

GENERATING RANDOM SHAPES FOR MONTE CARLO ACCURACY
TESTING OF PAIRWISE COMPARISONS

by

Abdullah Almowanes

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science (MSc) in Computational Sciences

The School of Graduate Studies
Laurentian University
Sudbury, Ontario, Canada

© Abdullah Almowanes, 2013

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE

Laurentian Université/Université Laurentienne
School of Graduate Studies/École des études supérieures

Title of Thesis Titre de la thèse	GENERATING RANDOM SHAPES FOR MONTE CARLO ACCURACY TESTING OF PAIRWISE COMPARISONS		
Name of Candidate Nom du candidat	Almowanes, Abdullah		
Degree Diplôme	Master of Science		
Department/Program Département/Programme	Computational Sciences	Date of Defence Date de la soutenance	August 06, 2013

APPROVED/APPROUVÉ

Thesis Examiners/Examineurs de thèse:

Dr. Waldemar W. Koczkodaj
(Supervisor/Directeur de thèse)

Dr. Amr Abdel-Dayem
(Committee member/Membre du comité)

Dr. Haibin Zhu
(Committee member/Membre du comité)

Dr. Andrzej Grzybowski
(External Examiner/Examineur externe)

Approved for the School of Graduate Studies
Approuvé pour l'École des études supérieures
Dr. David Lesbarrères
M. David Lesbarrères
Director, School of Graduate Studies
Directeur, École des études supérieures

ACCESSIBILITY CLAUSE AND PERMISSION TO USE

I, **Abdullah Almowanes**, hereby grant to Laurentian University and/or its agents the non-exclusive license to archive and make accessible my thesis, dissertation, or project report in whole or in part in all forms of media, now or for the duration of my copyright ownership. I retain all other ownership rights to the copyright of the thesis, dissertation or project report. I also reserve the right to use in future works (such as articles or books) all or part of this thesis, dissertation, or project report. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that this copy is being made available in this form by the authority of the copyright owner solely for the purpose of private study and research and may not be copied or reproduced except as permitted by the copyright laws without written authority from the copyright owner.

Abstract

This thesis shows highly encouraging results as the gain of accuracy reached 18.4% when the pairwise comparisons method was used instead of the direct method for comparing random shapes. The thesis describes a heuristic for generating random but nice shapes, called placated shapes. Random, but visually nice shapes, are often needed for cognitive experiments and processes. These shapes are produced by applying the Gaussian blur to randomly generated polygons. Afterwards, the threshold is set to transform pixels to black and white from different shades of gray. This transformation produces placated shapes for easier estimation of areas. Randomly generated placated shapes are used to perform the Monte Carlo method to test the accuracy of cognitive processes by using pairwise comparisons. An on-line questionnaire has been implemented and participants were asked to estimate the areas of five shapes using a provided unit of measure. They were also asked to compare the shapes in pairs. Such Monte Carlo experiment has never been conducted for 2D case. The received results are of considerable importance.

To my parents, for their unconditional love, support and encouragement throughout
my life.

Acknowledgements

First, I would like to express my deep gratitude towards my MSc thesis supervisor Dr. Waldemar W. Koczkodaj, for all the time and effort that he devoted to help me complete this thesis. Prof Koczkodaj defined the subject of this thesis, motivated me, and helped throughout the past two years even during the most difficult times of my study. His useful comments, remarks and engagement through the learning process of the project were very supportive to finalize this master thesis. I would also like to thank Dr. A. Abdel-Dayem and Prof. H. Zhu, for serving on the Advisory Committee, their comments and reviewing the thesis. Moreover, I wish to thank my wife Husa for her personal support and great patience at all times. Without her love, help and support, I would not have been able to finish this thesis. My gratefulness goes also to my parents and sisters who have given me their unequivocal support throughout my studies. In addition, I like to thank the participants in my survey, who have willingly shared their precious time during the process answering my questionnaire. I would like to acknowledge the generous financial support provided by the he Ministry of Higher Education in the Kingdom of Saudi Arabia. Finally, I would like to extend my gratitude to all those who surrounded me and supported me during this process.

Table of Contents

Abstract	iii
Dedication	iv
Acknowledgements	v
Table of Contents	vi
List of Figures	ix
List of Tables	xiii
1 Introduction	1
2 The method of pairwise comparisons	4
2.1 Background	5
2.2 Saaty’s Analytical Hierarchy Process	9
2.3 Reciprocal pairwise comparison matrix	11
2.4 Inconsistency	15
3 Description of A Heuristic for Placated Random Shape Generation	21
3.1 Rationale	23
3.2 Gaussian Blur	24
3.2.1 Background	24
3.2.2 Gaussian Blur Low Pass Filter	34

3.3	The random shape heuristic algorithm	45
3.3.1	Generate Random Polygon	45
3.3.2	Applying the blur and the cut-off	47
3.3.3	Dealing with holes	53
4	Survey	62
4.1	Background	62
4.2	Preparation	65
4.3	Experiment 1	68
4.3.1	Sample	68
4.3.2	Measures	68
4.3.3	Stimuli	72
4.3.4	Procedure	74
4.3.5	Analysis	77
4.3.6	Results	78
4.3.7	Problems encountered in Experiment 1	81
4.4	Experiment 2	82
4.4.1	Sample	85
4.4.2	Measures	85
4.4.3	Stimuli	87
4.4.4	Procedure	88
4.4.5	Analysis	91
4.4.6	Results	91
5	Conclusions	100
	Appendix A Appendix	111
A.1	Java Code	111
A.1.1	Point Class	111

A.1.2	Pointlist Class	112
A.1.3	RandomPoint Class	113
A.1.4	Gaussian Class	115
A.1.5	Image2Array Class	120
A.1.6	SquareUnit Class	122
A.2	PHP Code	125
A.2.1	Consent form error checking and validating	125
A.2.2	Experiment 1	126
A.2.3	Experiment 2	132
A.3	MySQL	137
A.3.1	Experiment 1	139
A.3.2	Experiment 2	140

List of Figures

2.1	A decomposition of a problem into a hierarchy [59]	8
3.1	The first images to be transmitted using cable picture transmission system[18]	25
3.2	An image sent in 1922 using the improved system [18]	25
3.3	The first picture of the moon taken by Ranger 7 on July 31, 1964 [18]	26
3.4	An edge detection example[17].	28
3.5	Image sharpening example [17].	29
3.6	Image blurring example [17].	29
3.7	Applying $5 * 5$ mean low pass filter [13].	34
3.8	A 1D Gaussian distribution	38
3.9	The shape before applying Gaussian blur	41
3.10	The shape after applying Gaussian blur	42
3.11	A $500 * 500$ image with 11 points shape	47
3.12	An output of a kernel with 9 radius	49
3.13	Blurring a random shape example	50
3.14	The smooth looking shape after applying the cut-off using a 200 threshold	52
3.15	A terminal screenshot of the area of the shape output along with other information	52
3.16	A 7 points shape	53
3.17	A 7 points shape with 20 Gaussian radius	53
3.18	The shape after applying a 100 threshold	54

