Linear Stochastic Sampling(LSS) in Computer Graphics

By Jun Ro

A Thesis

Submitted of the Requirements for the Degree of

Master of Science in Engineering with an Emphasis in

Computer Science

Northern Arizona University

May 2018

Approved:

James Palmer, Ph.D., Chair

Dieter Otte, Ph.D.

Mike Gowanlock, Ph.D.

ABSTRACT

Linear Stochastic Sampling (LSS) in Computer Graphics

Jun Rao

This thesis presents a new technique to randomly sample from the large datasets in linear order. Existing stochastic sampling methods work well but are limited to the small datasets. If existing stochastic sampling methods are implemented for large datasets, thrashing results, wherein the Operating System will spend most of its time swapping the pages of memory rather than executing instructions.

We make two contributions to our research. First, we derive explicit formulas that minimize the stochastic sampling time and generate a higher quality of the output images at the same time. Second, we analyze the new algorithm in the context of visual quality, memory usage, and performance. The results of our analysis show that this technique is competitive with other stochastic sampling methods while avoiding thrashing and using computer memory more efficiently.

Key Words: stochastic, sampling, linear algorithm, thrashing, efficiency

Acknowledgments

I thank all my friends and family for their support through my entire thesis. Without your help, this paper would not exist. I love you all.

For the knowledge that served as the foundation for this work, I thank all the NAU faculty whose tutelage I was privileged to receive. Foremost among them are Dr. Palmer, Dr. Otte, Dr. Fofanov, and Dr. Gowanlock whose teaching efforts and joy for the field set me toward a lifelong love of investigation and discovery through computation. It has honestly been a pleasure.

For the laughter and encouragement that fueled this work, I thank all my friends. I love you all.

For the belief that I could even begin this program, I thank my parents. They have never failed to be wise counselors. No one has ever believed in me as unflaggingly as they have, and this has given me the strength to overcome obstacles as significant as I may ever encounter. I love you both.

For the motivation to finish this program and the drive to do what I know I should, I thank my lovely and encouraging girlfriend Ying. She is a prize of which I am not worthy. Nonetheless, she loves me and supports me continually. I love you my darling.

For a word in which experience like these can be had, for the continued shepherding that has led me down this path, for a worldview in which I may live at peace, and for the part of my character that is the love of learning, I thank you, God.

Table of Contents

1 Introduction 1

1.1 Motivation 1

1.2 Thesis Organization 5

2 Background 6

2.1 Hardware Level 8

2.2 Software Level 11

2.3 Existing Sampling Methods 13

2.3.1 Poisson Disk Sampling 13

2.3.2 Jittered Sampling 14

2.3.3 N-rooks Sampling 15

2.3.4 Multi-Jittered Sampling 17

2.3.5 Gaussian Pyramid Sampling 18

2.4 Image Organization 20

2.4.1 JPEG 20

2.4.2 GIF 21

2.4.3 PNG 21

2.4.4 PPM 22

2.4.5 Image Structure 23

3 Linear Stochastic Sampling 24

3.1 Principle 26

3.2 Variables 28

3.2.1 Scale 28

3.2.2 Span 29

3.2.3 Average 30

3.3 Structure 32

3.4 Algorithm 34

4 Experiment 39

4.1 Simple Random Sample 40

4.2 Experiment Design 42

4.3 Formula 45

4.3.1 Equation 45

4.3.2 Coefficient 49

4.3.3 Constant 55

4.3.4 Validation 57

5 Analysis 59

5.1 Visual Quality 60

5.2 Memory Usage 66

5.3 Performance 68

5.4 Survey 71

6 Conclusion and Future work 72

6.1 Conclusion 73

6.2 Critical Value 75

6.3 Convenience Interval 77

List of Figures

1 Introduction 1

1.1 The Pixel Array and Sampling Direction 1

1.2 Thrashing Example 3

2 Background 6

2.1 Computer Memory Hierarchy 9

2.2 Memory Ussage Without Virtual Memory 11

2.3 Memory Ussage With Virtual Memory 13

2.4 Poisson Disk Sampling Pattern 14

2.5 Jittered Sampling Pattern 15

2.6 N-rooks Sampling Pattern 16

2.7 The Multi-jittered Sampling Pattern 17

2.8 The Gaussian Pyramid Sampling Pattern 19

2.9 The PPM Image File Format 22

2.10 Internal Image Sturcture 23

3 Linear Stochastic Sampling 24

3.1 Sampling Direction in LSS Algorithm 26

3.2 Termination in LSS Algorithm 28

3.3 2D Pattern in LSS 27

3.4 The Variable *Scale* in the LSS Algorithm 29

3.5 The Variable *Span* in the LSS Algorithm 30

3.6 The Variable *Sample Average* in the LSS Algorithm 31

3.7 Structures in the LSS Algorithm 34

3.8 Step 1 of the LSS Algorithm 34

3.9 Step 2 of the LSS Algorithm 34

3.10 Step 3 of the LSS Algorithm 34

3.11 Step 4 of the LSS Algorithm 34

3.12 Workflow of the LSS Algorithm 34

4 Experiment 39

4.1 Pseudocode for the LSS stochastic sampling algorithm 40

4.2 Experiments 42

4.3 Experimental Convergnce 43

4.4 Experimental Design 45

4.5 Span Power VS Number of Experiment Trials 45

4.6 C_1 VS Number of Experiment Trials 53

4.7 Plot of Derived Function 58

5 Analysis 59

5.1 Visual Quality Test..... 60

5.2 Comparison of Visual Quality..... 62

5.3 Image Used for Performance..... 66

5.4 Pseudocode of the LLS Algorithm and the Stochastic Sampling Algorithm..... 68

5.5 Output Images Generated by LLS and the Stochastic Sampling Algorithm..... 69

6 Conclusion and Future work 72

6.1 Output Quaality VS Sample Average 75

6.2 Using a Critical Value to Mark Converage..... 77

List of Tables

1 Using Convergence to Find the Span Power 47

2 Relationhip Between $\boldsymbol{k_1}$ and Scale 51

3 Using Convergence to Find the $\boldsymbol{c_1}$ constaant 53

4 Relationship between the Scale and the Sample Average 55

5 Comparing Experimental and Derived Values 57

6 Compaaring LSS and SS Performance 70

Chapter 1

Introduction

1.1 Motivation

In computer programming, when confronted with a large program, such as gigapixel image sampling, we use virtual memory as we do not have enough memory to store the whole image. In essence, separate memory spaces are used, which means the Operating System includes software for managing the movement of pages between main memory and secondary memory.[1]

Because the pixel in the image is stored in a single-dimensional (one-dimensional) array in the computer memory, if we want to maximize the usage of the memory on the computer, the index of sample points in the pixel array should be arranged in ascending order (See Figure 1.1). Thus, each page will be loaded into RAM only once in the entire programming.

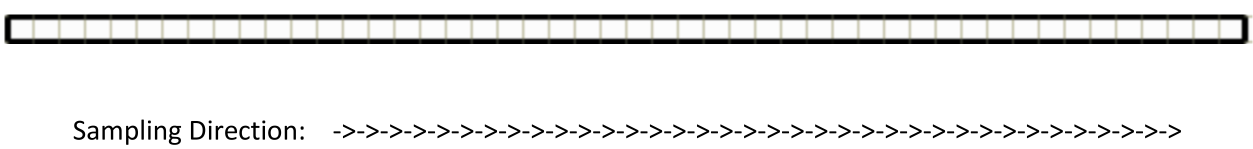


Figure 1.1: The pixel array and sampling direction in the in the Linear Stochastic Sampling algorithm. The pixel is stored in a single-dimensional array in the big image, and the sampling direction is in ascending order in the Linear Stochastic Sampling (LSS) method.

However, research so far has been mainly focused on anti-aliasing, while the existing efficient stochastic sampling techniques do not readily permit to get samples from large data sets

in ascending order.[2] In particular, Poisson and jittered sampling, have been developed and analyzed .[3] Jittered Sampling is a refinement of the classical Monte Carlo sampling method.[4]Among existing stochastic sampling method, jittering remains popular as it reduces aliasing substantially.

The sampling pattern in the existing stochastic sampling techniques (for more details see section 2.2) is too random, which means the stochastic sampling algorithm has to get samples back and forth in the pixel array.

In gigapixel images sampling, the big image is divided into equal fixed-size chunks, known as pages, which are assigned to available pieces of memory on the computer, like frames, or page frames (See Figure 1.2).

When memory is insufficient to sample gigapixel images, then when the Operating System brings one page in, it must throw another out. With existing stochastic sampling method, we need to load samples back and forth in the pixel array, which means the OS throws out a page it has already used and then will have to access that page again immediately.

Too much of this leads to a condition known as **thrashing**: The System spends most of its time swapping pieces rather than executing instructions. If we need to sample back and forth in the Pixel Array, the OS has to load the same page into the RAM multiple times.

In Figure 1.2, if the 10000th sample point, the 10010th sample point, and the 10050th sample point are on the same page. And there is no free space in the RAM; the OS needs to throw out this page, then the OS will have to get the same page again immediately.

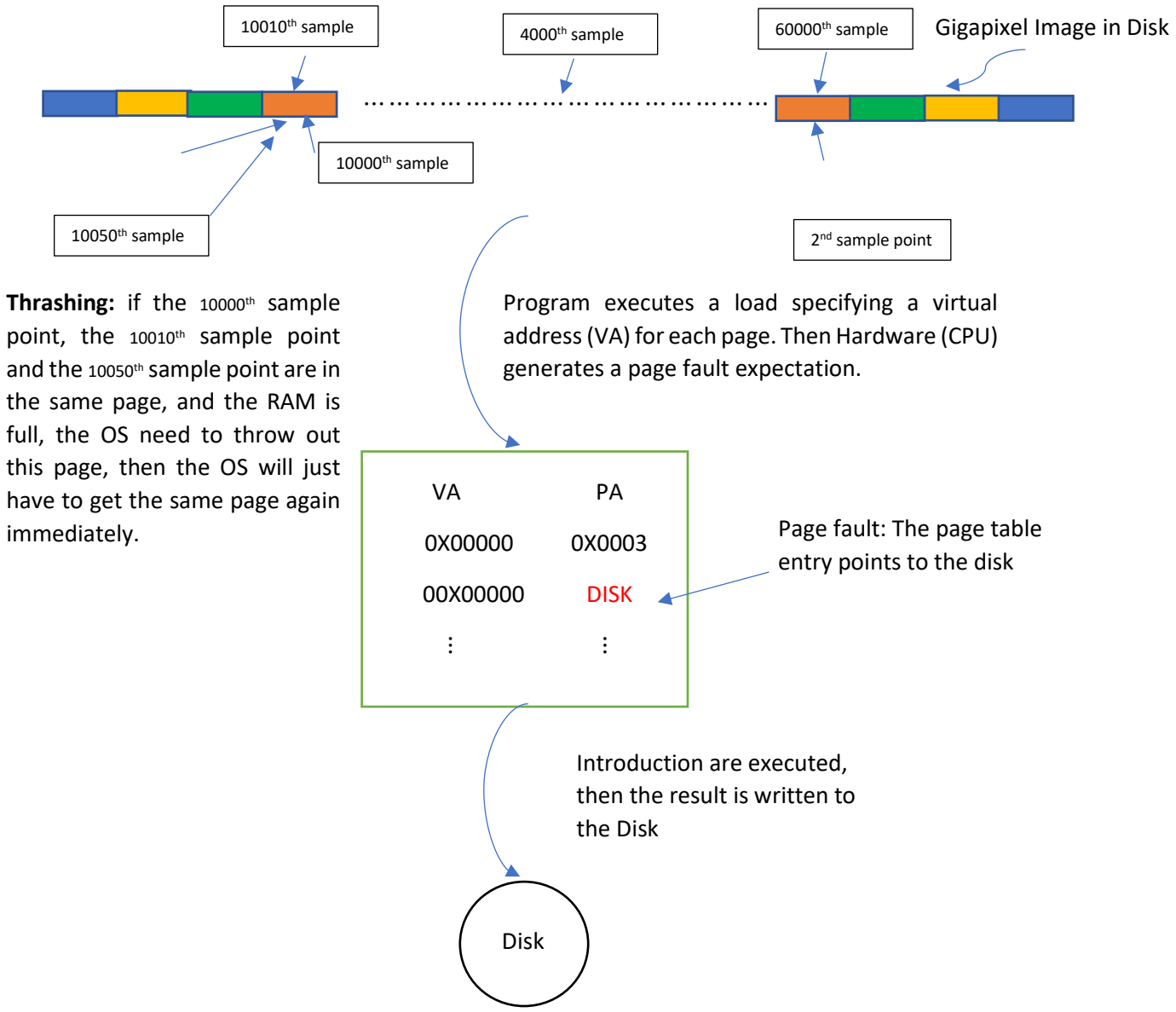


Figure 1.2: Thrashing Example. If we use the existing stochastic sampling methods for large datasets, the result is thrashing, which means the Operating System will spend most of its time swapping the pages of memory rather than executing instructions.

Otherwise, in the existing stochastic sampling methods, it is immensely comfortable to create a page fault, which is a type of interrupt, called trap, raised by computer hardware when a running program accesses a memory page that is mapped into the virtual address space, but not loaded into main memory.[5] Because disks are much slower than the RAM, if we have a page fault, it will take a long time. If we have enough RAM in the computer, then we will never use the paging technique in gigapixel images sampling.

In this paper, we introduce a new stochastic sampling method based on multi-jittered sampling and the Gaussian Pyramid sampling, which is named the Linear Stochastic Sampling Algorithm. We intend to intend to improve upon existing stochastic sampling method, by enhancing the use of primary computer memory and avoiding thrashing when randomly sampling in large datasets.

1.2 Thesis Organization

In Chapter 1, we introduce the motivation for the development of the Linear Stochastic Sampling Algorithm (LSS). After this critical term is defined, we present an overview of the topics to be discussed and introduces a small number of conditions that provide coverage of this research.

In Chapter 2, we show the backgrounds of the gigapixel images sampling. It reviews the architecture and the software of a computer. Next, we introduce some related stochastic sampling methods of the Linear Stochastic Sampling Algorithm. Finally, we provide some basic ideas of the image format, which will be used in the experiment.

The purpose of Chapter 3 is to introduce the concepts of the Linear Stochastic Sampling Algorithm. We start with a platform, which specifies the architecture of the Linear Stochastic Sampling Algorithm. Then we show the pseudocode and explain each step of the Linear Stochastic Sampling Algorithm.

In Chapter 4, we analyze the preconditions of the Linear Stochastic Sampling Algorithm which are connected explicitly to the motivation given in Chapter 1. Then, we design an experiment to derive a function that presents the internal relationship between each variable of the Linear Stochastic Sampling Algorithm.

Chapter 5 provides all experiment details for assessing the Linear Stochastic Sampling Algorithm and illustrates related aspects of the study, such as the quality of the output image, the memory usage, and the speed of the Linear Stochastic Sampling.

Finally, Chapter 6 offers a distillation of this work about all previous discussion and presents a sketch of two possible future studies.

Chapter 2

Background

This chapter discusses the theoretical background that serves as a basis for the Linear Stochastic Sampling algorithm.

It begins with a section explaining the computer organization and architecture. We present computing systems as a series of layers, starting with low-level hardware and progressing to higher-level software, such as Operating System.

Next, we discuss five well-known stochastic sampling algorithms, which are Poisson Disk Sampling algorithm, jittered sampling algorithm, N-rooks sampling algorithm, Multi-Jittered Sampling algorithm, and Gaussian pyramid Sampling algorithm. In this section, we will focus on the description of the algorithm along with a consideration of each method's advantages and disadvantages in the large datasets sampling.

Finally, we discuss five popular image formats, demonstrate the fundamental principles of the test image format in our research, and analyze the reasons why we choose 'PPM6' as the test image format in the experiment.

In Geoinformatics, we encounter big images such as gigapixel images or the Mar's Image from NASA every day. These images contain a multitude of data and scientists use them to carry out their results.

When confronted with a significant amount of data, such as gigapixel images, we seek to summarize the data to capture the essence of the data with as few numbers as possible .[6] There is an old saying that "without data, you are just another person with an opinion."

