136-145.

[44] Al-Hamdan S and Hamad H. "*Non-Contact Profile Measurement Using Linear Fringe Patterns*". SSCCII-2004 International Symposium of Santa Caterina on allenges in the Internet and Interdisciplinary Resear, Amalfi, Italy Jan 29 – Feb 1, (2004).

[45] Al-Hamdan S and Hamad H. "*Profile Measurement using Intensity-Based SawTooth Structured Light Fringe Patterns*". ABHATH AL-YARMOUK: "Basic Sci. & Eng." Nov. 19, 2006.

[46] Weisstein and Eric. "*Least Squares Fitting - from Wolfram MathWorld* " [online]. Available: http:// Mathworld.wolfram.com (1999-2015).

[47] https://www.visualstudio.com/

[48] http://www.opencv.org/about.html

[49] Xu, Ying, et al. "*Phase error compensation for three-dimensional shape measurement with projector defocusing*". Applied optics 50.17 (2011): 2572-2581.

تحسين السرعة و استخدام البرمجة المتوازية في تقنية اسقاط نمط ضوئي على شكل أسنان المنشار

إعداد	إشراف
محمد ماجد أحمد سليم	الدكتور سامي الحمدان

ملخص

تقنية إنتاج خارطة ثلاثية الأبعاد لمجسم معين تتم عن طريق جمع و تحليل المعلومات المتعلقة بالمجسم. هذه التقنية مستخدمة في العديد من المجالات مثل المجال الصناعي و المجال الطبي و الالعاب و غيرها من المجالات العديدة. اسقاط نمط ضوئي معين و استخدام انعكاس هذا النمط الضوئي و تحليله يعد من اهم التقنيات المستخدمة في انتاج الخرائط ثلاثية الأبعاد. في هذا البحث تم تحسين سرعة تقنية اسقاط نمط ضوئي على شكل أسنان المنشار و التي طورت حديثا من قبل الحياري باستخدام برمجية MATLAB. في هذا البحث تم تحويل لغة البرمجة المستخدمة الى لغة C++ و استخدام البرمجة المتوازية حيث تم تجربتها على عدة حواسيب متعددة النواة. الخطوتان الرئيسيتان لتقنية اسقاط النمط الضوئي على شكل أسنان المنشار هما: أولا اسقاط النمط الضوئي على سطح مستوي بدون المجسم ثم مع المجسم, الخطوة الثانية هي التقاط الصور للسطح بدون المجسم و مع المجسم و تحسين الصور و تحليلها و استخراج النمط الثلاثي الابعاد للمجسم. تمت تجربة التقنية المستخدمة و حساب سرعتها و كانت سرعة البرنامج المتسلسل المستخدم في هذا البحث هي 3.5 ضعف سرعة البرنامج المستخدم من قبل الحياري.ثم تم تجربة البرنامج الذي طبقت فيه خاصية البرمجة المتوازية على عدة حواسيب متعددة النواة . تمت تجربة البرنامج اولا على حاسوب ثنائي النواة و كانت السرعة 1.8 ضعف البرنامج المتسلسل. و تمت ايضا تجربته على حاسوب رباعي النواة و كانت النتيجة 3.7 ضعف البرنامج المتسلل. تمت أيضا تجربة البرنامج على حواسيب متعددة النواة و تستخدم فيها خاصية Hyper Threading والتي تجعل من نواة الحاسوب تظهر و كأنها نواتين افتراضيتين. النتيجة كانت 3.4 ضعف البرنامج المتسلسل عند تجربته على حاسوب ثنائي النواة مع خاصية Hyper Threading أي 4 نواة افتراضية. تم أيضا تجربته على حاسوب رباعي النواة مع خاصية Hyper Threading اي 8 نواة افتراضية و كانت النتيجة 6.2 ضعف البرنامج المتسلسل.