3.19	The shape after applying a 200 threshold	54
3.20	The shape after applying a 255 threshold	55
3.21	The shape after applying a 40 Gaussian kernel	55
3.22	The shape after applying a 127 threshold	56
3.23	The shape after using a 40 Gaussian radius and a 218 threshold	56
3.24	A star looking shape	57
3.25	A star looking shape with Gaussian blur of 10	57
3.26	A star looking shape with Gaussian blur of 10 and a 254 threshold	58
3.27	A star looking shape with Gaussian blur of 35	58
3.28	A star looking shape with Gaussian blur of 35 and a threshold of 240	58
3.29	An 8 points shape	59
3.30	An 8 points shape and a Gaussian of size 25	59
3.31	An 8 points shape and a Gaussian of size 25 and 127 threshold	60
3.32	An 8 points shape and a Gaussian of size 45 and 150 threshold	60
3.33	An 8 points shape and a Gaussian of size 45 and 157 threshold	60
3.34	An 8 points shape and a Gaussian of size 45 and 206 threshold	61
3.35	A more extreme example	61
3.36	A 50 points shape with 50 Gaussian radius and 127 threshold	61
3.37	A 50 points shape with 150 Gaussian blur radius and a 127 threshold	61
4.1	The 5 shapes to used for comparison [62]	63
4.2	The 5 Equal shapes with the square unit used in [1]	64
4.3	The questionnaire used in [1]	64
4.4	A flowchart describing the step taken to perform the survey	66
4.5	The consent form used	67
4.6	The select 5 shapes page	69
4.7	How to use the unit to estimate the area example	70
4.8	Taks 2: estimate area of a shape in units	71
4.9	Taks 3: Compare the two shapes	72

4.10	Experiment 1 flowchart	73
4.11	Not too simple shape	74
4.12	A shape with original area 121787 pixels before scaling	76
4.13	A shape with original area 71895 before scaling	76
4.14	Analysing and calculating the relative error for area estimation in units	77
4.15	Histogram showing the average error when using the direct method .	79
4.16	Histogram showing the average error when using pairwise comparisons method	80
4.17	Histogram showing the inconsistency when using pairwise comparisons method	80
4.18	The average time needed to complete each task in experiment 1 . . .	81
4.19	Task 3: pairwise comparison page (experiment 1)	83
4.20	Order shapes from largest to smallest screen	86
4.21	Pairwise comparisons used in experiment 2	88
4.22	Experiment 2 Flowchart	89
4.23	A pairwise comparisons matrix in JConcluder	91
4.24	Weights of criterias example in JConcluder	92
4.25	Histogram showing the average error when using the direct method in experiment 2	94
4.26	Histogram showing the inconsistency in the pairwise comparisons in experiment 2	95
4.27	Histogram showing the average error when using the pairwise compar- isons method in experiment 2	95
4.28	Comparing the average error rate when using the pairwise comparisons and the direct method for area estimation of random shapes	96
4.29	The time taken to complete each task in experiment 2 in minutes . .	96
4.30	The most popular shape selected	97
4.31	The second most popular shape selected	97

4.32	The third most popular shape selected	98
4.33	The shapes that was not selected not even once	98
4.34	Shapes selected only once or twice	99
A.1	Creating the unit square interface	125
A.2	The MySQL overall view	138

List of Tables

2.1	A Nine point scale for pairwise comparison by [56]	11
2.2	A pairwise comparison matrix A	13
2.3	Average RI for different n	16
2.4	Average value of λ_{max} of randomly generated pairwise comparison matrices, RI_n , the number of matrices with $CR \leq 10\%$, $GD \leq 1$ and $GD \leq 2$ [5]	19
3.1	Pascal's Triangle	36
3.2	A commonly used 15 x 15 Gaussian filter	39
3.3	Comparison between the 3 different algorithms	44
3.4	The placated random shape generation heuristic algorithm	45
4.1	A pairwise comparison matrix for the 5 geometric shapes [62]	62
4.2	A pairwise comparison matrix example for one of the records	78
4.3	The impact of bad screen design [15]	84
4.4	Experiment 2 steps	90

1 Introduction

In this thesis, random placated shapes [3] are generated for Monte Carlo accuracy testing of the pairwise comparisons method. The Monte Carlo method [45] is a statistical numerical method that may be used for solving problems and has been applied to many scientific fields [42]. It relies on the generation of a sequence of random numbers [26]. The Monte Carlo method follows the following pattern. First, the domain of the input is defined. Next, inputs are randomly generated. Afterwards, the computation and testing are performed to discover the outcomes.

A placated shape is a smooth-looking shape without sharp edges and corners. It is a random but visually nice smooth-looking shape. Random but visually nice shapes are often needed for cognitive experiments and processes. These shapes also can be used in many other areas such as computer games or software testing. Visual perceptual skills can be assessed using placated shapes and the pairwise comparisons method. The pairwise comparison method quantifies the relationship between the intensity of physical stimuli and their perceptual effects [65]. Random shapes that are not too tricky to estimate their area are used for the cognitive experiment. Our study demonstrates an algorithm for generating these placated nice random shapes. No one really knows what a nice shape is. However, we can recognize nice shapes once we see them and more importantly, we can generate them. The second part of this study relies on placated shapes, which are used for testing the accuracy of the pairwise comparisons method.

The pairwise comparison is a practical and simple method. Its main goal is to establish the relative preference of n stimuli in situations where it is impractical to provide

estimates for the stimuli [23]. The pairwise comparison method has been widely used in various domains such as in nuclear power [48] and in transportation systems [58]. In everyday life, people tend to make decisions. A great decision is one that leads to the best outcome. In every country or region, there are standards regarding measurements that may help in the process of decision-making. For example, people use meters, feet, grams, and pounds for measuring length or weight. These assist in selecting the better optimal option when there are different alternatives. Nonetheless, there are many entities that cannot be measured using standard measures, so the pairwise comparison method can be used in such situations. The method of pairwise comparisons is probably as old as humankind. It is easy to imagine how people in the past compared two different commodities by weighing them in each hand for a fair exchange. Assessment of intangible criteria (e.g., the degree of environmental pollution or public safety) involves not only imprecise or inexact knowledge, but also inconsistency in our own assessments. The pairwise comparisons method can always be used to reach final conclusions elegantly. The pairwise comparisons method is of considerable importance in situations where direct measurements are impossible to perform. It makes a natural and a powerful tool for decision-making. It is a natural approach for processing subjectivity, although objective data can also be processed this way. By common sense, and for any type of comparisons, taking two criteria or alternatives at a time works better than taking all of them at once. Evidently, handling multiple things at once is more difficult.

To perform the random shape Monte Carlo accuracy testing of pairwise comparisons, an online questionnaire was implemented and acted as our data collection method. Participants were asked to estimate areas of five shapes using a provided unit. In addition, they were asked to compare the shapes in pairs. The average error rate was then calculated for both and compared. The results were encouraging as the gain of accuracy reached 18.4% when the pairwise comparisons method was used. To our own knowledge and based on an intensive search, this is the first study in the world

for Monte Carlo 2D accuracy testing of pairwise comparisons.