In data science, sampling allows us to collect data sensibly to generate a representative sample. In particular, a sampling unit is an intersection of the channel and a pixel in an image.[7] By using the data sample, we can describe, analyze and estimate the big picture.

Because the data are discrete in the image, we can quickly get a sample of graphics by using different sampling methods. One such sampling method is the stochastic sampling, which is a process to generate uniformly distributed samples and compute pixel values randomly.

To understand many things that can affect sampling performance, we need to have a rudimentary understanding of the building blocks that make up a computer and the operating system that also manages secondary and I/O (input/output) devices.

2.1 Hardware Level

We already know that most computers are built based on the Von Neumann model, which is centered on memory and the programs that perform the processing are stored in memory.

A computer consists of a processor, memory, and I/O components. These components are interconnected in some fashion to achieve the primary function of the machine, which is to execute programs. The central processing unit (CPU) is the heart of the computer. The purpose of the CPU is to locate and run the program instructions. It carries out arithmetic operations such as addition, subtraction, production, division, load, jump and so on. The CPU also retrieves data from storage and input/output devices and sends data back.[8]

The computer uses two kinds of storage:

1. Primary storage, also called random access memory (RAM), is fast and expensive. A more appropriate name of RAM is a read-write memory, which means the RAM used to store program and data, which is used for executing the program. However, RAM is volatile, which means when power is turned off, the data of the memory are lost.
2. Secondary storage, usually a hard disk, which can either be a magnetic or solid state. Because the data in secondary memory cannot be accessed directly by the CPU, the content must be transferred to main memory when the data are needed.

Engineers use a set of electrical lines to interconnect the CPU, the RAM, the hard disk and other devices in the computer and call it bus .[9] By using the bus, data are transferred from the system memory and peripheral devices to the CPU and back. But the latency of the data transfer based on the “distance” between the computer memory and the processor in the computer.

The closer memory is to the processor in the computer, the faster, smaller, and more expensive it will be. When the computer memory is further from the central processor, it is tolerable to afford longer access time.

Hence, the slower technologies are used for this memory, such as hard disk, while the faster techniques are used in the CPU, such as RAM, Cache, and CPU registers.

These smaller memories have better performance in data transfer, and they also have a higher cost than computer memories found lower in Figure 2.1, which represents the memory hierarchy in the computer.

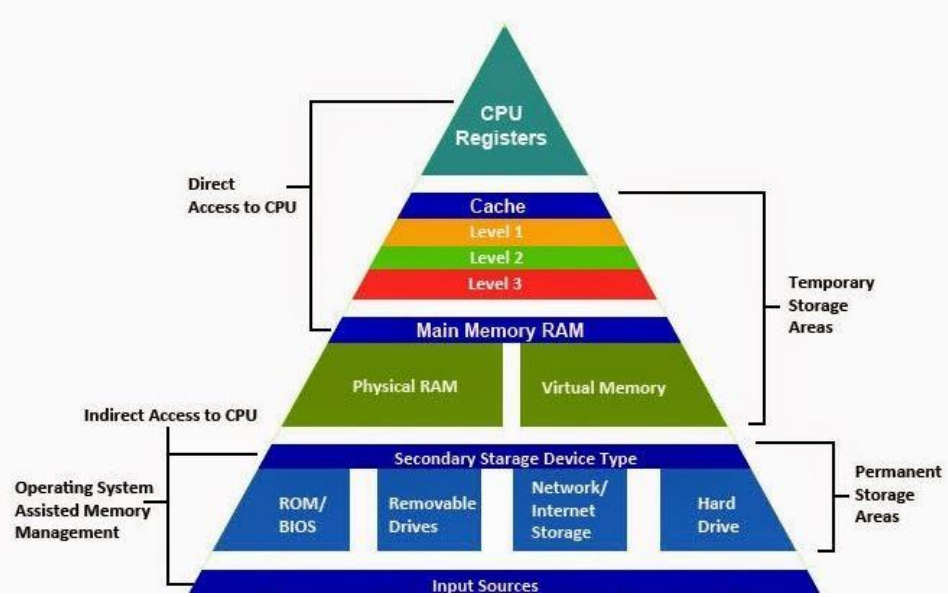


Figure 2.1: Computer memory Hierarchy. The closer memory is to the processor in the computer, the faster, smaller, and more expensive it will be.

It is not feasible to extend the RAM whenever the computer lacks sufficient primary memory for big data tasks.

As mentioned in Chapter 1, in computer programming, when confronted with a big program, such as the gigapixel images sampling, we always seek to use virtual memory as we do not have enough memory to store the whole image.

We are most interested in the virtual memory in Figure 2.1. Virtual memory is a non-system memory, which acts as an extension of main memory (more details will be discussed later in this chapter).

2.2 Software Level

As mentioned in Section 2.1, the RAM is fast but expensive. If we do not have enough primary memory, the programmer has to devise ways to structure the program into pieces, which can be loaded separately in some overlay strategy.[10]

Virtual memory is an essential resource used in computer programming and is typically implemented using a hard drive. The virtual memory technology gives the impression that the program could have a significant, continuous working main memory, when the program may exist, or in fragments, in primary storage, and on disk.

If we do not use virtual memory technique in a large program processing, the program may exist, or in fragments when we try to access more memory than we have (See Figure 2.2).

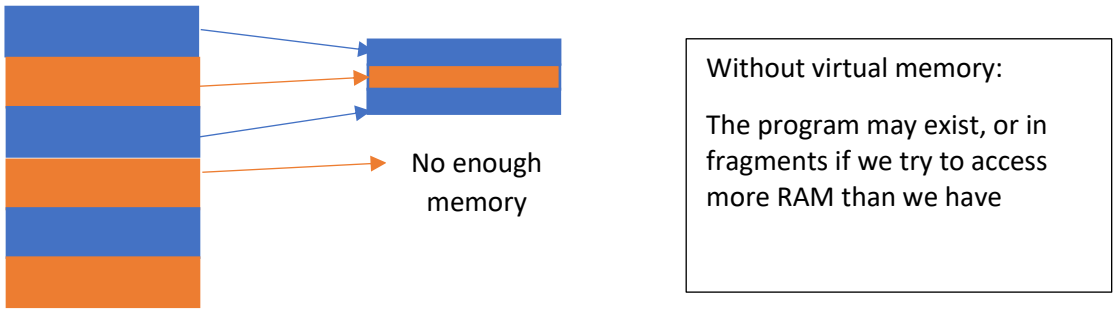


Figure 2.2: Memory Usage without Virtual Memory. A large program without virtual memory where the program address is the same as the RAM Address.

With virtual memory, we can make better use of memory by loading in just a few pieces. Thus, at any one time, only a few parts of the process are in computer memory, and therefore more

procedures can be maintained in memory (See Figure 2.3). Furthermore, time is saved because unused pieces are not swapped in and out of the computer memory.[11]

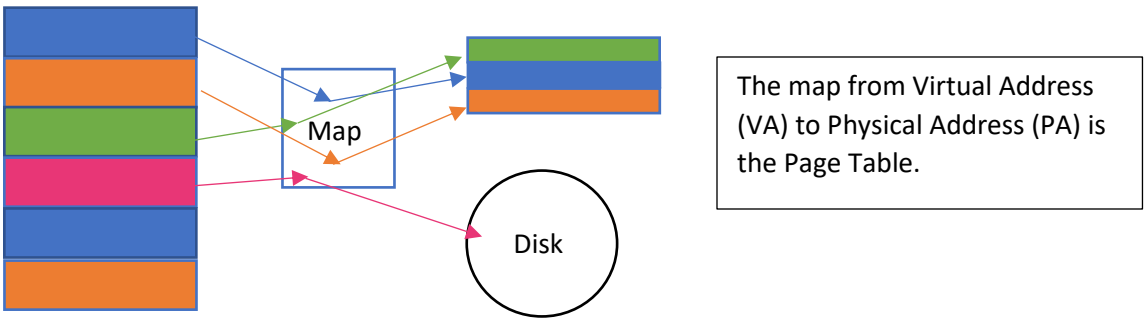


Figure 2.3: Memory Usage with Virtual Memory. A large program with virtual memory where the program address maps to a RAM address.

If the program size is larger than the RAM, when the Operating System brings one piece in, it must throw another out. However, if the OS throws out one page just before it used, then it will have to get that piece again immediately. Too much of this leads to a condition knows as thrashing: the System spends most of its time swapping pieces rather than executing instructions.[12]

If we want to use virtual memory efficiently, we need to have two ingredients. First, there must be hardware support for the paging to be employed. Second, the Operating System must include software for efficiently managing the movement of pages between secondary memory and main memory.

2.3 Existing Sampling Methods

We will consider five existing stochastic sampling methods in this section, which are Poisson Disk Sampling, Jittered Sampling, N-rooks Sampling, Multi-Jittered Sampling, and Gaussian Pyramid Sampling. In the following subsections, each tool is described and then considered regarding its positive and negative impacts on the stochastic sampling process.

2.3.1 Poisson Disk Sampling

Poisson Disk Sampling is a classic stochastic sampling method, which uses sample points that are uniformly distributed throughout the sample space. The Poisson sampling technique can be generalized to a minimum distance Poisson process where the randomly distributed sample points are all separated by at least some minimum distance.[13]

Poisson Disk Sampling has been considered necessary by some, and not efficient by others. The Poisson Disk Sampling can avoid the primary weakness of regular sampling patterns, which is the traditional sampling only sample a fixed set of points so that there are substantial regions that are never tested (See Figure 2.4).

However, Poisson Disk Sampling still should not be considered a baseline of a large datasets sampling algorithm due to its drawback, which is the naive rejection-based approach for generating Poisson disk samples, dart throwing, is impractically inefficient.[14]

Poisson Disk Sampling will not perform very well with large datasets, because the sampling direction in the Poisson Disk Sampling is too random.

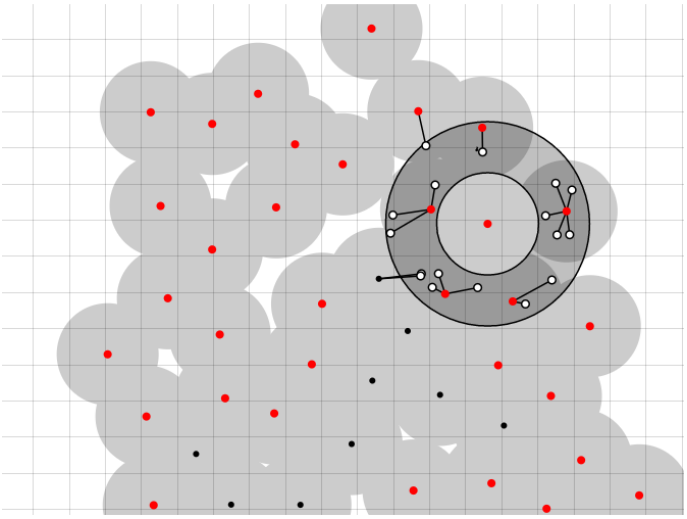


Figure 2.4: Poisson Disk Sampling Pattern. The randomly distributed sample points are all separated by at least some minimum distance. Poisson Disk Sampling will not perform very well with large datasets, because the sampling direction in the Poisson Disk Sampling is too random.

2.3.2 Jittered Sampling

Another important class of stochastic sampling patterns is the jittered sampling pattern. In the Jittered stochastic sampling method, the jittered sampling pattern is well distributed in two dimensions and the pixel is divided into n × n rectangular grid of cells. The algorithm will randomly select only one sample point from each cell (See Figure 2.5).

Jittered sampling patterns perform better than random sampling patterns because they limit the degree of clumping that can occur. However, if the number of samples is not very large and a sampling pattern is "too random," then it can overemphasize some parts of the pixels.[15]

Thrashing will happen if we are trying to use the Jittered Stochastic Sampling method to get sample points in large datasets randomly because the samples must be transferred back and forth in the pixel array.

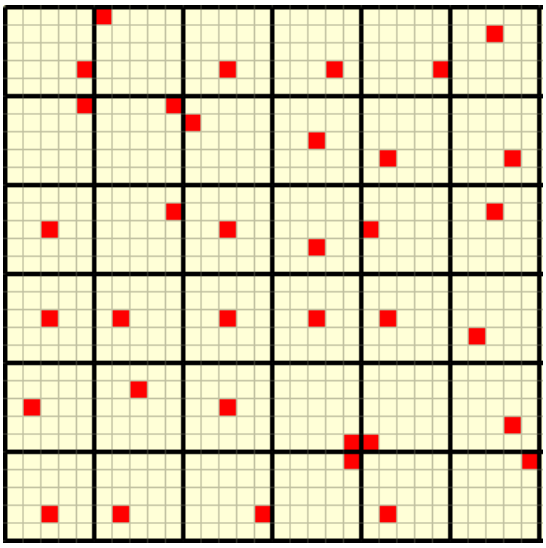


Figure 2.5: The Jittered sampling. The pixel is divided into n × n rectangular grid of cells. The Jittered sampling will randomly select one sample point from each cell. Thrashing will happen if we are trying to use the Jittered Stochastic Sampling method to get sample points in large datasets randomly because the samples must be transferred back and forth in the pixel array.

2.3.3 N-rooks sampling

In 1991, Peter Shirley suggested a new stochastic sampling method which is the N-rooks sampling pattern (See Figure 2.6) that is used to improve the distribution of the Jittered Sampling Pattern. The pixel is divided into an N × N rectangular grid of cells. Then the algorithm will

randomly select N cells with the constraint that no two cells chosen in the same row or column. Then a sample point is randomly chosen from each selected cell.[16]

The N-rooks stochastic sampling method amounts to jittering separately in each dimension and is a proper distribution in one aspect when projected to the X-axis or Y-axis, but not in two sizes combined. Consequently, it works better then Jittered sampling pattern when the number of samples is small, but considerably worse than Jittered sampling when the number of samples is large.[17]

N-rooks stochastic sampling will still not perform very well in the large datasets, because the sampling direction in the N-rooks stochastic sampling is too randomly.

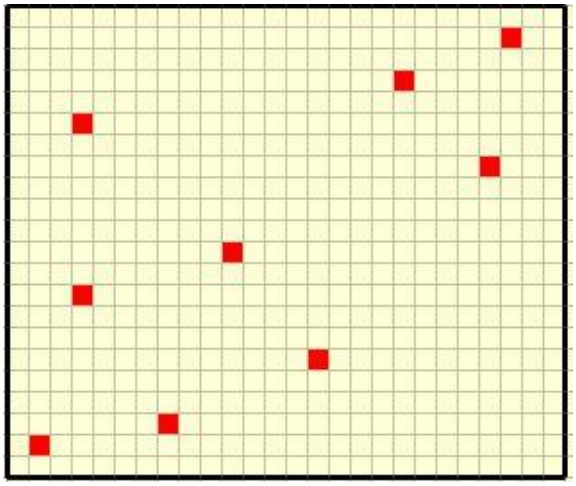


Figure 2.6: The N-rooks sampling for N=9. The pixel is divided into n × n rectangular grid of cells. The N-rooks sampling algorithm will randomly select N cells with the constraint that no two cells chosen in the same row or column. N-rooks stochastic sampling will still not perform very well with large datasets, because the sampling direction in the N-rooks stochastic sampling is too random.

2.3.4 Multi-Jittered Sampling

In 2D Jittered sampling stratifies a set of N samples by dividing the unit square into equal area cells using an m × n grid where N = m × n and m ≈ n and randomly positioning a single sample within each cell (See Figure 2.7) .[18]

This method performs better than either jittering or N-rooks and reduces clumping since samples can only clump near the cell boundaries; no more than four samples can ever clump a given location.