This thesis is composed of three different chapters, in addition to the introduction and conclusion. Chapter 2 of this thesis describes the method of pairwise comparisons in full details. It provides a broad background survey of the method as well as a description of Saaty's Analytical Hierarchy Process. It also describes the inconsistency concept of the pairwise comparison matrix including Koczkodaj's distance-based inconsistency introduced in [32] and independently analyzed in [5]. The aim of Chapter 3 is to provide an overview description of the placated random shape generation algorithm. In Chapter 4, the questionnaire, which was used for data collection, is described and all the steps performed to complete the two experiments are described.

2 The method of pairwise comparisons

When comparing different entities, one tends to assign a single quality score to each entity. If we have three different images of the same object, each with different quality scale, many questions come to mind. Which question should be asked: How much does the quality of one image look compared to the other or would it be better to ask simply if the image looks good [68]. Scales of measurement with zero placement and measurement unit are known as interval scales [64]. The interval between any two scale values has a meaning, but the numerical value of any single score is arbitrary. Equivalent interval scales can be defined with different zeros and units. For example, the Fahrenheit and Celsius temperature scales are equivalent interval scales with different zero placements, and different description of the amount of heat represented by 1 degree. It is possible to convert between any two equivalent interval scales by shifting and multiplicatively scaling the scale values [68]. In some situations, it is very difficult to imagine measuring some objects without having universal standards. In every country or region there are some standards regarding measurements. For example, when measuring length or weight people use, meter, foot, kg, and pounds. Nonetheless, there are many entities that cannot be measured using standard measures.

The pairwise comparisons method can always be used to reach strong and clever final conclusions without too much difficulty. It is a common sense rule to take two criteria or alternatives at a time rather than all at once. The reason is that handling multiple things at once is more difficult. The pairwise comparisons method is of high importance due to the fact that there are certain situations where direct

measurements are impossible to perform. This is a natural and a powerful tool that is not practicality widely used, which may be due to the lack of scientific evidence where better accuracy can be achieved using the pairwise comparisons method [35]. Therefore, we are trying in our research to show that the error rate decreases when pairwise comparison method is used for area estimation for smooth random shapes. Therefore, the accuracy increases when the pairwise comparisons method is used.

2.1 Background

The pairwise comparisons method has been in use for thousands of years. People of the Stone Age were using it as a method of trade. To illustrate, they weighed a fish in one hand with a bird in another hand to judge how valuable is each one. They would then decide if they wanted to complete the transaction or not. However, it was not until the 1300's that the pairwise comparison method was scientifically introduced. It was introduced by Remon Llull who was born in 1232 and died in 1316. Llull lived in the Kingdom of Majorca in Catalan, Spain and he is one of the "founding fathers of voting theory and social choice theory" [8]. Llull proposed an efficient system of exhaustive binary comparisons. His voting systems is used in the analysis of certain modern sports tournaments [8].

Nicolas de Condorcet, who is a French social scientist, mathematician, philosopher and a human rights advocate, introduced a few important concepts related to pairwise comparisons in 1785. One of these important concepts was the Condorcet's jury theorem. The theory states and translates to the following. If each member of a voting group is more likely than not to make a correct decision, the probability that the highest vote of the group is the correct decision increases as the number of members of the group increases. It was used to simulate pairwise elections between all candidates in an election. He also explained the Condorcet's paradox which demonstrates that the majority of preferences become intransitive with three or more candidate [43] [9].

In 1927, Thurstone [11] introduced the law of comparative judgment, which is the base of all experimental work for all educational and psychological scales in which comparative judgments are involved. The law of comparative judgment can be used in many areas such the comparison of physical stimulus intensities, qualitative comparative judgments, and psychological values. It can be used with samples with different gray value, weights, or any other quantitative or qualitative value that need to be compared [67]. When examining two samples for comparison, there should be some kind of method to choose a preferred option from the compared pair. According to [67], observers are not consistent in the comparative judgments when judging the same pair in successive times if there is a just noticeable difference. Therefore, responses from the same responder for a given sample may fluctuate. The law of comparative judgment “applies fundamentally to the judgments of a single observer who compares a series of stimuli by the method of paired comparison when no equal judgments are allowed. It is a rational equation for the method of constant stimuli. It is assumed that the single observer compares each pair of stimuli a sufficient number of times so that a proportion may be determined for each pair of stimuli” [67]. The mathematical formula for the law of comparative judgment is shown below [67]

$$S_1 - S_2 = x_{12} \sqrt{\sigma_1^2 + \sigma_2^2 - 2r\sigma_1\sigma_2} \quad (2.1)$$

Where S_1 = the psychological scale value of stimuli 1.

S_2 = the psychological scale value of stimuli 2.

x_{12} = the sigma value corresponding to the proportion of judgment. It is either a positive or a negative value.

σ_1 = discriminal desperation 1

σ_2 = discriminal desperation 2

r = correlation between σ_1 and σ_2

Pairwise comparison method is mainly used to subjectively compare objects. This

means that it is used to compare objects that are difficult or impossible to measure. It may be used in all kinds of preference testing. In some situations, it is the only feasible experimental technique [11].

It is generally accepted that the use of the pairwise comparison that is performed today took place in 1977 [57]. It is a natural approach for processing subjective data, although objective data can be also processed this way. In 1977, Thomas L. Saaty proposed a hierarchical structure of pairwise comparison. According to [57], the biggest problem of decision theory is how to get the weight for a set according to the importance of each element of that set. This is due to the fact that importance is judged according to criterion that may be shared by all or some of the elements. In other words, the most important obstacle for making a decision is to develop weights for some set of activities agreeing to the importance of each one of them. Saaty's method scaled the weights of the elements in each level of the hierarchy with respect to the higher level element and a pairwise comparison matrix was constructed. Take the "best house to buy" shown in [59] as an example. When an average income family wants to buy a house, they identify 8 criterias that they have to look for in a house. The problem has to be seen as a hierarchy with the top level as the goal, the second level as the criteria, and the third as the candidate houses to be evaluated. See Figure 2.1 for how the problem is illustrated in an hierarchy. Intuitively, it is obvious that the "two at a time" approach is better than "everything at once" method for any kind of comparisons. To show that the pairwise comparisons method is superior to the common sense by an expert's eye approach, is not entirely a trivial task since there are many hurdles to overcome. Saaty's Analytical Hierarchy Process introduced in 1980 a method that aims to derive ranking order from pairwise comparisons. The values of comparisons can be obtained from either an actual measurement or a subjective perspective [56]. These days, there are many evidences that the pairwise comparisons method can be applied in many aspects in life. It can be used in the medical field as shown in [30] where medical scale predictably improved using the pairwise

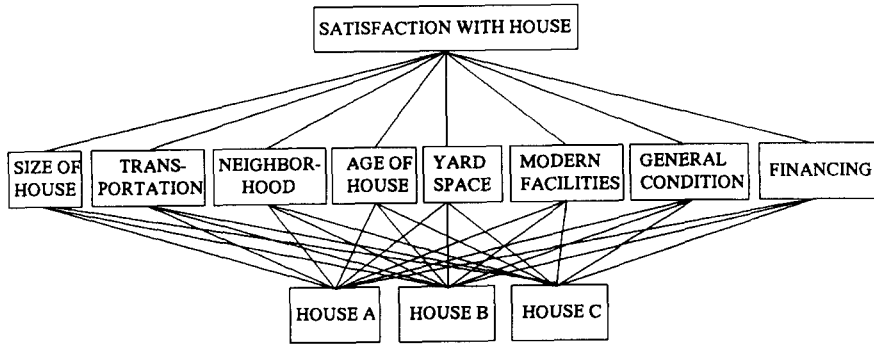


Figure 2.1: A decomposition of a problem into a hierarchy [59]

comparisons method. The method of pairwise comparisons can be applied to both objective measures, such as distance, by comparing length and subjective measures like air pollution [74]. In addition, a study was introduced in [29] that demonstrates how to strengthen the commonly used World Health Organizations Quality of Life Index (WHOQOL) by using the consistency driven pairwise comparisons method. It is also used in medical knowledge mining for image data to assess medical images for early stroke detection [40]. In 2010, a new concept of pairwise comparisons ranking data and pairwise comparisons consistent ranking data have been introduced [74]. Moreover, development and validation of a pairwise comparison scale for user experience (UX) evaluations with preschoolers was introduced in [73]. It has even been used in work and publication of material that deals with the conflict between Israel and the Palestinians in the Middle East [62]. Likewise, [28] propose a strong error tolerant algorithm for ranking that only requires that the pairwise comparisons are probably correct. Besides, the pairwise comparisons method was used to build an expert system for construction tending process [37]. It was also used in selecting the best strategy in the software certification process [4]. Furthermore, it was used on nuclear power plants [48] and in transportation systems [58].

Research in pairwise comparison went even further. By not using any numbers, a consistency driven algorithm has been presented and shown is in [74] as a method of pairwise comparison. In [6], a useful tool for showing how to handle incomplete pair-

wise comparison matrices as a natural extension of the complete case was introduced. Other examples include fuzzy pairwise comparisons extensions of the multi criteria analysis with pairwise comparisons under a fuzzy environment [41] [12], and deriving priorities from fuzzy pairwise comparisons judgments [46]. According to [24], it is also encouraging to know that the triad-based algorithm for improving consistency in the pairwise comparisons matrix are convergent. This also means that theoretical grounds for trust in the pairwise comparisons method have been established. The study of pairwise comparisons began to receive more attention due to the great accomplishments achieved in research. Additional research on pairwise comparisons is still undergoing as it is a very important part of the decision making process.

2.2 Saaty's Analytical Hierarchy Process

The Analytic Hierarchy Process (AHP) is a theory of measurement that uses the pairwise comparisons method. The theory relies on the judgments of experts to derive a priority scale. The comparisons are made using a scale of absolute judgments. These judgments represent how much is one element better than another with respect to a given attribute. The judgments may be inconsistent and consistency is a concern in the analytical hierarchy process [61]. An organized decision maker should follow the following analytical hierarchy process. The first step is to define the problem and to determine the level of knowledge needed. The next step is to structure the decision hierarchy from the top level (goal), through the intermediate level (criteria), to the lowest level (alternatives). Next, construct the pairwise comparison matrices where elements in the upper level are compared with the elements in the level below. The last step is to use the priorities obtained from comparisons to weigh the priorities in the level immediately below. This has to be repeated for all elements. Then for each element in the level below, add its weighed values and obtain its overall or global priority. The process of weighing and adding continues until the final priorities of the alternatives in the lowest most level are obtained [61].

According to [40], the hierarchy reduces the number of comparisons from (n^2) to $(nlmn)$. For example, a case with 49 features would require 1176 comparisons without a hierarchy and only 168 comparisons if a hierarchy is used, grouping them into seven features each. In order to assist in the process of making the decision using the pairwise comparison method, a nine point scale is used. This is to quantify pairwise preference of one element over the other. The scale, introduced by Saaty, is shown in Table 2.1 [56].

Table 2.1: A Nine point scale for pairwise comparison by [56]

Description	Scale
Equally preferred	1
Equally to moderately	2
Moderately preferred	3
Moderately to strongly	4
Strongly preferred	5
Strongly to very strongly	6
Very strongly preferred	7
Very strongly to extremely	8
Extremely preferred	9

2.3 Reciprocal pairwise comparison matrix

Making decisions requires comparing alternatives with respect to a set of criterias. Whenever the number of criteria increases, the number of pairwise comparisons increases. There are $n(n - 1)/2$ pairwise comparisons for any n criterias. Criterias in the pairwise comparisons method are presented in pairs. We need to evaluate individual alternatives to derive the respective weight for the criteria. The next step is to construct the overall rating scheme of the alternative. That is to reach the best choice possible [35]. For stimuli $A_1, A_2, A_3, \dots, A_n$, weights are $w_1, w_2, w_3, \dots, w_n$ and the weight ratio matrix is W . The pairwise comparisons matrix is A which represents the preference between individual pairs of alternatives. The element a_{ij} is the estimate of the actual weight ratio w_{ij} [36].

$$W = \begin{pmatrix} \frac{w_1}{w_1} & \frac{w_1}{w_2} & \frac{w_1}{w_3} & \dots & \frac{w_1}{w_n} \\ \frac{w_2}{w_1} & \frac{w_2}{w_2} & \frac{w_2}{w_3} & \dots & \frac{w_2}{w_n} \\ \frac{w_3}{w_1} & \frac{w_3}{w_2} & \frac{w_3}{w_3} & \dots & \frac{w_3}{w_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{w_n}{w_1} & \frac{w_n}{w_2} & \frac{w_n}{w_3} & \dots & \frac{w_n}{w_n} \end{pmatrix} \quad (2.2)$$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n2} & \dots & a_{nn} \end{pmatrix} \quad (2.3)$$

All elements in the two matrices are positive and both matrices are reciprocal. A reciprocal matrix is a matrix that satisfies the following:

$$a_{ij} = \frac{1}{a_{ji}}$$

For $i, j = 1, 2, 3, \dots, n$.

A number of different methods have been recommended to convert judgments in the pairwise comparisons matrix A into a numerical scale. For a positive reciprocal matrix $A = [a_{ij}]$ and a vector $w = (w_1, w_2, \dots, w_n)$, where the ratio matrix $[w_i, w_j]$ is an approximation to A . To find w , eigenvalue is used where the vector of weights is an eigenvector w corresponding to the eigenvalue λ_{max} [57] [35].

Suppose that a set of n objects in pairs according to their relative weights and have a ratio scale need to be compared. Where $A_1, A_2, A_3, \dots, A_n$ are the objects and $w_1, w_2, w_3, \dots, w_n$ are the weights. The pairwise comparison is represented by the following pairwise comparison matrix A shown in Table 2.2.