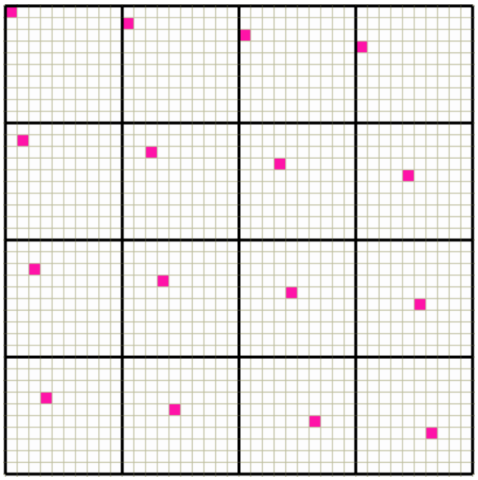


Figure 2.7: Multi-jittered sampling for m=4, n=4, N=16. The pixel is divided into n × n rectangular grid of cells. The Multi-jittered sampling reduces clumping since samples can only clump near the cell boundaries; no more than four samples can ever clump a given location. The thrashing will also happen if we are trying to use the Multi-jittered stochastic sampling method to get sample points in the large datasets randomly.

However, the multi-jittered sampling suffers when these samples are projected onto the X-axis or Y-axis. In this case, we effectively have only m or n strata rather than N. This can significantly increase the variance at edges, especially when they are nearly axis-aligned.

Thrashing also occurs if we are trying to use the multi-jittered stochastic sampling method to get sample points in the large datasets randomly.

2.3.5 Gaussian pyramid Sampling

In the Gaussian Pyramid Sampling, we suppose the image is represented initially by the array g_0 which contains C columns and R rows of pixels. Each pixel represents the light intensity at the corresponding image point by an integer I between 0 and K-1. This image becomes the bottom or zero level of the Gaussian pyramid. Pyramid level 1 contains image g_1, which is a reduced or low-pass filtered version of g_0. Each value within level 1 is computed as a weighted average of values in level 0 within a n × n window (See Figure 2.8).

Each value within level 2, representing g_2, is then obtained from values within level 1 by applying the same pattern of weights. The size of the weighting function is not critical .[19] We have selected the n × n pattern because it provides adequate filtering at low computational cost .[20]

In each level, the Gaussian pyramid Sampling gets sample points also too randomly in each box, which means the thrashing will also happen if we are trying to use the Gaussian pyramid Sampling method to get sample points in the large datasets randomly.

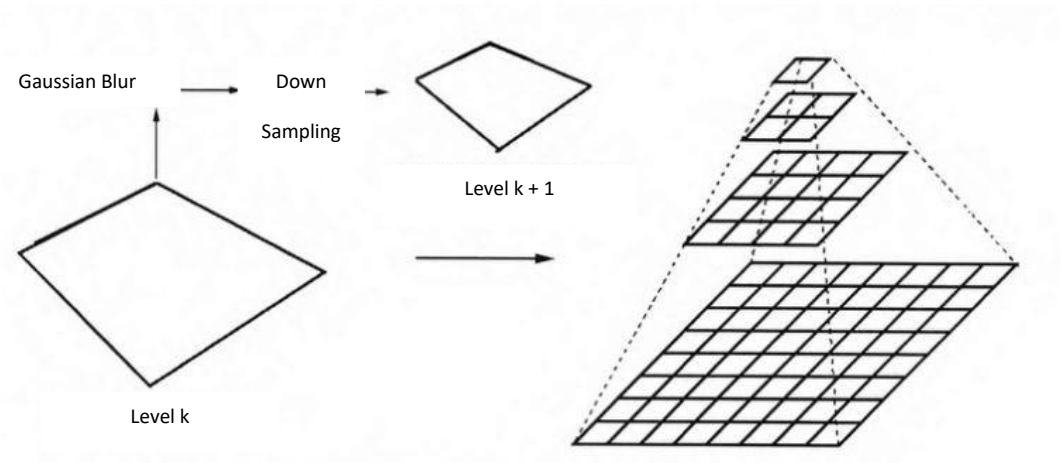


Figure 2.8: The Gaussian pyramid Sampling. The pixel is divided into n × n rectangular grid of cells in each level. The Gaussian pyramid sampling method randomly selects some in each m × m square grid, and calculate the average for these sample points as one single output pixel value. The thrashing will also happen if we are trying to use the Gaussian pyramid Sampling method to get sample points in the large datasets randomly.

2.4 Image Organization

We will consider four existing image file formats in this section, which are JPEG, GIF, PNG, and PPM. Because people also frequently use these four image file formats today, in the following subsections, each form is described and then considered regarding its positive and negative impacts on sampling.

2.4.1 JPEG

JPEG stands for Joint Photographic Experts Group, the name of the committee that created the JPEG standard and also other still pictures coding standards.[21] It is a common file type used on the web, although it is primarily designed for photographs, which means the JPEG file is often used for the processing and storing the full-color images with realistic elements, brightness, and color transitions.

The most prominent advantage of the JPEG file format is that it takes less space in comparison to other file formats, allowing for convenient transmission of compressed images on the Internet. And, the JPEG file format can also produce the smaller files for large photos photographic---like content, or where there are large areas of color gradients. However, the JPEG format does not support transparency and degrades the image quality when compressed.

Hence, the JPEG file format is best for photographs and large, complicated images with lots of gradients where colors fade in from one color to another.

2.4.2 GIF

GIF, also known as Graphics Interchange Format, is a bitmap image format that was developed by a team at the bulletin board service (BBS) provider CompuServe led by American computer scientist Steve Wilhite on June 15, 1987.[22]

GIF supports transparency and animation. Animated GIFs are images in the GIF file format that appear to move or change. Animated GIFs are often applied for banner ads, in place of much more extensive full motion video files, and as accents to homepages. However, the GIF format only supports up to 256 colors, and the pixel in GIF format is either transparent or not. Hence, the GIF images are best for line art with limited colors and images with large flat areas of color.

2.4.3 PNG

The PNG file format, also known as Portable Network Graphics, was created as a free, open-source alternative to GIF.[23] PNG supports full transparency, a full spectrum of colors, and is ultimately lossless, which means PNG does not degrade the image quality when we saved image multiple times. The PNG file format also supports progressive rendering, which means the browser will present a high-quality version of the image to the user, while it loads in additional data for the more beautiful details of the model.

However, the PNG format doesn't support alpha transparency in older browsers without workarounds. Hence, PNG format images are best for images with lots of flat colors and those images that require flat areas of transparency.

2.4.4 PPM

For the linear stochastic sampling algorithm, we will look closely at a very simple, but very widely used image file format. The image format that is used in our experiment is Portable Pixmap (PPM). The PPM, or Portable Pixmap, the form was devised to be an intermediate format for use in developing file format conversion system.[24]

A "magic number" is used to identify the file type (See Figure 2.9). In the PPM format, the magic number is either the ASCII character string "P1", "P2", "P3" or "P4", "P5", "P6" depending upon the storage method used .[25]The lower numbers – "P1", "P2", "P3" – indicate that the image data is stored as ASCII characters; i.e., all numbers are stored as character strings. The format has the advantage that you can read the file in a text editor. The higher numbers – "P4", "P5", "P6" – indicate that image data is stored in a binary encoding – commonly known as Portable Pixmap raw-bits format.[26]

In PPM P6, the data block begins with the first pixel of the top scanline of the image and pixel data is stored in scanline order from left to right in 3-byte chunks giving the R, G, B values for each pixel, encoded as binary numbers.

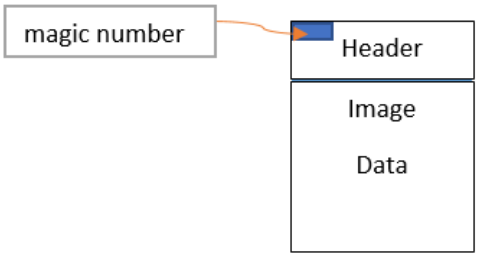


Figure 2.9: The PPM Image File Layout. The magic number indicates the format of the image. Image Data are stored in binary encoding in PPM6.

2.4.5 Structure of Image

In our study, we will only look at "P6" type files, which means the magic number in the experiment is “P6”. The structure of the Image defines a new data type that represents all attributes for an image.

Based on the description of the Image in the previous section, we can define the type of Image that can hold width, height, magic number and Pixel (Figure 2.10).

```
typedef struct Image {
        int width;
        int height;
        int magic_number;
        u_char *pixel;
} Image;
```

Figure 2.10: The structure of the Image in the Linear Stochastic Sampling. The data in the image are stored in a one-dimensional pixel array.

`u_char` stands for the unsigned character. One way of accessing a contiguous chunk of memory, instead of with an array, is with a pointer. Since we already know about strings which are made up of characters, strings are just arrays; we can access each character in the array using subscript notation.

Chapter 3

Linear Stochastic Sampling

We will now define and analyze the Linear Stochastic Sampling method in this chapter. We already know that a simple jittered stratified a set of N samples by dividing the unit square into equal area cells and posting a single sample point with each cell.[27] Whereas, the sample space in the Linear Stochastic Sampling method is still subdivided into N regions and each sample pixel needs to satisfy a random minimum distance constraint.

As we know, the pixels are viewed in a one-dimensional array in the main memory of the computer. If we want to avoid the thrashing in the large datasets Sampling, the index of currently selected sample point should be higher on the pixel array than the previous one, which means the index of sample points in the pixel array is arranged in ascending order (See Figure 3.1).

And at the same time, the distance or gap between the currently selected pixel point and the previously selected pixel point should be a random positive number within a given range which is decided by the investigator. So, we can guarantee that each sample in the population has an equal chance to be selected (See Figure 3.1).

And if the index of the last observation point that is selected by the algorithm is over the pixel array size, the algorithm will stop the Linear Stochastic Sampling process.

We already know that the sampling direction in the new technique is arranged in ascending order and each sample point has an equal chance to be selected in the population. That is why we call the new algorithm is the Linear Stochastic Sampling method.

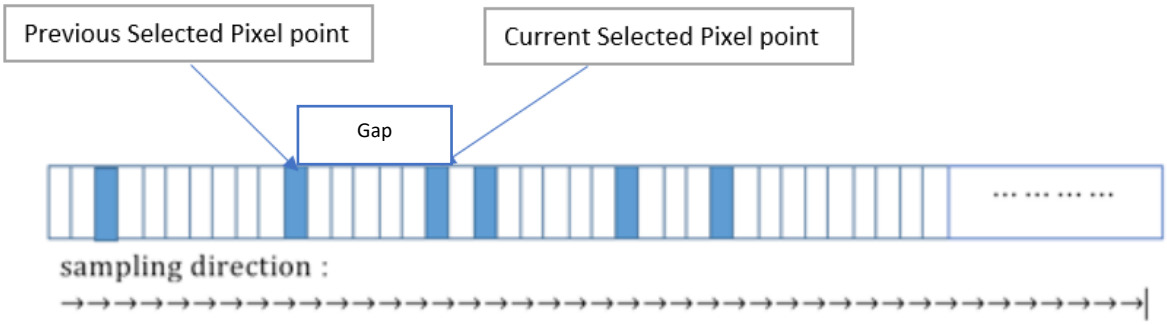


Figure 3.1: Sampling Direction in the Linear Stochastic Sampling. The index of currently selected sample point should be higher on the pixel array than the previous one; the gap is a random positive number within a given range.

Specifically, in Figure 3.1, the first selected pixel point also has some distance between the first pixel position in the pixel array (See Figure 3.2). The length of the gap at here is also a random number within the given range. If gap = 0, then the first selected Pixel Point and the first Pixel Point will be the same point.

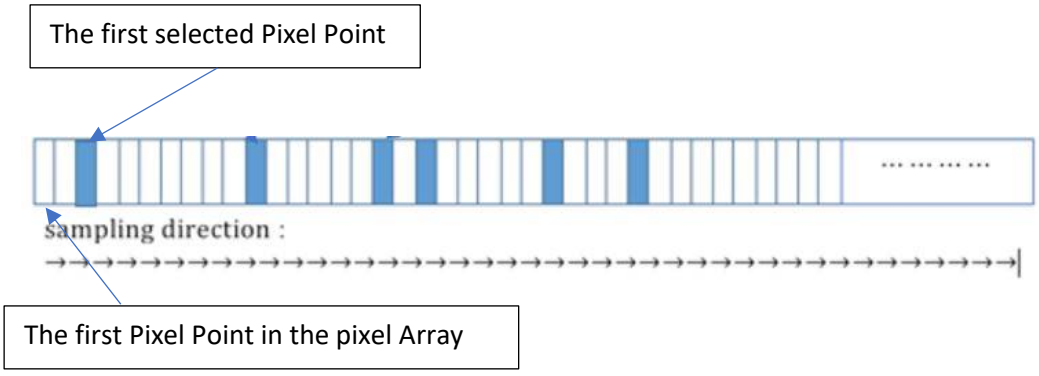


Figure 3.2: Termination of Linear Stochastic Sampling. The length between the first selected Pixel Point and the first Pixel Point in the pixel Array is a random number. If length =0, then the first pixel is chosen at the point, and the first Pixel Point will be the same point.

3.1 Basic principle

As we mentioned in Chapter 2, the Linear stochastic sampling algorithm is based on jittered sampling and the Gaussian pyramid Sampling, but the sampling speed is much faster. At the same time, by using the linear stochastic sampling method, the RAM in the computer will be used more efficiently.

The purpose of this chapter is to introduce the basic concepts of the Linear stochastic sampling algorithm.

The *index* denotes the position of the selected pixel in the pixel array, which is a one-dimensional array (See Figure 3.1).

The *width* denotes the width of the jittered sampling or the original image in the Linear Stochastic Sampling algorithm (See Figure 3.3).

By using the indexes of the selected samples from the pixel array, we can use the functions listed below to find the position for each selected pixel in two- dimensional pattern (See Figure 3.2).

$$row = index \div width$$

$$column = index \,\%\, width$$

The linear stochastic sampling pattern is well distributed in one dimension (See Figure 3.1). Each pixel in the image can only be selected once, and each of these pixels has an equal chance to be selected.

If the *range*, which is the random distance between the currently selected sample and the previous one is performed not very well, the clumping can still be present in two-dimensional array

view. However, the massive clumping in any specific place is a small probability event in the linear stochastic sampling algorithm (See Figure 3.3).

The *range* of the random difference number between the currently selected sample and the previous is one of three essential measures of the quality of the linear sampling process. The other two standards of the quality of the Linear Stochastic Sampling Algorithm are the *scale* and the *average*.

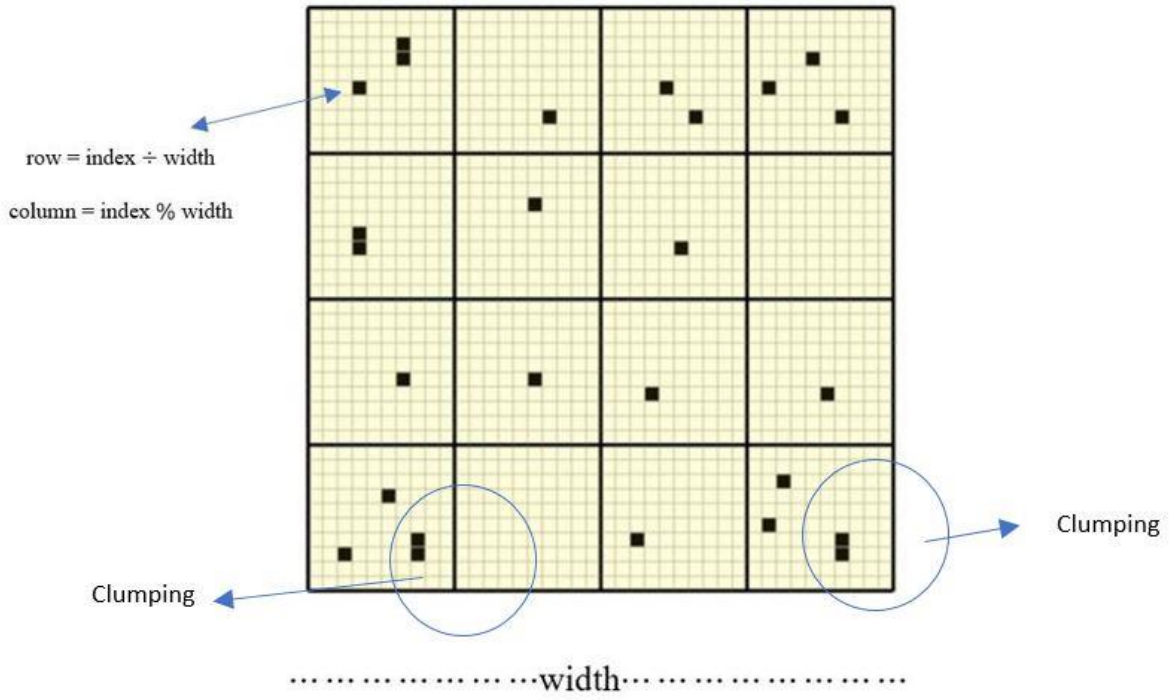


Figure 3.3: 2D Pattern in the Linear Stochastic Sampling Algorithm (LSS). The Linear Stochastic Sampling Algorithm randomly chooses samples from the one-dimensional pixel array and collect the index of these samples. Then Linear Stochastic Sampling Algorithm (LSS) converts these selected samples from one one-dimensional array to the two-dimensional pattern.