According to the Perron-Frobenius Theorem [54] [14], the max eigenvalue λ_{max} is a real and positive number. In addition, the elements of the reciprocal matrix A are all positive [57]. If A is multiplied by the $w^t = (w_1, w_2, w_3, \dots, w_n)$,

Table 2.2: A pairwise comparison matrix A

	A_1	A_2	A_3	\dots	A_n
A_1	w_1/w_1	w_1/w_2	w_1/w_3	\dots	w_1/w_n
A_2	w_2/w_1	w_2/w_2	w_2/w_3	\dots	w_2/w_n
A_3	w_3/w_1	w_3/w_2	w_3/w_3	\dots	w_3/w_n
\vdots	\vdots	\vdots	\vdots	\ddots	\vdots
A_n	w_n/w_1	w_n/w_2	w_n/w_3	\dots	w_n/w_n

$$Aw = nw.$$

To find w , solve the system

$$(A - nI)w = 0$$

for the unknown w . This gives a non zero solution if and only if n is an eigenvalue of A . Furthermore, all the eigenvalues λ_i for $i = 1, 2, 3, \dots, n$ are zeros except one. The matrix A also satisfies the cardinal consistency property $a_{ij}a_{jk} = a_{ik}$ and is called consistent. This means that if a row in A is given, other entries can be determined from this relation [57].

Here is an example to illustrate how the eigenvalue method is used. This is a 3×3 reciprocal matrix

$$A = \begin{pmatrix} 1 & \frac{1}{3} & 5 \\ 3 & 1 & 7 \\ \frac{1}{5} & \frac{1}{7} & 1 \end{pmatrix}$$

To normalize A , add the values of each column.

$$1 + 3 + \frac{1}{5} = \frac{21}{5}, \quad \frac{1}{3} + 1 + \frac{1}{7} = \frac{31}{21}, \quad 5 + 7 + 1 = 13.$$

Now, divide each element of the matrix by the sum of its column to normalize relative weights.

$$A = \begin{pmatrix} \frac{5}{21} & \frac{7}{31} & \frac{5}{13} \\ \frac{15}{21} & \frac{21}{31} & \frac{7}{13} \\ \frac{1}{21} & \frac{3}{31} & \frac{1}{13} \end{pmatrix}$$

After that, normalized principal eigenvector has to be calculated by getting the average of the sum of each row of the normalized matrix.

$$w = \begin{pmatrix} \frac{5}{21} & \frac{7}{31} & \frac{5}{13} \\ \frac{15}{21} & \frac{21}{31} & \frac{7}{13} \\ \frac{1}{21} & \frac{3}{31} & \frac{1}{13} \end{pmatrix} = \begin{pmatrix} 0.848 \\ 1.930 \\ 0.221 \end{pmatrix}$$

$$w = \frac{1}{3} \begin{pmatrix} 0.848 \\ 1.930 \\ 0.221 \end{pmatrix} = \begin{pmatrix} 0.282 \\ 0.643 \\ 0.073 \end{pmatrix}$$

The relative weights vector W is the relative weight between subjects being compared. Notice that the sum of $0.282 + 0.643 + 0.073 = 1$, because the eigenvector is normalized. The following shows how to calculate the normalized principal vector W for A using the eigenvalue.

$$A = \begin{pmatrix} 1 & \frac{1}{3} & 5 \\ 3 & 1 & 7 \\ \frac{1}{5} & \frac{1}{7} & 1 \end{pmatrix}$$

Then, use $\det(A - \lambda I) = 0$ to solve for λ and since I is the identity matrix and is already known.

$$\det(A - \lambda I) = \begin{vmatrix} 1 - \lambda & \frac{1}{3} & 5 \\ 3 & 1 - \lambda & 7 \\ \frac{1}{5} & \frac{1}{7} & 1 - \lambda \end{vmatrix} = 0$$

By calculating λ , $\lambda_{max} = 3.0649$ can be obtained. By substituting λ in the previous equation,

$$\det(A - \lambda I) = \begin{pmatrix} -2.0649 & \frac{1}{3} & 5 \\ 3 & -2.0649 & 7 \\ \frac{1}{5} & \frac{1}{7} & -2.0649 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = 0$$

$$W = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} 0.392 \\ 0.9140 \\ 0.1013 \end{pmatrix}$$

and the normalized W would look like

$$W = \begin{pmatrix} 0.279 \\ 0.649 \\ 0.0719 \end{pmatrix}$$

which adds to one. Notice that this W , or W_2 and the previous estimated W or W_1 are almost the same.

$$W_1 \approx W_2$$

$$\begin{pmatrix} 0.282 \\ 0.643 \\ 0.073 \end{pmatrix} \approx \begin{pmatrix} 0.279 \\ 0.649 \\ 0.071 \end{pmatrix}$$

2.4 Inconsistency

Consistency is an important aspect of representing real life problems using scales. Inconsistency of the pairwise matrix was introduced to make the pairwise comparisons matrix in a relatively acceptable scale. However, statistical measures, intuition, and logic are important too. Saaty, the greatest single contributor to the popularization of the pairwise comparisons method [35], proposed a method to analyze the inconsistencies in the pairwise comparisons matrix [57]. Inconsistency is a rescaling of

the largest eigenvalue. Saaty's Inconsistency Index is calculated using the following formula.

$$CI_n = \frac{\lambda_{max} - n}{n - 1} \quad (2.4)$$

Where CI_n is the consistency index and λ_{max} is the largest eigenvalue of a pairwise comparisons $n * n$ reciprocal matrix A . It is also important to mention that the inconsistency index means nothing by itself. It should be compared with a benchmark to determine the magnitude of the deviation from consistency. Saaty demonstrated that if a decision maker is fully consistent, then $\lambda_{max} = n$ and $CI = 0$. On the other hand, $\lambda_{max} > n$ is used if the decision is not fully consistent. CI is always non negative since $\lambda_{max} \geq n$. A consistency ratio CR was introduced by Saaty to measure the consistency. Let RI be the benchmark, $CR = \frac{CI}{RI}$ RI is the average value of CI from 500 positive reciprocal pairwise comparison matrices whose entries are randomly generated using the 1 to 9 scale introduced earlier. See Table 2.3[53].

Table 2.3: Average RI for different n

n	1-2	3	4	5	6	7
RI_n	0	0.58	0.90	1.12	1.24	1.32

For the given example above, $\lambda_{max} = 3.0649$ and the size of the PC matrix is $n = 3$. Thus, the consistency index is

$$CI = \frac{\lambda_{max} - n}{n - 1} = \frac{3.0649 - 3}{3 - 1} = 0.03245.$$

According to Table 2.3 for $n = 3$, $RI = 58$. Therefore,

$$CR = \frac{CI}{RI} = \frac{0.03245}{0.58} = 5.59\% \leq 10\%$$

which means that the pairwise comparisons matrix is consistent. If the matrix is consistence, $\lambda_{max} = n$, $CI_n = 0$, and $CR_n = 0$. Saaty concluded that an inconsistency ratio of 10% or less is acceptable. The decision maker is sufficiently consistent when $CR < 0.10$ [53]. Saaty improved this ratio in [60] to 0.08 and to 0.05 for $3 * 3$ and $4 * 4$ matrices respectively.

Koczkodaj introduced a new definition of inconsistency in [32] because of the two main problems with Saaty's inconsistency definition. According to Koczkodaj, these problems are the 10% rule of thumb and the lack of being able to exactly locate the inconsistency. The eigenvalues is a global characteristic of a matrix and one can't say which matrix element contributes more to the inconsistency. Koczkodaj proposed a method to calculate inconsistency based on a measure of deviation from the nearest consistent reciprocal matrix. It is known as the distance-based inconsistency definition. For example, when $n = 3$, and if a basic reciprocal matrix,

$$A = \begin{pmatrix} 1 & a & b \\ \frac{1}{a} & 1 & c \\ \frac{1}{b} & \frac{1}{c} & 1 \end{pmatrix}$$

then it is reduced to a vector of three coordinates $[a, b, c]$ [5]. In the consistent case, $b = ac$ holds and we can create three consistent reciprocal matrices. This is done by the computation of one coordinate from computing the remaining two coordinates.

$$\left(\frac{b}{c}, b, c\right), (a, ac, c), \text{ and } \left(a, b, \frac{b}{a}\right)$$

The inconsistency index of a 3×3 pairwise comparisons matrix is $CM(a, b, c)$.

$$CM(a, b, c) = \min\left\{\frac{1}{a}|a - \frac{b}{c}|, \frac{1}{b}|b - ac|, \frac{1}{c}|c - \frac{b}{a}|\right\}$$

For any $n * n$ matrix with $n > 2$,

$$CM(A) = \max\left\{\min\left\{|1 - \frac{b}{ac}|, |1 - \frac{ac}{b}|\right\} \text{ for each triad } (a, b, c) \text{ in } A\right\}.$$

The number of all possible triads of an $n \times n$ pairwise comparisons matrix equals:

$$n(n-1)(n-2)/3!$$

In the case of $4 * 4$ pairwise comparison matrix and a scale of 1 to 5, the threshold should be $1/3$ [38]. To define a threshold for higher dimensions pairwise comparisons matrices, Koczkodaj proposes one grade off and two grades off rules [38].

The grade difference, $GD(a, b, c)$ determines the approximation of an element by the two other elements [38].

$$GD(a, b, c) = \min\{\max\{|a - \frac{b}{c}|, |\frac{1}{a} - \frac{c}{b}|\}, \max\{|b - ac|, |\frac{1}{b} - \frac{1}{ac}|\}, \max\{|c - \frac{b}{a}|, |\frac{1}{c} - \frac{a}{b}|\}\}$$

The one grade off rule is

$$GD(a, b, c) \leq 1$$

and the two grade off rule is

$$GD(a, b, c) \leq 2$$

For matrices of higher order, the one grade off and the two grades off rules are

$$GD(A) = \max\{GD(a, b, c) \text{ for each triad } (a, b, c) \text{ in } A\} \leq 1 \text{ or } 2,$$

Koczkodaj's inconsistency index for 4×4 pairwise comparisons matrices is stricter than Saaty's. The following is an example which shows that Saaty's 10% threshold allows higher inconsistency when $\frac{1}{9}, \dots, 1, \dots, 9$ scale is used [5].

$$A = \begin{pmatrix} 1 & \frac{1}{8} & 2 & 6 \\ 8 & 1 & 7 & 9 \\ \frac{1}{2} & \frac{1}{7} & 1 & 2 \\ \frac{1}{6} & \frac{1}{9} & \frac{1}{2} & 1 \end{pmatrix}$$

Where $CR = CR(4, 9) = 9.47$ and $CM = 0.8125$. Analysis of $GD \leq 1$, $GD \leq 2$, and $CR \leq 10\%$ is shown in Table 2.4 [5].

Although Saaty's consistency ratio and Koczkodaj's consistency index are relevant, they have drawbacks that still need to be investigated. For Saaty's, "What is the relation between an empirical matrix from human judgments and a randomly generated one? Is an index obtained from several hundreds of randomly generated matrices the right reference point for determining the level of inconsistency of pairwise comparison matrix built up from human decisions, for a real decision problem?"

Table 2.4: Average value of λ_{max} of randomly generated pairwise comparison matrices, RI_n , the number of matrices with $CR \leq 10\%$, $GD \leq 1$ and $GD \leq 2$ [5]

n	Sample Size	N. Matrices $CR \leq 10\%$	Ratio of Matrices $CR \leq 10\%$	N. Matrices $GD \leq 1$	Ratio of Matrices $GD \leq 1$	N. Matrices $GD \leq 2$	Ratio of Matrices $GD \leq 2$
3	10^7	2.08×10^6	20.8%	1.42×10^6	14.2%	2.08×10^6	26.8%
4	10^7	3.15×10^5	3.15%	2.76×10^4	0.276%	1.7×10^4	1.7%
5	10^7	2.39×10^4	0.239%	61	0.00061%	2404	0.024%
6	10^7	770	0.0077%	0	0%	13	0.00013%
7	10^7	9	0.00009%	0	0%	0	0%
8	10^7	0	0%	0	0%	0	0%
9	10^7	0	0%	0	0%	0	0%
10	10^7	0	0%	0	0%	0	0%

How to take the size of matrices into account in a more precise form?”. For Koczkodaj consistency index, “the elaboration of the thresholds in higher dimensions or to replace the index by a refined grade off rule” [5]. The method of pairwise comparisons is a very important concept and can be applied to many fields. However, additional research on pairwise comparison inconsistency should be performed.

3 Description of A Heuristic for Placated Random Shape Generation

This chapter explains how random but visually nice shapes can be generated, which is often needed for cognitive experiments and processes. A nice shape is a shape with no sharp edges and corners. It is a smooth looking placated shape. The construction of random polygons is used in psychological research, testing algorithms, creation of scenes for animation and interactive art. Generating random polygons in a computationally efficient manner is important, particularly in a resource limited environment such as the web browser [10]. The proposed heuristic is based on applying the Gaussian blur to randomly generated shapes. Subsequently, the threshold is set to convert pixels to black and white from different shades of gray, giving shapes which are not sharp or otherwise hard to distinguish the area (such as, a porcupine or a sun with many rays). Randomly generated placated shapes are used for testing the accuracy of cognitive processes (such as the process induced by pairwise comparisons). They can also be used in many other areas such as computer games or software testing. The heuristic algorithm for generating nice random shapes can be perceived as an ideal example of a heuristic algorithm. Not only it ignores whether or not the solution can be proven to be correct, but such proof can very likely to never be provided since no one really knows what a nice shape is. However, we can recognize nice shapes once we see them. The observer can judge how successful we were in that attempt. Generating totally random shapes is simple: generating random coordinates, placing one pixel, and keep adding other random pixels to it. We have