3.2 Variables

We mentioned at the end of section 3.1 that there are three critical variables in the Linear Stochastic Sampling Algorithm, which are *Span*, *Scale,* and the *Sample Average*. The type of the *Sample Average* is *double*, but both of *Span* and *Scale* are *int.*

3.2.1 Scale

The ***Scale*** is a positive integer that presents the quantitative relationship between the input file size and the output file size.

For instance, consider that the input file size is 40×40, and the *scale* is 10. The original image will be subdivided into 4×4 regions, which are 16 regions; each area is a 10×10 rectangular grid of cells. The output image also has 16 compartments that each cell is a 1×1 square grid of cells (See Figure 3.4).

The *height* and *width* denote the height and width of the original image.

The *output _height* and *output_width* denote the height and width of the output image.

So, we have:

$$output\_height = height \div ratio$$

$$output\_width = width \div ratio$$

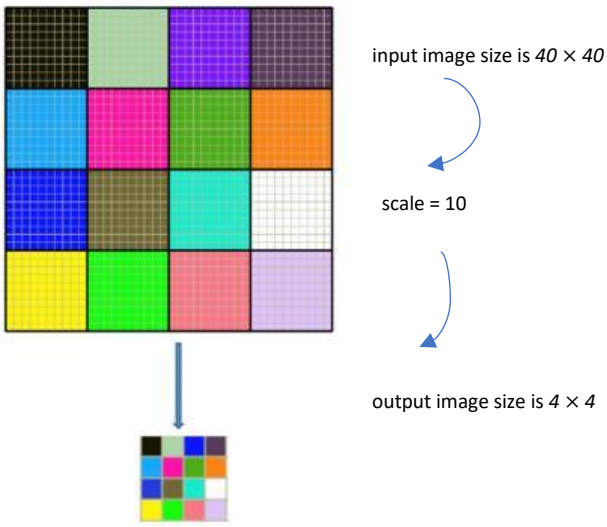


Figure 3.4: The Variable *Scale* in the Linear Stochastic Sampling Algorithm. *Scale* represents the quantitative relationship between the input image size and the output image size.

3.2.2 Span

The ***Span*** is a positive integer, which presents a range where the distance between the currently selected sample point and the previous one must be located in this range.

For example, if the value of the *span* is *20*, then the distance between the current sampled pixel and the previous one must be a random integer number from *1* to *20*. If the currently selected sample's index is *x* in the pixel array, then the index of next selected sample point must be between *x+1* and *x+20* (See Figure 3.5).

Therefore, the Linear Stochastic Sampling technique is generalized to a minimum sampling process where the randomly distributed sample points are all separated by at a random distance. It means each sample point in the population has an equal chance being selected.

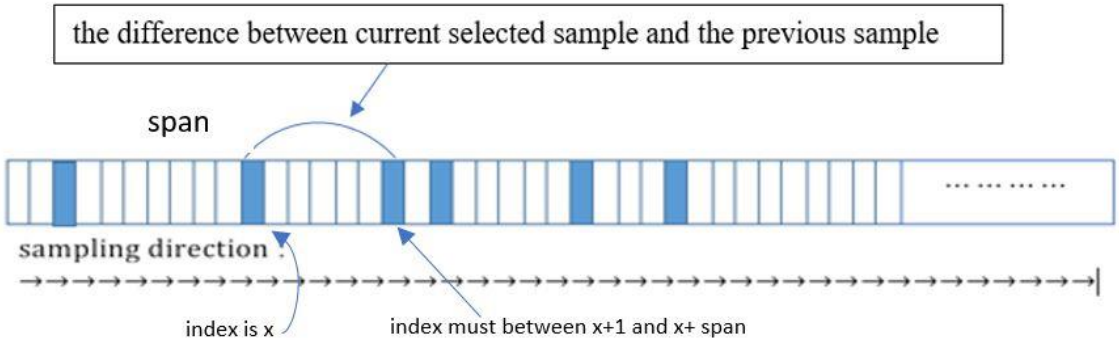


Figure 3.5: The Variable *Span* in the Linear Stochastic Sampling Algorithm (LSS). *Span* is the distance between the currently selected sample and the previous one. If the currently selected sample's index is *x* in the pixel array, then the index of next selected sample point must be between *x+1* and *x+20*

Also, by controlling the value of the *span*, we can avoid the clumping as much as possible in Figure 3.3. However, the tricky part is determining an appropriate span value.

3.2.3 Average

The *average* is a positive float number that presents the average pixel value in each cell. The average sampled pixel value in each region will be a new single pixel in the output. Sometimes, the algorithm may not select any pixel in one cell, so the algorithm will choose the first pixel in this cell as the average pixel value that will be written into the output image (See Figure 3.6).

If the value for these n selected pixels are (r_1, g_1,b_1), (r_2, g_2,b_2), ……, and (r_n, g_n,b_n), then the average pixel value will be ($\frac{r_1+r_2+\cdots+r_n}{n}$),($\frac{g_1+g_2+\cdots+g_n}{n}$),($\frac{b_1+b_2+\cdots+b_n}{n}$), which is ($r_{avg}$, g_{avg}, b_{avg}).

For example, three pixels are selected in the first cell in Figure 3.5. If the value for these three-selected pixels are (*186*, *204*,*121*), (*159*, *198*,*234*) and (*147*,*57*,*245*), then the average pixel value will be ($\frac{186+159+147}{3}$,$\frac{204+198+57}{3}$,$\frac{121+234+245}{3}$), which is (*164*,*153*,*200*).

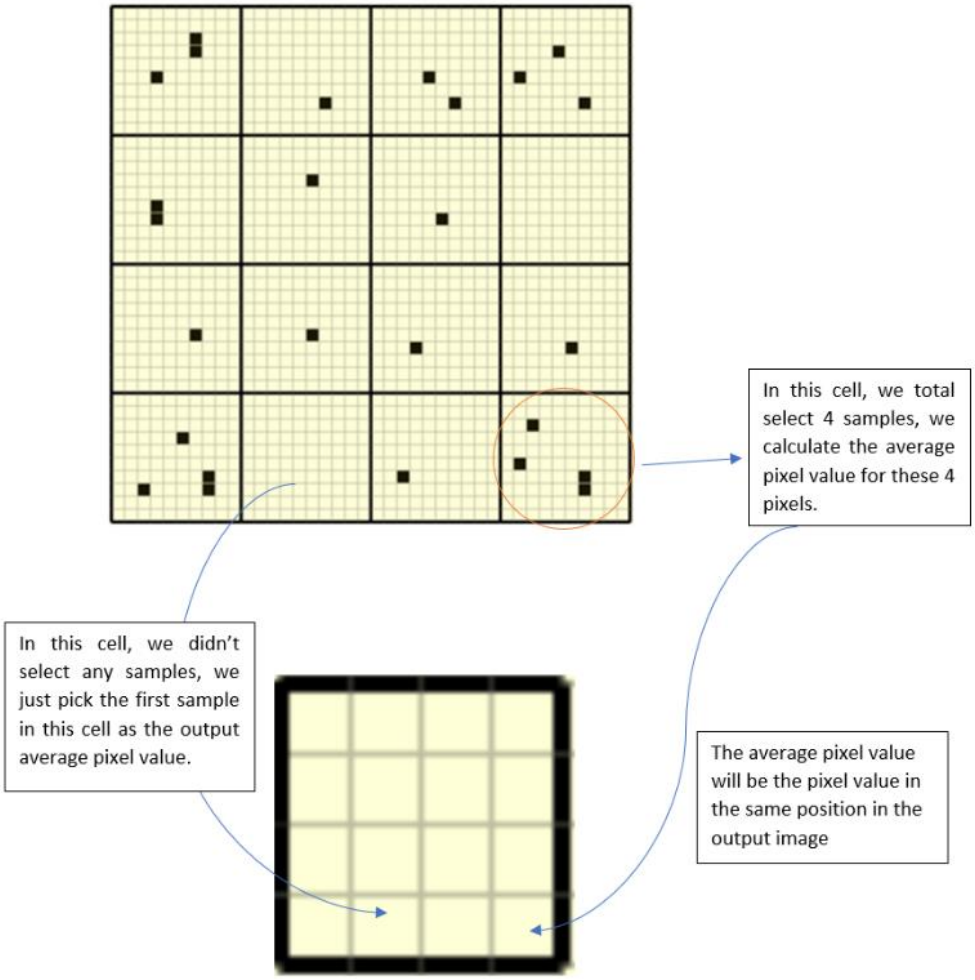


Figure 3.6: The Variable *Sample Average* in the Linear Stochastic Sampling Algorithm (LSS). If the value for these 3 selected pixels are (*186*, *204*,*121*), (*159*, *198*,*234*) and (*147*,*57*,*245*), then the average pixel value will be ($\frac{186+159+147}{3}$,$\frac{204+198+57}{3}$,$\frac{121+234+245}{3}$), which is (*164*,*153*,*200*).

3.3 Structure

The Linear Stochastic Sampling algorithm has three significant structures in the implementation, which are Image, Buffer, and Cell.

The diagram given in Figure 3.7 shows the primary structures of the Linear Stochastic Sampling Algorithm and their fundamental relationships.

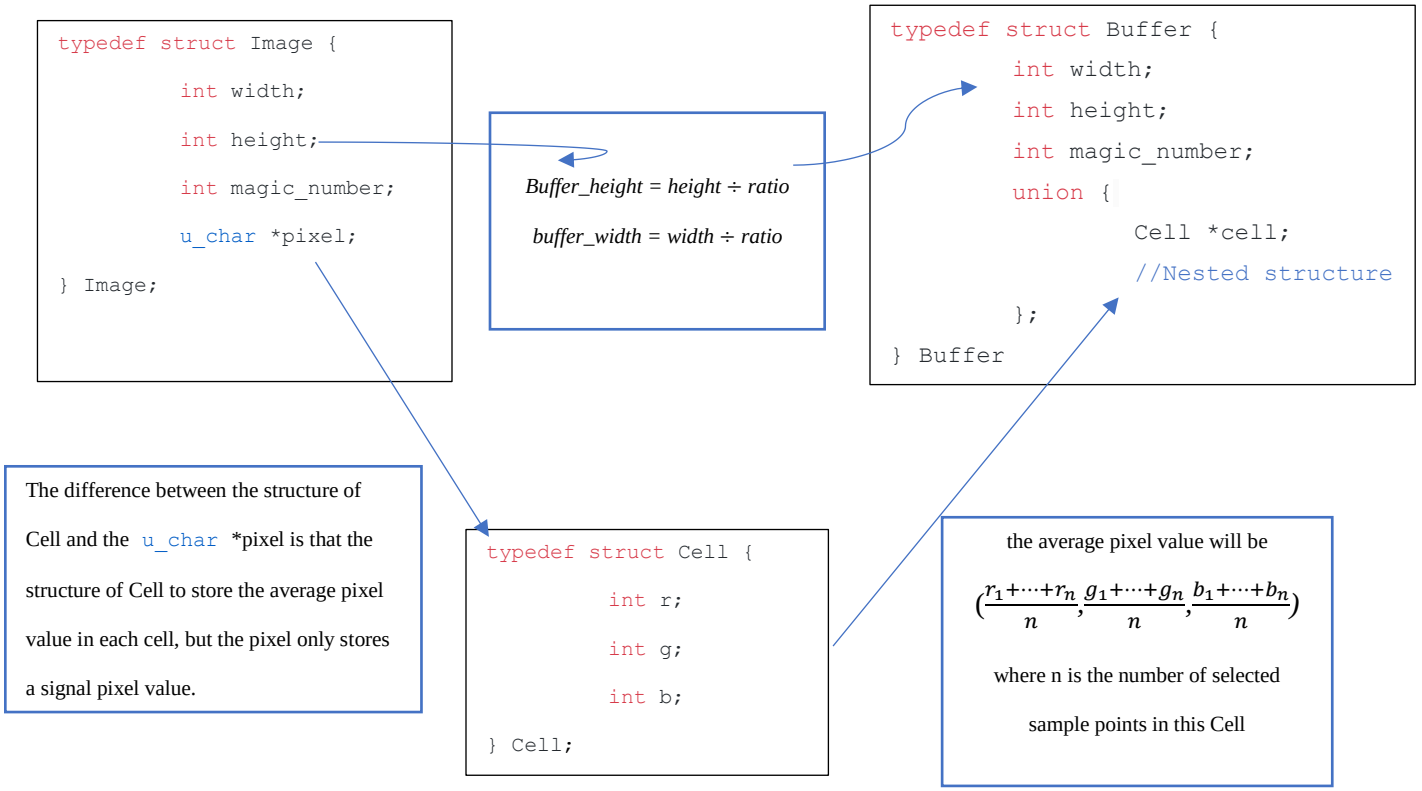


Figure 3.7: Structures in the Linear Stochastic Sampling Algorithm (LSS). This Figure shows all the primary structures of the Linear Stochastic Sampling Algorithm (LSS) and their fundamental relationships.

The only difference between the structure of Buffer and the structure of Image is that the Buffer use the structure Cell to store the pixel value. The purpose of structure Cell is to save the average pixel value in each cell.

Pixel is the smallest element of an image. Each pixel corresponds to any one value. In an 8-bit grayscale image, the value of the pixel between *0* and *255*. The amount of a pixel at any point corresponding to the intensity of the light photons striking at that end. Each pixel stores an amount proportional to the light intensity at that particular location.

A color perceived by the human eye can be defined by a linear combination of the three primary colors red, green and blue. These three colors form the basis for the RGB-color space.

Hence, each visible color can be defined by a vector in the three-dimensional color space. The intensity is given by the length of the vector, and the actual brightness by the two angles describing the orientation of the vector in the color space.[28]

3.4 Algorithm

The process of Linear Stochastic Sampling Algorithm (LSS) is divided into four steps and an appendix as follows:

- **Step 1:** The Linear Stochastic Sampling algorithm (LSS) randomly chooses some samples from the pixel array (a single-dimensional array) in ascending order and collect the index for these selected samples (See Figure 3.8).

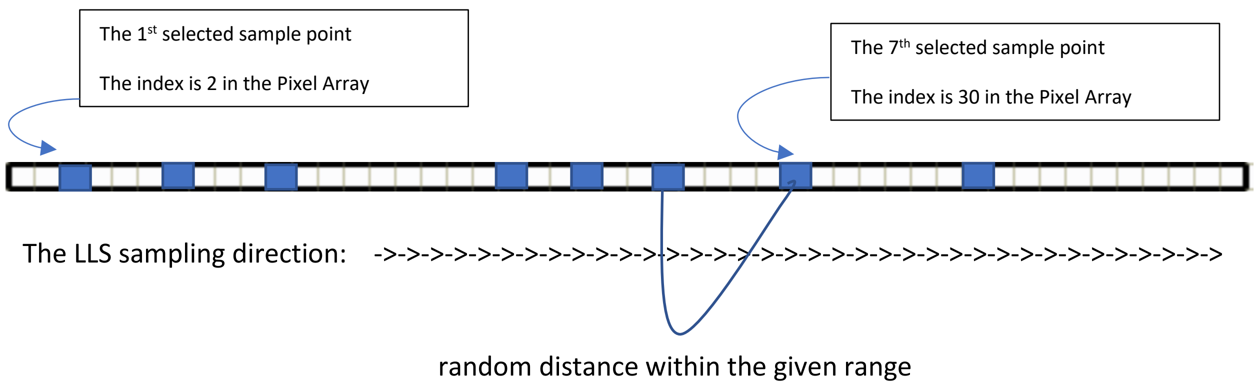


Figure 3.8: Step 1 of the Linear Stochastic Sampling Algorithm (LSS). The Linear Stochastic Sampling algorithm (LLS) randomly chooses samples from the pixel array (a single-dimensional array) in ascending order and collect the index for these selected samples.