not done it in this project since random shapes, which are not only nice or smooth but also not too difficult to estimate their areas, is needed for the Monte Carlo testing of the pairwise comparisons method accuracy. A 1D case (randomly generated bars) for testing the accuracy of pairwise comparisons was published in [34] and [35] as the first statistically correct in the world. The random bar length estimation error was reduced from approximately 15 % (by direct method) to approximately 5 % by using pairwise comparisons method. Evidently, it was not simple to find a solution to a 2D case, since random but nice shapes needed to be generated. A diabolical experiment was designed and published in [1] where random but equal in area shapes were used. However, the shapes, although random were a product of a human hand (as described in [39]) where a combination of rough pencil and brush strokes were used for Gaussian blurring. Respondents were tricked in comparing random but equal shapes (according to the area) without knowing that all shapes had an identical area, by a cognitive experiment designed and reported in [1]. The results of the cognitive experiment were rather astonishing. Only few respondents guessed the equality, while the estimation of one shape to measure up to 10 times bigger than the other shape was rather frequent. We need random shapes but these random shapes cannot be too tricky to estimate. For example, a porcupine or a fairy-tale sun with many rays are not easy for the area estimation purpose and must be excluded from this experiment. On the other hand, random shapes cannot be trivial to estimate, for example, randomly generated rectangles should not be used because of the proportionality to one dimension. Circles are more valid but are still proportional to the square of the diameter. Intuitively, it is obvious that the two at a time approach is better than the everything at once method for any kind of comparisons. The main goal of the presented experiments is to compare the accuracy of area assessments based on the pairwise comparisons method with the direct method, which is also referred to as by eye estimation. At the current stage of pairwise comparisons theory, there is no possibility of proving, or disproving, by analytical means which method is superior.

3.1 Rationale

Our shape generation algorithm begins with generating a random polygon. It can be degenerated (e.g., with holes). Several heuristics were presented for the random generation of polygons and implemented as the RPG (Random Polygon Generator) software package [66]. Although it is possible to generate nice random shapes, suited for the use in cognitive studies by the smoothing option, the RPG heuristic algorithms are considerably complex to implement. They are designed to generate polygons with specific features from arbitrary sets of prescribed vertices. We present a more efficient approach by assuming that there is no need to compute intermediate strict polygons if one blurs the results to obtain smooth shapes at the end, since blurring and cutting at a randomly chosen threshold is needed to obtain smooth shapes. In particular, there is no need to enforce topological shape constraints. So, we make no effort to ensure that the polygons are simply connected in the sense of topology. This property could be lost after blurring. In our approach, connectedness follows from generating a closed polygonal curve from the seed points. Moreover, blurring and then thresholding with a sufficiently permissive cut-off maintains connectedness. Obviously, the results depend on the number of points, N , and how we generate random coordinates. The method used to generate a curve from the random coordinates, the blur radius, and the threshold value also influence the appearance of shapes. In principle, any interpolatory or smoothing curve generation method such as: B-splines, Bezier, or Hermite could be used to generate the curve from the random points. We use the simplest linear spline interpolation, or “join the dots with straight lines.” The curve should not be too thin otherwise the final thresholding may introduce aliasing artifacts when the blurring is applied. Blurring gives a smoother intensity surface, which in terms lead to a smoother shape post-thresholding. The number of randomly generated points, N , should be large enough so that the polygonal curve is non-trivial. We recommend using at least ten points, unless N itself is a random variable. By assigning lower values for N , very simple shapes may be generated. On the other

hand, N should not be so large. This can result random variations that make random coordinates averaged out by the blurring process, resulting in a featureless blob. A Java random numbers function has been used for generating point coordinates by rescaling the random number. The points are uniformly distributed in a square. It has been done in such way that points fit in the assumed canvas, which is $500 * 500$ pixels. Blurring adds the mild and soft look to an initial rough shape and spreads the curve inward and outward. For this reason, there should be a large enough margin all around to accommodate the shape's final bulk. Attractive for its simplicity, this approach has great potential.

3.2 Gaussian Blur

3.2.1 Background

According to [18], increase gain of interest in image processing arises to improve the graphic information for human interpretation and the processing of scene data for machine perception. Image processing was needed in the early days in the newspaper industry. Pictures were sent by submarine cable between London and New York for the first time in the 1921. It shortened the time usually needed to transmit the photo, which was a week, to less than 3 hours. The system that was used to transmit the picture is called the Bartlane cable picture transmission system. Special printing equipment coded the picture for cable transmission and then recoded it at the receiving end for display. The first images to be transmitted using this method is shown in Figure 3.1. Thereafter, an improved technique was invented in 1922. It was based on photographic reproduction made from tapes that contained holes at the telegraph receiving terminal, which produced a better image (see Figure 3.2). Even though the transmission of these two images was related to digital images, the images were not considered as a digital image processing result because the process did not include the use of any computer. Therefore, the history of digital image processing is dependent



Figure 3.1: The first images to be transmitted using cable picture transmission system[18]

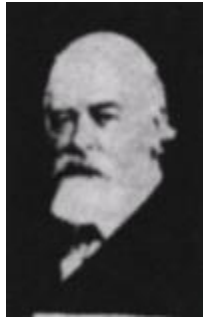


Figure 3.2: An image sent in 1922 using the improved system [18]

on the development of computers. In the 1960s, the first computer, powerful enough to process an image, was born and which is when the birth of image processing took place. During that period of time, image processing began to improve in a faster manner. In 1964, pictures of the moon transmitted by Ranger 7 were processed by a computer at the Jet Propulsion Laboratory in Pasadena, CA. The image shown in Figure 3.3 was processed to cleanup various types of image distortions. It was the first picture of the moon that was taken by a US spacecraft.

Research in image processing has grown rapidly and is continuing to grow because of the importance of that field. During the last 50 years, various techniques have been introduced and developed in image processing. Nowadays, image processing systems are becoming more and more popular due to the advances made in computer processors, memory, and graphics software. Image processing is used in many profession fields in life such as remote sensing, bioinformatics, medical imaging, forensic science, textiles, material science, military, film industry, document processing, printing in-



Figure 3.3: The first picture of the moon taken by Ranger 7 on July 31, 1964 [18]

dustry, and many more. There are many image processing techniques and operations which can be used such as image scaling, image rotation, contrast stretching, image analysis, image segmentation, image restoration, image compression, and image filtering.

Consistent with [19], an image can be thought of as a two-dimensional light intensity function $f(x, y)$. Where $f(x, y)$ gives the intensity at position (x, y) . This point is proportional to brightness or gray level of the image at that point. $f(x, y)$ is sampled so that the resulting digital image has m rows and n columns. The values of the (x, y) become discrete quantities and each sample has to be quantized or rounded to the nearest integer. The result will be a matrix of real numbers. Each element of this matrix array is called a pixel, which allows to represent an image in a matrix. In image processing, an operation defines a new image g in terms of an original image f .

$$g(x, y) = t(f(x, y)) \quad (3.1)$$

For instance, image filtering can be named as one of the many image processing techniques and operations. A filter h can change a range of an image or changes the image completely.

$$g(x) = h(f(x)) \tag{3.2}$$

Gaussian blur is an image filtering technique.