- **Step 2:** By using the index of each selected sample in the pixel array that is already collected on step 1, the Linear Stochastic Sampling algorithm (LSS) find the position for each selected sample point in the 2D pattern. Then, LSS converts the selected pixel from one-dimensional view (pixel array) to a two-dimensional model (See Figure 3.9).

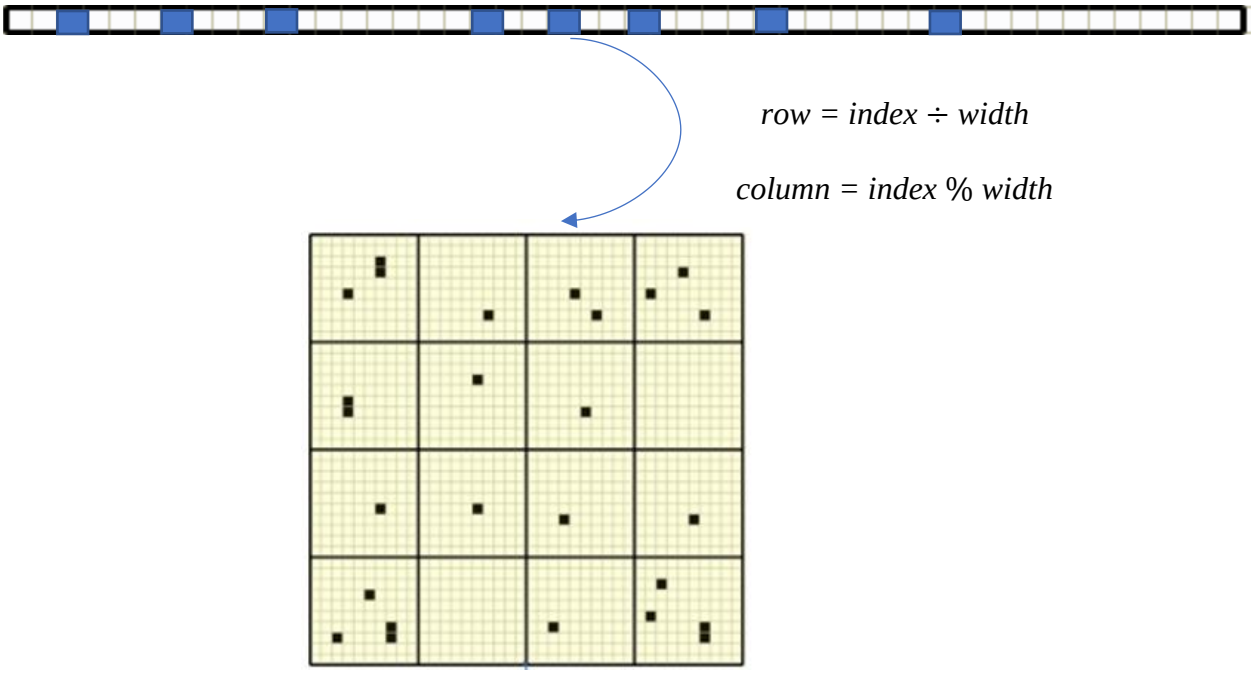


Figure 3.9: Step 2 of the Linear Stochastic Sampling Algorithm (LSS). The Linear Stochastic Sampling algorithm (LSS) converts the selected pixel from one-dimensional view (pixel array) to a two-dimensional model by using the index that is already collected on step 1.

- **Step 3:** Based on the original Image and scale, the Linear Stochastic Sampling algorithm (LSS) create a buffer and calculate the sample average pixel value in each cell on the two-dimensional pattern. If the algorithm does not select any pixel in one cell, the algorithm will choose the first pixel in this cell as the average pixel value that will be written into the output image (See Figure 3.10).

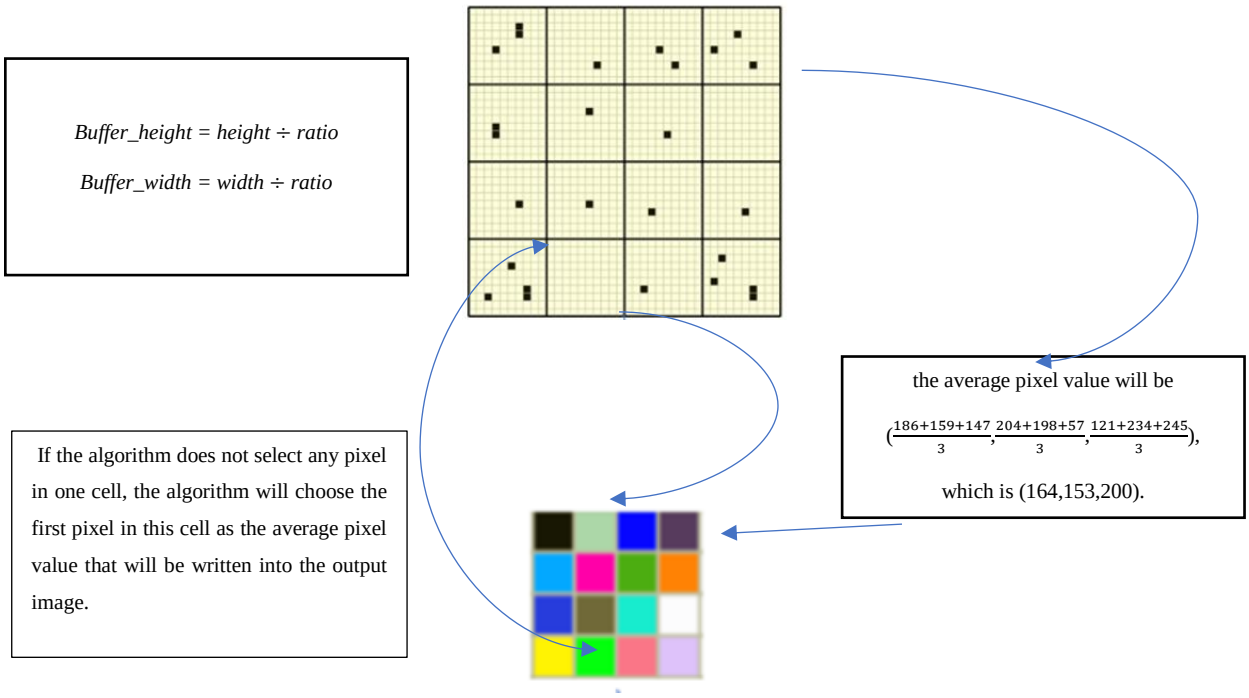


Figure 3.10: Step 3 of the Linear Stochastic Sampling Algorithm (LSS). If Linear Stochastic Sampling algorithm (LSS)does not select any pixel in one cell, the algorithm will choose the first pixel in this cell as the average pixel value that will be written into the output image.

- **Step 4:** The Linear Stochastic Sampling algorithm (LSS) generate the output file based on the buffer, which is created on step3(See Figure 3.11).

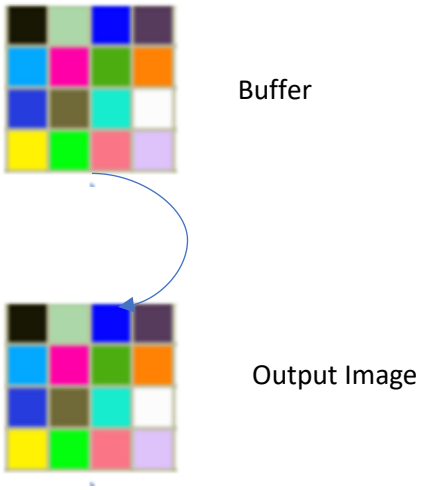


Figure 3.11: Step 4 of the Linear Stochastic Sampling Algorithm (LSS). The Linear Stochastic Sampling algorithm (LSS) generate the output file based on the buffer.

The workflow diagram given in Figure 3.12 shows the significant levels of the process and their essential relationships.

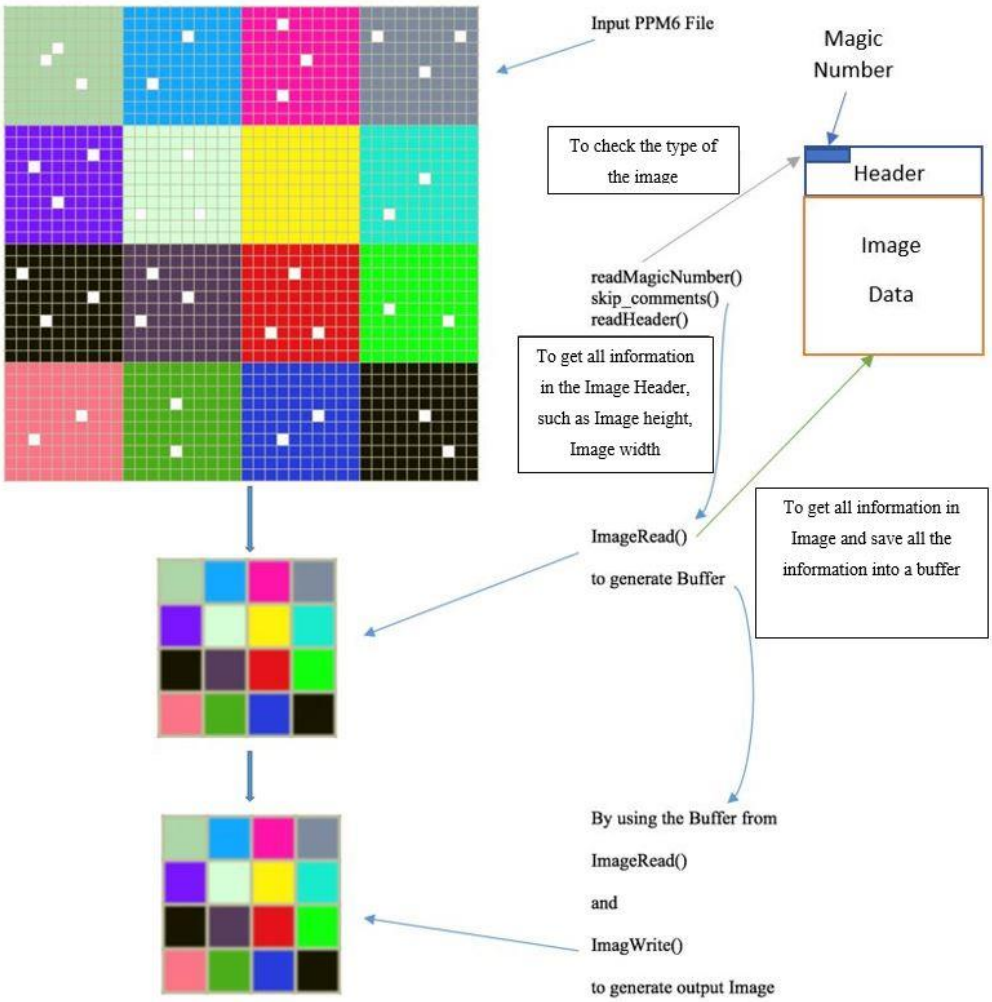


Figure 3.12: Workflow of the Linear Stochastic Sampling Algorithm

Chapter 4

Experiment

In this Chapter, we switch from Algorithm, the focus of Chapter 3, "Processing," to Statistics. As we mentioned before, there are three critical variables in the Linear Stochastic Sampling algorithm, which are *Span*, *Scale*, and *Average*. The purpose of this chapter is to design an experiment to find the relationship between these three variables.

In the first section of this chapter, we proof the precondition of the Linear Stochastic Sampling algorithm based on the code of the implementation, which is the sample that is selected by the algorithm is a simple random sample. (See Section 4.1 Simple random sample).

Then, in the second section, we overview the experiment goal that our setup was designed to address. We are conducting a multiple-variable experiment setup for the collection of multiple-variable interaction data for the linear stochastic sampling algorithm. This step up was developed and used to collect experimental data as part of our project aimed at finding the equation for the Ratio, Span and the Average. (See Section 4.2 Design an experiment).

The main aim of the linear stochastic algorithm is to find the relationship between the *Span*, *Scale* and the *Average*. Finally, in the last section, we will derive an expression, which represents the relationship of the three critical variables in the Linear Stochastic Sampling algorithm (See Section 4.3 Formula).

4.1 Simple Random sample

We know that a simple random sample is a subset of individuals (a sample) chosen from a more extensive set (a population). Each sample point is chosen randomly and entirely by chance, such that each sample point has the same probability of being selected at any stage during the sampling process, and each subset of *k* individuals has the equal chance of being chosen for the sample like any other subset of *k* individuals.[29]

The code given in below shows the pseudocode of the sampling process in the Linear Stochastic Sampling algorithm:

```
Buffer *ImageRead(const char *filename,int Ratio,int max){
        time_t t;
        srand((unsigned)time(&t));
        int span = rand()%max;
        int index = span;

                ⋮
                ⋮

        while (index < size) {
                int x = ((index%source_width)/(Ratio));
                int y = ((index/source_width)/(Ratio));
                fread(&c, 1, 1, f_source);
                buffer->box[y*buffer->width+x].r += c;
                fread(&c, 1, 1, f_source);
                buffer->box[y*buffer->width+x].g += c;
                fread(&c, 1, 1, f_source);
                buffer->box[y*buffer->width+x].b += c;
                buffer->box[y*buffer->width+x].count++;
                span = rand()%max;
                fseek(f_source,3*span, SEEK_CUR);
                index += 1+span;
        }


                ⋮
                ⋮
}
```

Figure 4.1: Pseudocode for the Linear Stochastic SamplingAlgorithm

The above algorithm enjoys all the advantage in flexibility. It requires only linear access, does not require writable input .[30]

And, the pseudocode shows us some preconditions of the Simple random sample:

1. It is random access

   The code --- `srand((unsigned)time(&t))` --- generates a random positive integer number every time.

2. The population size, which is the test image size is known.

3. The code --- `span = rand()%max` --- tells us that each sample point has the same probability of being selected at any stage during the sampling process.

4.2 Experiment Design

We will describe some of the details of the Linear Stochastic Sampling experiment. The experiment setup is shown schematically in Figure 4.2.

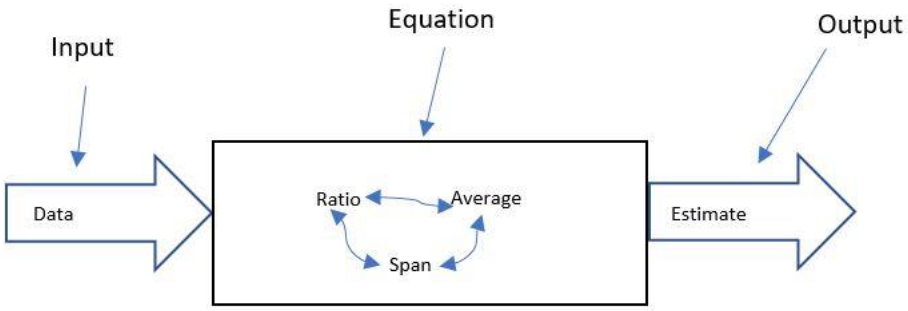
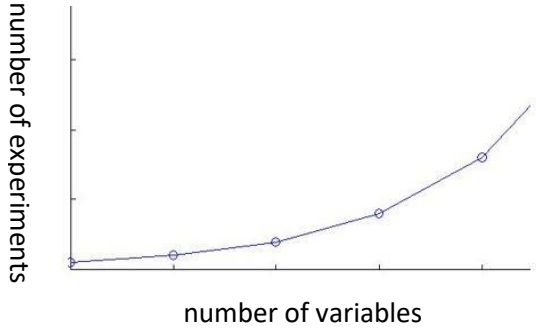


Figure 4.2: Experiments. This Figure shows how to set up an Experiment in the study to find an equation to presents the relationship between the *span* (*ratio*), *Sample Average*(*Average*), and *Span*.

An investigator who plans to conduct operations with multiple independent variables must decide whether to use a complete or reduced factorial design. We can't observe every member of the population, which would be prohibitively expensive, hence, in our tests, we will use minimized factorial design in our experiment (See Figure 4.3).

The experiment should give us data on how the potential variables interact with the Linear Stochastic Sampling algorithm (LSS). But we also need the information on the multivariable interaction strategies that the linear stochastic algorithm should employ to achieve the desired attributes, flexibility, and stability.

Our goal is to gather interactions where the equation can combine *Scale*, *Average*, and *Span* results of a multiple-variable experiment, and the user can control the variable. Once data are collected from an experimental study, it must be put into a form—usually numerical—to be analyzed.



The relationship between the number of variables and the required number of experiments

Figure 4.3: Experiment Convergence. The relationship between the number of variables and the number of experiments

Steps to designing the Linear Stochastic Sampling experiment:

1. Identify the variables of interest which are the three independent variables (*Scale*, *Average*, and *Span*) and the population which is all the pixels in the test image.
2. Develop a method for collecting the data (see Section 4.3).
3. Collect the data.
4. Analysis the data using descriptive statistics.
5. Use inferential statistics based on the data to conduct the expression (see Figure 4.4).
6. Validation

The step of the Linear Stochastic Sampling (LSS) experiment is shown schematically in Figure 4.4.

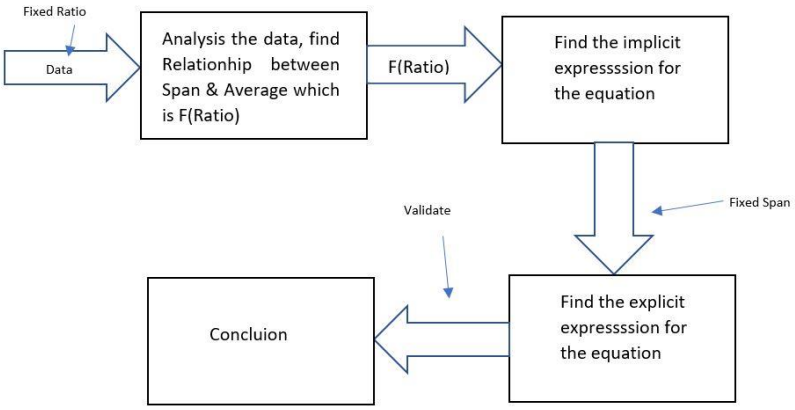


Figure 4.4: Experiment Design. This Figure shows the how to design the Linear Stochastic Sampling experiment.

4.3 Formula

As we mentioned in Chapter 3, we have three critical variables in the Linear Stochastic Sampling algorithm, which are *Scale(Ratio)*, *Span*, and *Sample Average*.

And one of the main aims of the Linear Stochastic Sampling algorithm is to find the relationship between these three critical variables.

In this section, we are going to use elimination method to eliminate one of the variables and see the relationship between the other two variables.

4.3.1 Equation

In the first step in the experiment, a small test group was formed by a fixed value of the *Scale*, in particular, *Scale =18*, and we will try to find the relationship between the *Average* and *Span*. The data pairs are formed from a tuple of the *average* and *span*.

When we collect *100* data pairs and find the equation between span and average is:

$$Average = 467Span^{-0.922}$$

When we collect *200* data pairs and find the equation between span and average is:

$$Average = 506.7Span^{-0.949}$$

When we collect *500* data pairs and find the equation between span and average is:

$$Average = 551.49Span^{-0.971}$$

When we collect *1000* data pairs and find the equation between span and average is:

$$Average = 579.13Span^{-0.982}$$

When we collect *5000* data pairs and find the equation between span and average is:

$$Average = 620.35Span^{-0.994}$$

⋮

⋮

⋮

When we collect *10000* data pairs and find the equation between span and average is:

$$Average = 630.28Span^{-0.997}$$

From these experiments, we find that both the value of the coefficient and the value of the power of the span in the equation is associated with the amount of the paired data.

We will focus on the power of the span in this section. Because *Span* is one of the critical variables in the Linear Stochastic Sampling algorithm, and if the value of the power is changed, the type of the equation will be different.

The data given in Tables *1* shows when the *scale = 18* in the test, the relationship between the value of the power of the *Span* in each equation and the number of the collected data n.

n	100	200	300	400	500	600	700	800	900	1000
power	-0.922	-0.949	-0.96	-0.967	-0.971	-0.975	-0.977	-0.979	-0.981	-0.982
n	1100	1200	1300	1400	1500	1600	1700	1800	1900	2000
power	-0.983	-0.984	-0.985	-0.986	-0.986	-0.987	-0.988	-0.988	-0.989	-0.989
n	2100	2200	2300	2400	2500	2600	2700	2800	2900	3000
power	-0.989	-0.99	-0.99	-0.99	-0.991	-0.991	-0.991	-0.991	-0.992	-0.992
n	3100	3200	3300	3400	3500	3600	3700	3800	3900	4000
power	-0.992	-0.992	-0.992	-0.992	-0.993	-0.993	-0.993	-0.993	-0.993	-0.993
n	4100	4200	4300	4400	4500	4600	4700	4800	4900	5000
power	-0.993	-0.994	-0.994	-0.994	-0.994	-0.994	-0.994	-0.994	-0.994	-0.994
n	5100	5200	5300	5400	5500	5600	5700	5800	5900	6000
power	-0.994	-0.994	-0.995	-0.9955	-0.995	-0.995	-0.995	-0.995	-0.9955	-0.995
n	6100	6200	6300	6400	6500	6600	6700	6800	6900	7000
power	-0.9955	-0.9955	-0.995	-0.995	-0.995	-0.995	-0.995	-0.995	-0.996	-0.996
n	7100	7200	7300	7400	7500	7600	7700	7800	7900	8000
power	-0.996	-0.996	-0.996	-0.996	-0.996	-0.996	-0.996	-0.996	-0.996	-0.996
n	8100	8200	8300	8400	8500	8600	8700	8800	8900	9000
power	-0.996	-0.996	-0.996	-0.996	-0.996	-0.996	-0.996	-0.996	-0.996	-0.996

Table 1: Using convergence to find the Span Power. This Figure shows the relationship between the value of Span Power and the Number of Experiment Trials.

Based on the data that we gathered in the previous, we can describe the relationship between the amount of received data and the power of the *Span* in Figure 4.5.

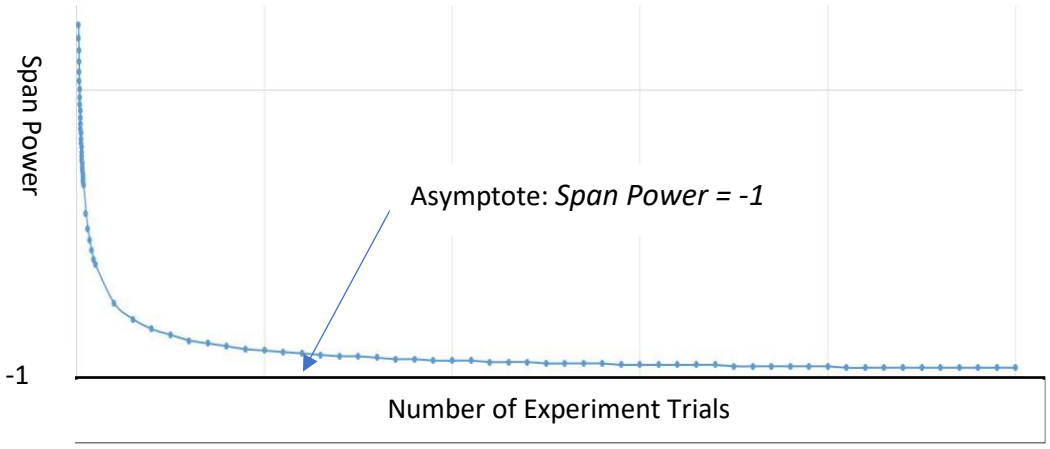


Figure 4.5: Span Power VS Number of Experiment Trials. This Figure shows the relationship between the value of Span Power and the Number of Experiment Trials.

From the Figure 4.5, we find that when the number of the data pairs goes to infinity, the value of the Span Power approaches to a constant value, which is -1. Hence, we can make an ***assumption***: the power of the Span in the research is -1.

Based on the assumption, we can find the implicit expression of the formula, which is:

$$Average = Coefficient \times Span^{-1}$$

To be more generic, we can write the equation as:

$$S_{AVG} = \frac{k_1}{Span + k_2}$$

S_{AVG} : The Sample Average

K_1: Constant

K_2: Constant

4.3.2 Coefficient

In the previous subsection, we have found the implicit expression of the formula, which is:

$$S_{AVG} = \frac{k_1}{Span+k_2}$$

S_{AVG} : *The Sample Average*

K_1: *Constant*

K_2: *Constant*

In this section, a small test group was formed by the number of paired data, in particular, we only collect 100 matched data in each experiment, which means the span from 1 to 100, and try to find the value of k_1.

When the *Scale* is 1, the expression is:

$$Average = 1.4339Span^{-0.922}$$

When the *Scale* is 2, the expression is:

$$Average = 5.7597Span^{-0.922}$$

When the *Scale* is 3, the expression is:

$$Average = 12.96Span^{-0.922}$$

When the *Scale* is 4, the expression is:

$$Average = 23.08Span^{-0.922}$$

When the *Scale* is 5, the expression is:

$$Average = 35.977Span^{-0.922}$$

When the *Scale* is 6, the expression is:

$$Average = 51.849Span^{-0.922}$$

When the *Scale* is 7, the expression is:

$$Average = 70.577Span^{-0.922}$$

When the *Scale* is 8, the expression is:

$$Average = 92.155Span^{-0.922}$$

When the *Scale* is 9, the expression is:

$$Average = 116.685Span^{-0.922}$$

When the *Scale* is 10, the expression is:

$$Average = 143.985Span^{-0.922}$$

From these ten experiments, we find that if the number of paired data is fixed, the exponent of the *Span* will be the same, and the coefficient of the formula is associated with the value of the *Scale*

The data give in Table 2 shows the relationship between the *Scale* and the factor in each test.

Scale	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
k_1	*1.4339*	*5.7597*	*12.96*	*23.038*	*35.977*	*51.849*	*70.577*	*92.155*	*116.68*	*143.98*

Table 2: The relationship between the value of the Scale and the value of the k_1

Based on the information in Table 2, we can find that:

$$\frac{c_2}{c_1} = \frac{5.7597}{1.4339} = 4.0168073 \approx 4 = \frac{2^2}{1^2} = \frac{(s_2)^2}{(s_1)^2}$$

$$\frac{c_3}{c_1} = \frac{12.96}{1.4339} = 9.0382871 \approx 9 = \frac{3^2}{1^2} = \frac{(s_3)^2}{(s_1)^2}$$

$$\frac{c_4}{c_1} = \frac{23.038}{1.4339} = 16.066671 \approx 16 = \frac{4^2}{1^2} = \frac{(s_4)^2}{(s_1)^2}$$

$$\vdots$$

$$\vdots$$

$$\vdots$$

$$\frac{c_{10}}{c_1} = \frac{143.98}{1.4339} = 100.41146 \approx 100 = \frac{10^2}{1^2} = \frac{(s_{10})^2}{(s_1)^2}$$

$$\vdots$$

$$\frac{c_n}{c_1} = \frac{n^2}{1^2} = \frac{(n)^2}{(s_1)^2}$$

Now, we could guess that:

$$c_n = c_1 \times (\frac{S_n}{S_1})^2$$

c_n: when the *scale* is s_n the coefficient is c_n

Because s_1= 1 and s_n = s, hence,

$$c_n = c_1 \times s^{\,2}$$

Now, the expression $S_{AVG} = \frac{k_1}{Span+k_2}$ will be looks like:

$$S_{AVG} = \frac{c_1 \times scale^{\,2}}{Span+k_2}$$

S_{AVG} : *The Sample Average*

C_1: *Constant.*

K_2*: Constant*

When the *scale* =1, the coefficient of the equation is C_1

The main aim is to find the explicit expression of formula, but we still have two unknown variables in the phrase, which are c_1 and k_2. So, the remainder of this research is to figure out c_1 and k_2. In this section, we are trying to find c_1.

The data given in Tables 3 shows when the *scale* =1 in the experiment, the relationship between the amount of the collected data and the value of c_1.Also, based on the information in the table, we can find that: when the number of the collected data goes bigger and bigger, $c_1$will be bigger and bigger. In specific, c_1 will be closer and closer to 2.

n	100	200	300	400	500	600	700	800	900	1000
c_1	1.4339	1.5603	1.6257	1.6688	1.7003	1.7248	1.7443	1.7604	1.774	1.7857
n	1100	1200	1300	1400	1500	1600	1700	1800	1900	2000
c_1	1.7986	1.8067	1.8147	1.8219	1.8284	1.8343	1.8397	1.8447	1.8493	1.8536
n	2100	2200	2300	2400	2500	2600	2700	2800	2900	3000
c_1	1.8576	1.8613	1.8648	1.868	1.8711	1.874	1.8767	1.8792	1.8816	1.8839
n	3100	3200	3300	3400	3500	3600	3700	3800	3900	4000
c_1	1.8862	1.8883	1.8903	1.8922	1.8941	1.8958	1.8975	1.8992	1.9007	1.9022
n	4100	4200	4300	4400	4500	4600	4700	4800	4900	5000
c_1	1.9036	1.9051	1.9064	1.9077	1.909	1.9102	1.9114	1.9125	1.9136	1.9147
n	5100	5200	5300	5400	5500	5600	5700	5800	5900	6000
c_1	1.9157	1.9167	1.9177	1.9186	1.9196	1.9205	1.9214	1.9222	1.9231	1.9243

Table 3: Using convergence to find the c_1Constant. This Figure shows the relationship between the value of c_1 and the Number of Experiment Trials.

Based on the information in table 3, we can draw a graphic to represent the relationship between the number of paired data collected and the value of the c_1

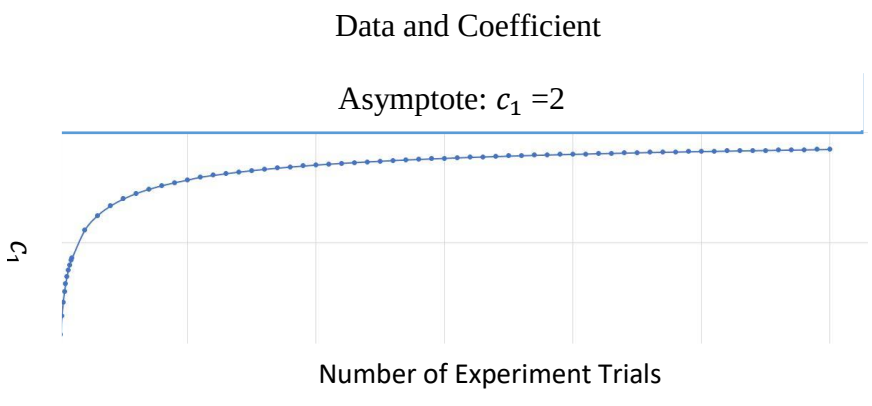


Figure 4.6: c_1 VS Number of Experiment Trials.The correlation between the Number of Experiment Trials n and c_1.

Hence, we can make an **assumption**:

$$c_1 = 2$$

We already know that $c_n = c_1 \times scale^{\,2}$ and based on the assumption in this section, hence we can conclude that

$$c_n = 2 \times scale^{\,2}$$

So,

$$S_{AVG} = \frac{2 \times scale^2}{Span + k_2}$$

S_{AVG} *: The Sample Average*

K_2*: Constant*

4.3.3 Constant

The last step is to figure out k_2 in the formula.

In this section, we will set up a small group of experiments (10 experiments). And in each test, we will let *span =1*.

When *Span = 1*, the formula is:

$$S_{AVG} = \frac{2scale^2}{k_2+1}$$

S_{AVG} *: The Sample Average*

K_2*: Constant*

The data give in Tables 4 shows when the span equals 1 in the experiment, the relationship between *scale* and *Sample Average*

scale	1	2	3	4	5	6	7	8	9	10
S_{AVG}	1	4	9.000562	16	25	36.00899	49.01531	64	81.03542	100

Table 4: The relationship between the *Scale* and the *Sample Average* when the span equals 1

From the table 4, we can find that when *span =1*

$$S_{AVG} = scale^2$$

It means

$$S_{AVG} = \frac{2scale^2}{k_2+1} = scale^2$$

So

$$\frac{2}{k_2+1} = 1$$

Then

$$k_2 = 1$$

Now, we can declare that the relationship between *Sample Average*, *Scale* and *Span* is:

$$\boldsymbol{S_{AVG} = \frac{2scale^2}{Span+1}}$$

4.3.4 Validation

After we find the expression for the equation, we need to validate the assumptions in the process, and the conclusion of the experiments is correct.

We will randomly collect some data to test the equation that we just found (Table 5).

Scale	*Span*	*Average(experiment)*	*Average(Formula)*
1	*13*	*0.142858*	*0.1428571428*
2	*120*	*0.066124*	*0.0661157*
3	*102*	*0.174796*	*0.17475728*
4	*100*	*0.316833*	*0.31683168*
5	*88*	*0.561794*	*0.5617977528*
6	*70*	*1.014384*	*1.01408450704*
7	*15*	*6.126843*	*6.125*
8	*55*	*2.28655*	*2.2857142857*
9	*7*	*20.259176*	*20.25*
10	*99*	*2.000176*	*2.0*
11	*65*	*3.66736*	*3.66667*
12	*1*	*144.035966*	*144.0*
13	*25*	*13.008524*	*13.0*
14	*46*	*8.347248*	*8.34042553*
15	*36*	*12.169706*	*12.162162162*
16	*24*	*20.480091*	*20.48*
17	*47*	*12.045614*	*12.041667*
18	*80*	*8.007081*	*8.0*

Table 5: The Average Sample Value from the Experiment and the Formula

As you can see, Table 5 tells us that the *Sample average* value from the tests matches the *Sample average* value from the equation.

Now, we can conclude that the expression

$$S_{AVG} = \frac{2scale^2}{Span+1}$$

can present the relationship between the *Scale*, *Span*, and the *Sample Average* (See Figure 4.7).

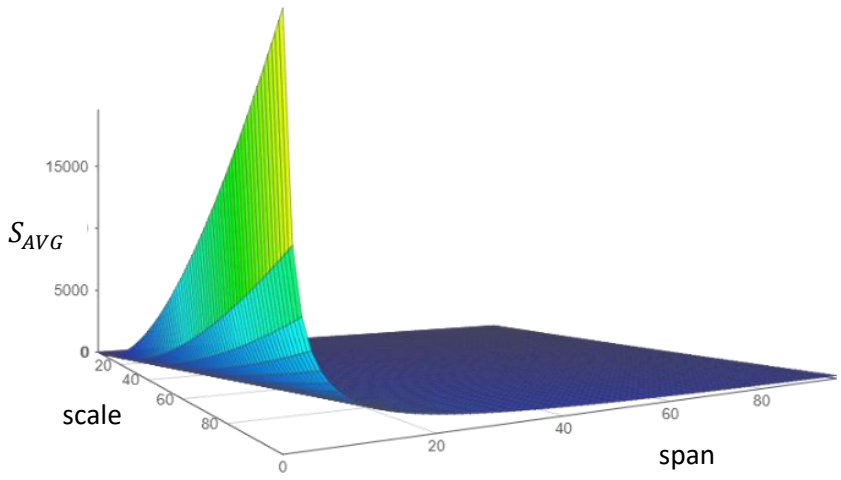


Figure 4.7: The shape of the function $S_{AVG} = \frac{2scale^2}{Span+1}$

in a *3D* plot

Chapter 5

Analysis

We will be interested in analyzing whether or not the Linear Stochastic Sampling algorithm is an excellent model. In this chapter, we explain the Linear Stochastic Sampling model in three different aspects, which are the visual quality, the memory usage, and the performance,

We already find the expression in the previous chapter, so, we can quickly calculate the value of the sample average by using the formula. In the first section, we compare ten different output images, each of these images are generated with different input sample average pixel value only by using the Linear Stochastic Sampling.

Then by comparing with the essential random sampling method, we detail the performance of the Linear Stochastic Sampling Algorithm method.

Next, in the third section, we examine the memory usage between the Linear Stochastic Sampling Algorithm and the primary random sampling method.

Finally, to confirm the conclusion that is delivered in the first section, we set a survey in the last part. The questions in the investigation are based on the output image in the first section. The review is presented in full in Appendix A.

5.1 Visual Quality

The name of the test image in the research is ‘Sunday Afternoon on the Island of La Grande Jatte’ which is one of George Seurat’s most famous works. La Grande Jatte, in English, means the large bowl. It was painted in 1884-1886 and is located in the Art Institute of Chicago. This oil on canvas measured 207.6cm × 308cm and was inspired by people relaxing at La Grande Jatte in Paris (See Figure 5.1) .[31]

Figure 5.1: The test Image in Visual Quality: Sunday Afternoon on the Island of La Grande Jatte

The test image is designed for analysis and quality assessment of different kinds of displays and image processing techniques. The Image depicts fashionable Parisians enjoying a Sunday afternoon at a favorite beauty spot located on the River Seine between Neuilly and Levallois-Perret.[29]

Based on the information in Figure *5.1*, we can conclude that the quality of test image is excellent, all information and the sense of layering in the test image is unambiguous. The test image is showing us it was a current leisure site for picnicking, walking, and boating. For example, the test image is crowded with some forty stereotypical Parisian figures, shown full-face or in profile. Carefully arranged in static groups across the picture, they appear uncommunicative and frozen in time, adding to the dreamlike quality of the painting.

There are *8* experiments in the research, and in each analysis, the *scale* is *100*. At the same time, we will control the sample average pixel value as an input variable; then we will observe the difference between these ten output images.

The pictures give in Figure *5.2* shows *8* different the output images in the experiment. From output a to output h, the average number of pixels in each cell is *1*,*2*,*5*,*10*,*20*,*40*,*100*, and *200*.

From the Figure *5.2*, we can find that when the average pixel value in each cell is 1, the quality of the output image is inferior, which means in *output-a*, we lost too much information. For example, the test image depicts a stylishly-dressed woman with a parasol walking a pet monkey on a leash, but in the *output-a*, we can't find any information to conclude there is a monkey in the same location.

When the average number of pixels in each cell is *2*, the quality of the output image is more apparent. The white spots in *output-b* are less than *output-a*, which means *output-b* contains more information than *output-a*.

Figure 5.2: Different output Images from the Linear Stochastic Sampling Algorithm

But, even though the quality of the *output -b* is better than *output-a*, it is still hard for us to use *output-b* to estimate the test image. At the same time, we can't even predict the outline of the test image by utilizing the *output-b*. For example, in the *output b*, we can see there is an animal in the monkey's location, but we still can't use the information in *output-b* to predict the animal is a monkey.

When the average pixel value is 5, the white spots in the *output-c* is significantly less than the *output-b*. Compared to *output-b*, the perceived sharpness, detail, contrast and the color rendition in *output-c* are much better. For example, we know there is an animal in the monkey's location at *output-b*, but we don't know what kind animal it is. But in *output-c*, we can estimate the shape of the monkey even though it is still not very clear.

Hence, we can conclude that the quality for *output -c* is much better than *output-b*, we could use output-c to estimate the outline of the test image roughly, but it is hard for us to use output-c to precisely predict the sketch for the test image. For instance, it is challenging for us to identify what exists further back in the picture.

When the average is 10, the output is *output-d*, and there are almost no white spots in *output d*. Some information that was lost in *output-c* appears in *output-d*. Scene up close in the image is apparent. For instance, we can clearly see there is a monkey located in the right corner. But we still can't very clearly make out the further back view of the *output d*. For example, there are two boats in the lake on the test image; there is only one boat in the lake that can be found very clearly.

Hence, we can use *output-d* to estimate the close shot for the test image, but it is difficult to evaluate the distant view only by using *output-d*.

Now we will see if we let the average pixel value equal 20, does the distant view in the output image become more explicit or not?

When the input average pixel value is 20, there is no difference in a close shot between *output-e* and *output-d*. But the distant view is enhanced in the *output-e*. For example, the boat located in the upper right corner of the lake is very clear, and the small ship situated in the center of the lake is also evident.

Hence, we can conclude that based on the performance of the output e, the information in the output e can be used to estimate the outline of the test image. Even though someplace in output e are still not very clear, but it is bright enough for us to figure out the shape of the test image.

What if we not only want to get the outline for the test image but also want to estimate the test image more precisely?

So, if we want more precisely to estimate the test image, we need to improve the amount of the average sample value in each cell.

When the input average pixel value is 40, we find that even the quality for *output-f* is not as good as the test image, but, all the information in output f is enough for us not only to estimate the general of smaller shape but also the test image. It looks like all the information in test image can be found in *output-f*. The only shortcoming for *output-f* is that the perceived sharpness is not as excellent as in the original image, it is a little dark compared to the test image.

After we have finished these five experiments, we can find that the image quality between these five output images are entirely different which means we can visually distinguish these images.

Hence, we can conclude that by increasing the value of the input average pixel value, the quality of the output image will be enhanced based on the previous five experiments.

Will this conclusion always be true?

Compared to the *output-f,* the *output g and output-h* are somewhat brighter. We can not quickly and intuitively distinguish between the *output-f, output-g,* and the *output-h*.

We can conclude that based on the test image, when the ratio is equal to 100, and the input average pixel value is or more than 100, the quality of the output image will considerably altered.

5.2 Memory Usage

On January *5th, 2015*, the NASA Hubble Space Telescope released the most significant image ever taken of the Andromeda Galaxy. The resulting image is *69536* × *22230* pixels, which total is *1.5* billion pixels, and requiring *4.3GB* of disk space.

It is the most prominent Hubble image ever released and shows more than100 million stars, and over thousands of star clusters embedded in a section.

The original image is *psb* format, in the experiment, we convert the image to 'PPM6' format and duplicate it. Then the test image in the analysis contains *80032* × *25588* × *3* pixels, which total is *2.05 billion* pixels, and requires *5.9 GB* of disk space.

Figure 5.3: The test image in the research. The NASA Hubble Space Telescope released the test image on January 5th, 2015 which is the most significant image ever taken of the Andromeda Galaxy.

However, the computer was used in the research only has 4GB free RAM. Hence, the Stochastic Sampling method will exceed the RAM capacity and start running slow.

In the Linear Stochastic Sampling Algorithm, the method analyzes all the information in one page by one page, so, the Linear Stochastic Sampling Algorithm is not running out of the free RAM.

If the Stochastic Sampling Algorithm wants to have the same performance as the Linear Stochastic Sampling Algorithm, the computer should have at least available *Image Size* RAM in the experiment.

5.3 Performance

In this section, we compare the performance of the Linear Stochastic Sampling Algorithm with the Stochastic Sampling Algorithm.

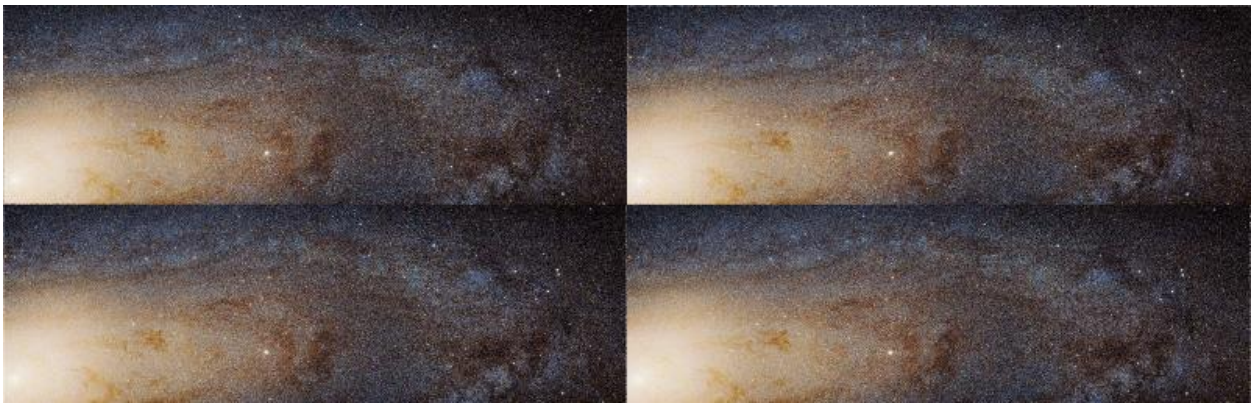
The code given in below shows the pseudocode of the *Linear Stochastic Sampling Algorithm* (Left) and the *Stochastic Sampling Algorithm* (Right) (See Figure 5.4).

```
time_t t;
srand((unsigned)time(&t));
int span = rand()%max;
int index = span;

       ⋮

while (index < size) {
       int x = ((index%source_width)/(Ratio));
       int y = ((index/source_width)/(Ratio));
       fread(&c, 1, 1, f_source);
       buffer->box[y*buffer->width+x].r += c;
       fread(&c, 1, 1, f_source);
       buffer->box[y*buffer->width+x].g += c;
       fread(&c, 1, 1, f_source);
       buffer->box[y*buffer->width+x].b += c;
       buffer->box[y*buffer->width+x].count++;
       span = rand()%max;
       fseek(f_source,3*span, SEEK_CUR);
       index += 1+span;
       }
                    ⋮
```

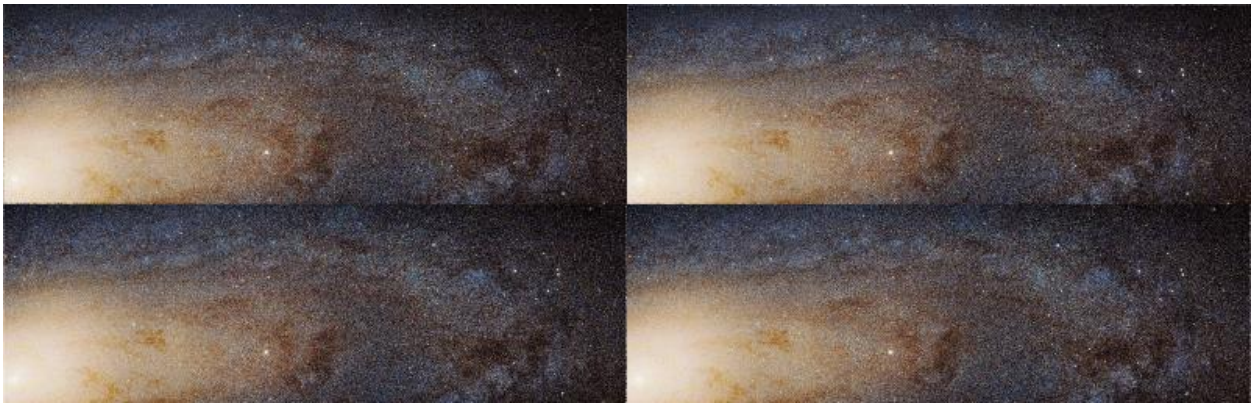
```
time_t t;
srand((unsigned)time(&t));
int span = rand()%max;
long int h_l = readHeader();//header_length
int total_sample_points = 2*size/(max+1);
       ⋮

for (int i =0; i< total_sample_points; i++) {
       int index = rand()% size;
       fseek(f_source,h_l+3*index,SEEK_SET);
       int x = ((index%source_width)/(Ratio));
       int y = ((index/source_width)/(Ratio));
       fread(&c, 1, 1, f_source);
       buffer->box[y*buffer->width+x].r += c;
       fread(&c, 1, 1, f_source);
       buffer->box[y*buffer->width+x].g += c;
       fread(&c, 1, 1, f_source);
       buffer->box[y*buffer->width+x].b += c;
       buffer->box[y*buffer->width+x].count++;
       }

                    ⋮
```

Figure 5.4: The pseudocode of the Linear Stochastic Sampling Algorithm and the Stochastic Sampling Algorithm.

The diagram given in Figure 5.5 shows the output images, which are generated by the Linear Stochastic Sampling Algorithm and the Stochastic Sampling Algorithm under the same condition.



a: The output Image from the *Linear Stochastic Sampling Algorithm*



b: The output Image from the *Stochastic Sampling Algorithm*

Figure 5.5: This Figure shows the output images are generated by the Linear Stochastic Sampling Algorithm and the Stochastic Sampling Algorithm under the same condition

We can find that the image quality between these two output images are very close to each other which means we can visually distinguish these images.

However, under the same condition, the Linear Stochastic Sampling Algorithm only takes *14 seconds* to generate the output image; whereas, Stochastic Samling Algorithm takes *17minutes* and *27 seconds*.

The data given in Tables 6 shows more details of the performance between the Linear Stochastic Sampling Algorithm and the Stochastic Sampling Algorithm.

Scale	*Span*	S_{AVG}	T_{AVG}(Experiment)	T_{AVG}(Forluma)	t_{ls}*(sec)*	t_s*(sec)*
100	*100*	*198*	*40417472*	*40551659*	*16.519*	*10486.989*
100	*500*	*39*	*8147489*	*8175085*	*15.641*	*2534.681*
100	*1000*	*19*	*4075814*	*4091626*	*15.603*	*1047.365*
500	*100*	*4950*	*40417467*	*40551659*	*14.069*	*20257.026*
500	*500*	*998*	*8145596*	*8175085*	*16.989*	*4427.362*
500	*1000*	*499*	*4076759*	*4091626*	*15.336*	*2337.03*
1000	*100*	*19801*	*39618556*	*40551659*	*16.619*	*19613.75*
1000	*500*	*3992*	*79867700*	*8175085*	*16.182*	*4550.812*
1000	*1000*	*1998*	*3997446*	*4091626*	*15.517*	*2308.735*

Table 6: The performance of the Linear Stochastic Sampling Algorithm and the Stochastic Sampling Algorithm

Based on the information in the table and the output images on the experiments, we can conclude that:

under the same condition, the quality of the output images from the Linear Stochastic Sampling Algorithm, and the Stochastic Sampling Algorithm are substantially similar.

However, the Stochastic Sampling Algorithm takes one thousand times longer to run the Linear Stochastic Sampling Algorithm.

5.4 Survey

The survey administered to the participants consisted of ten questions. The investigation is presented in full in Appendix A. The problems in the study were based on the experiments in the previous section. These ten questions are a single choice question.

The results of this survey should be considered preliminary due to the small size of the test group. Nonetheless, the results do seem to indicate that S_{AVG}, which is the average sample point in the Linear Stochastic Sampling Algorithm had an overall positive impact on the quality of the output image. Also, since each member of the test group utilizes the paper survey platform for the work, this does give some reason to believe that all the people are independent. All of these ten questions are given on a ten-points Likert scale.[32]

The test group size for this experiment was too small to make any generalized conclusions about the Linear Stochastic Sampling Algorithm and its usage. Additionally, this study was not conducted in such a way that direct causation of generalization of results to a population can justifiably be inferred, i.e., no random assignment of subjects to groups, and no random sampling from a well-defined population.[33]

Based on the result of the survey, we can conclude that the experimental results are the same as we expected.

When the input average pixel value in some range, the quality of the image is so sensitive which means small increases in the average input pixel yields significantly different image quality. Beyond a particular pixel value (critical value), the quality of the image no longer sensitive to average input values.

Chapter 6

Conclusion and Future work

This chapter gives the conclusion of this research and outlines possible paths for future research related to that of this work.

As stated in the previous chapters, when we fixed the *scale* and the *span*, then the quality of output image and the speed of the output image generation are based on input average pixel value, which is the *Sample Average*. By increasing the amount of the input average pixel value, the quality of the output image will be enhanced, but after a particular pixel value (critical value), when we change the average input value, the quality of the image will not be improved.

Many ideas have come to mind for further research over the period of working on this thesis. Some of these ideas will be outlined here, although it is understood that some of them may not be feasible until we gain more knowledge through newer observatories and subsequent observational data.

As mentioned in conclusion, after critical value, when we change the average input value, the quality of the image will not be improved. Hence, we must consider the relationship between the time to create the output image and the input average pixel value (S_{Avg}), and trying to find the critical value. As such, the further work should be conducted in a more thorough manner, which would allow for stronger conclusions that could better contribute to the body of knowledge concerning both average pixel value as an input and the output image for its production.

6.1 Conclusion

The Linear Stochastic Sampling has been introduced and analyzed which significantly reduce the sampling time and avoids the thrashing in gigapixel images sampling.

The approach of the Linear stochastic sampling is based on the existed stochastic sampling methods, such as jittered sampling method and the Gaussian pyramid Sampling method. The nature of the existing stochastic sampling methods makes it very suitable for the small image stochastic sampling.

If we are trying to use the existing stochastic sampling methods in gigapixel Images sampling, thrashing will happen in the stochastic sampling process.

The new technique avoids the problems of the existed stochastic sampling methods in the gigapixel images sampling. The linear stochastic sampling algorithm can be applied in astronomical and other fields of research that require image processing.

There are the three critical variables in the Linear stochastic sampling process, which are *Span*, *Sample Average*, and *Scale*. The equation relating these three variables has been demonstrated. At the same time, the application of this equation has been described. For instance, the number of samples and the quality of the output image can be controlled by the algorithm. Also, with correctly chosen the distance between the current sample and the next one and the input average pixel value, then, high-speed image processing and the high-quality output image will appear at the same time.

Thrashing avoided, sampling time reduced, and clumping reduction are the three essential aspects of the linear sampling design. The other improvements resulting from Linear Stochastic

Sampling can be achieved by choosing an appropriate value for the span, such that alias-free sampling can be performed.

To make better use of Linear Stochastic Sampling Algorithm, at the end of this document, we briefly introduced two promising aspects of future research. The first one is to maximize the functionality of the equation between the *Sample Average*, *Span*, and *Scale*; the second one is how to retain higher quality for the output image while simultaneously reducing the sampling time.

6.2 Critical Value

In this section, we will analyze the particular value, which was stated in chapter 5, and we will call this specific cost as the *critical value*.

We already know that by increasing the value of the input sample average, the quality of the output image will be enhanced. However, after the input sample average more than the critical value, when we change the average input value, the quality of the output image will not be improved. So, we can draw a picture to represent the conclusion (see Figure 6.1).

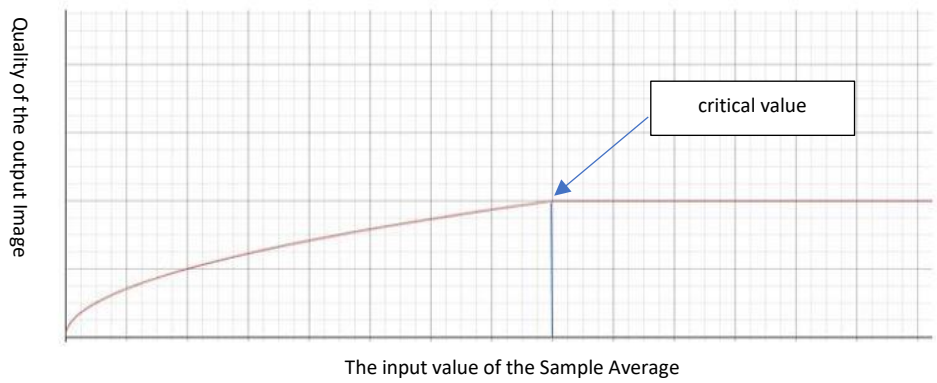


Figure 6.1: The relationship between the quality of the output image and the input value of the Sample Average

Based on Figure 6.1, we determined that the relationship between the quality of the output image and the input average pixel value is a piecewise function.

$$Piecewise\ function = \begin{cases} Function\ is\ a\ increasing\ function & if\ Average < Critical\ Value \\ Function\ is\ a\ Constant\ Value & if\ Average \geq Critical\ Value \end{cases}$$

Hence, if we know the critical value, the algorithm can generate the highest quality output image by using the least time.

So, a study of critical value which is a unique value for input average pixel would give a more definitive and compelling result.

6.3 Convenience Interval

The purpose of the linear stochastic algorithm is to try to save time in the gigapixel image sampling. So, we should focus on how to fast generate the high-quality output image as possible as we can.

The critical value is significant for us, but at the same time, it also will take a long time for us to find it. At the most time, it is reasonable that there is a bias which is the difference between the output image and the test image. Hence, it is not necessary for us to choose the critical value as the input average pixel value each time.

So, we can choose an input average pixel value which is located in a range, which will allow the user to keep the output image quality at a high level and has a more top speed to process the image. It is more comfortable for an estimator to find a range.

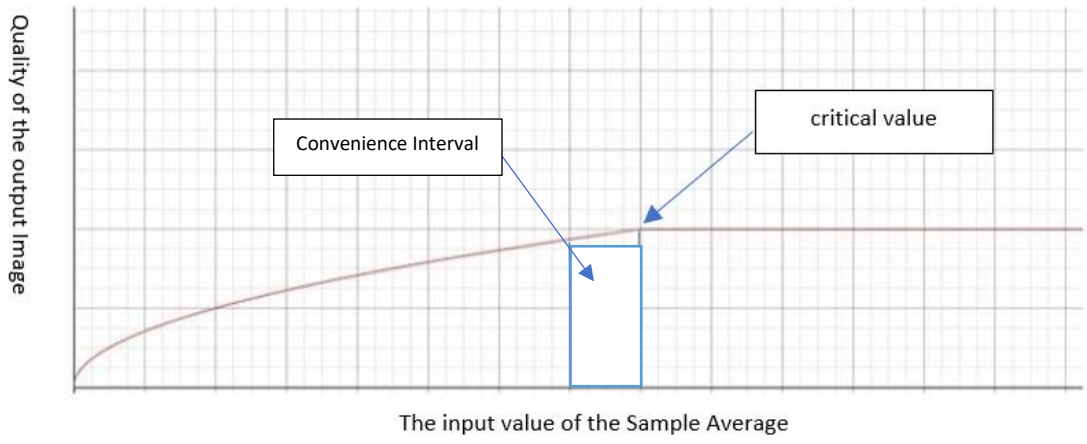


Figure 6.3: The relationship between the Critical Value and the Convenience Interval.

Based on Figure 6.3, we will find that within the convenience interval, the quality of output image will be changed (but not too much) followed by the change of the input average pixel value. Within the convenience interval, the quality of output image is very close to the output image that generated by the critical value.

So, a study of convenience interval which is a set value for input average pixel would provide a more definitive and compelling result.

References

[1] White, Jon L. "Address/memory management for a gigantic LISP environment or, GC considered harmful." Proceedings of the 1980 ACM conference on LISP and functional programming. ACM, 1980.

[2] Li-Yi Wei. Multi-Class Poisson Disk Sampling. January 1, 2009

[3] Robert L. Cook. Stochastic Sampling in Computer Graphics, 1984

[4] Florian Pausinger, Manas Rachh, Stefan Steinerberger. Optimal Jittered Sampling for two points in the unit square. April 20, 2017

[5] Wilson, Paul R. "Pointer swizzling at page fault time: Efficiently supporting huge address spaces on standard hardware." 1991

[6] Derek L. Sonderegger. Introduction to Statistical Methodology. May 2017

[7] Jonas Gomes, Luiz Velho. Image Processing for Computer Graphics. Page 260-262. 1997

[8] Cay S. Horstmann. Big Java 4th Edition. Page 3-4. December 30, 2009

[9] William Stallings. Computer Organization and Architecture 9th Edition.2000

[10] William Stallings. Operating Systems Internals and Design Principles 7th Edition. Page 342-343. January 1, 1998.

[11] Li, Kai. A Shared Virtual Memory System for Parallel Computing.page 94.1988

[12] Silberschatz, Abraham, Peter Baer Galvin, and Greg Gagne. Operating system concepts essentials. John Wiley & Sons, Inc., 2014.

[13] Mark A. Z. Dippi.Antialiasing Through Stochastic Sampling. November 3, 1985

[14] Robert Bridson.Fast Poisson Disk Sampling in Arbitrary Dimensions.August 5, 2007

[15] Kenneth Chiu. Peter Shirley. Changyaw Wang. Multi-Jittered Sampling. 1994

[16] Peter Shirley. Discrepancy as a quality measure for sampling distributions. In Eurographics pages 183-193, September 1991

[17] Don Mitchell. Ray tracing and irregularities of distribution. In Proceedings of Third Eurographics Workshop on Rendering, pages 61-69, 1992.

[18] Andrew Kensler. Correlated Multi-Jittered Sampling. March 5, 2013

[19] P.J. Burt. Fast filter transforms for image processing, Computer Graphics, Image Processing, vol. 6, pp. 20-51, 1981.

[20] Peter J. Burt. Edward H. Adelson. The Laplacian Pyramid as a Compact Image Code. April 1983

[21] ISO/IEC JTC 1/SC 29. "Programme of Work, (Allocated to SC 29/WG 1)". Archived from the original on 2013-12-31. Retrieved 2009-11-07.

[22] "Graphics Interchange Format, Version 87a". W3C. 15 June 1987. Retrieved 13 October 2012.

[23] "ISO/IEC 15948:2004 – Information technology – Computer graphics and image processing – Portable Network Graphics (PNG): Functional specification". Retrieved 2011-02-19.

[24] Allen B. Tucker. Computer Science Handbook, Second Edition. pp. 14-15, November 29, 2004

[25] Whitrow, Robert. OpenGL Graphics Through Applications, pages 39-59

[26] Bill Van Ryper, William Van Ryper, James D. Murray, Encyclopedia of Graphics File Formats, 1994

[27] Agrawala, M., Ramamoorthi, R., Heirich, A., & Moll, L. (2000, July). Efficient image-based methods for rendering soft shadows. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques (pp. 375-384). ACM Press/Addison-Wesley Publishing Co.

[28] Süsstrunk, Sabine, Robert Buckley, and Steve Swen. "Standard RGB color spaces." Color and Imaging Conference. Vol. 1999. No. 1. Society for Imaging Science and Technology, 1999.

[29] Yates, Daniel S.; David S. Moore; Daren S. Starnes (2008). The Practice of Statistics, 3rd Ed. Freeman.

[30] Erlandson, Erik J. (2014-09-11). "Faster Random Samples With Gap Sampling"

[31] Berns, Roy S., et al. "Rejuvenating the color palette of Georges Seurat's A Sunday on La Grande Jatte—1884: A simulation." Color Research & Application 31.4 (2006): 278-293.

[32] Rensis Likert. A technique for the measurement of attitudes. Archives of Psychology, 140:1-55, 1932.

[33] J.Barber. Lectures in statistical methods. “Lecture; given in the course of STA 570 at Northern Arizona University”, 2016

Appendix

User Survey

1: In section 5.1, on a scale from 1-10, compared to the test image, how many points would like give to *output-a* ?

1	2	3	4	5	6	7	8	9	10

2: In section 5.1, on a scale from 1-10, compared to the test image, how many points would like give to *output-b* ?

1	2	3	4	5	6	7	8	9	10

3: In section 5.1, on a scale from 1-10, compared to the test image, how many points would like give to *output-c* ?

1	2	3	4	5	6	7	8	9	10

4: In section 5.1, on a scale from 1-10, compared to the test image, how many points would like give to *output-d* ?

1	2	3	4	5	6	7	8	9	10

5: In section 5.1, on a scale from 1-10, compared to the test image, how many points would like give to *output-e* ?

1	2	3	4	5	6	7	8	9	10

6: In section 5.1, on a scale from 1-10, compared to the test image, how many points would like give to *output-f* ?

1	2	3	4	5	6	7	8	9	10

7: In section 5.1, on a scale from 1-10, compared to the test image, how many points would like give to *output-g* ?

1	2	3	4	5	6	7	8	9	10