3.2.1.1 Convolution

As Gaussian blur is a type of image filter and convolution is used for applying a filter on an image, the convolution should be described first before explaining Gaussian blur. Convolution is an operation performed on two functions and produces a third new modified version. It is the treatment of a matrix by another one, which is called a kernel. Many filters uses convolution as the method to apply the filter to the image. One can build any custom filter that suit one's needs.

When a 3 x 3 kernel matrix is used, the filter successively goes over every pixel of the image. It multiplies the value of this pixel and values of the 8 surrounding pixels with the kernel corresponding values. Then it adds the results, and the initial pixel is set to this final result value. Here is an example [17] where the pixel, which the calculations are performed on, is the one at the center of the matrix. The kernel action area, which is called the neighborhood, is the whole matrix since both the kernel and the original matrices are of the same size (3 * 3).

$$\begin{pmatrix} 40 & 42 & 46 \\ 46 & 50 & 55 \\ 52 & 56 & 58 \end{pmatrix},$$

the kernel matrix is

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

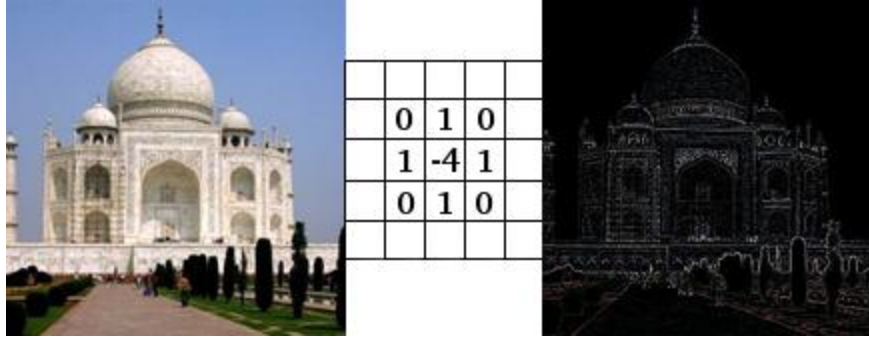


Figure 3.4: An edge detection example[17].

and the convolution result for the center pixel is represented here:

$$\begin{pmatrix} ? & ? & ? \\ ? & 42 & ? \\ ? & ? & ? \end{pmatrix}$$

The filter reads successively all the pixels of the kernel action area. It then multiplies the value of each of them by the kernel corresponding value and adds results. The result will be

$$(40*0) + (42*1) + (46*0) + (46*0) + (50*0) + (55*0) + (52*0) + (56*0) + (58*0) = 42.$$

It does the same for each other pixel.

Convolution is widely used in image processing and is used for many operations. The following filters are some examples. The input image, kernel matrix, and the output image are shown. The following convolution examples are taken from [17]. Figure 3.4 shows how convolution is used to detect the edges of an image. Another example, Figure 3.5, shows how to sharpen an image. Also, Figure 3.6 demonstrates how to blur an image.

The 2D convolution process requires $M*N$ multiplications for a kernel of size $M*N$. If the kernel size is $3*3$, then 9 multiplications and additions are needed which make the process expensive for larger size kernels. Consequently, a more efficient way to perform the convolution is needed. According to [2], it is known that these types of

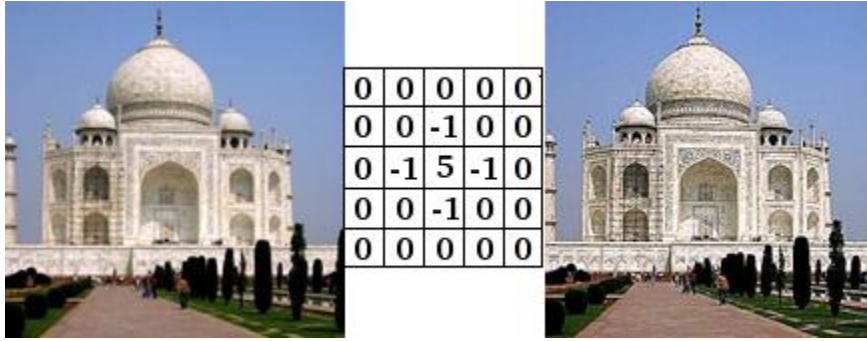


Figure 3.5: Image sharpening example [17].



Figure 3.6: Image blurring example [17].

kernels are separable, which can reduce the computations to $M + N$ multiplications and make the cost less expensive. This process is called separable 2D convolution. A separable matrix is one that can be decomposed into $M \times 1$ and $1 \times N$ matrices. For example,

$$\begin{pmatrix} A * a & A * b & A * c \\ B * a & B * b & B * c \\ C * a & C * b & C * c \end{pmatrix} = \begin{pmatrix} A \\ B \\ C \end{pmatrix} * (a \ b \ c)$$

is a separable kernel and convolution calculation looks like the following,

$$\begin{aligned} x[m, n] * \begin{pmatrix} A * a & A * b & A * c \\ B * a & B * b & B * c \\ C * a & C * b & C * c \end{pmatrix} &= x[m, n] \left(\begin{pmatrix} A \\ B \\ C \end{pmatrix} * (a \ b \ c) \right) \\ &= \left(x[m, n] * \begin{pmatrix} A \\ B \\ C \end{pmatrix} \right) * (a \ b \ c) \end{aligned}$$

Where $x[m, n]$ is the matrix element at position $[m, n]$.

The next example shows the two different ways of calculating the convolution by using both the regular convolution and the separable convolution. It shows that they yield the same answer

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

and the separable kernel

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}.$$

Then by using the regular 2D convolution shown earlier, the result the pixel at position $[1, 1]$ would be

$$y[1, 1] = 1 * 1 + 2 * 2 + 3 * 1 + 4 * 2 + 5 * 4 + 6 * 2 + 7 * 1 + 8 * 2 + 9 * 1 = 80.$$

When using the separable convolution, the first step is to apply the 1D vertical convolution where the rows $n = 1$, and the columns $m = 3$.

$$y[*, 1] = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 * 1 + 4 * 2 + 7 * 1 & 2 * 1 + 5 * 2 + 8 * 1 & 3 * 1 + 6 * 2 + 9 * 1 \end{bmatrix}$$

$$= \begin{bmatrix} 16 & 20 & 24 \end{bmatrix}$$

Then, apply the second step with the horizontal matrix

$$y[1, 1] = [16 \ 20 \ 24] * [1 \ 2 \ 1] = 16 * 1 + 20 * 2 + 24 * 1 = 80.$$

For this example, the reduction in the number of flops can not be seen but it surely would be apparent for larger sized kernels. The problem with the 2D separable convolution is that it requires more storage. A buffer is needed in order to keep the intermediate computation. For an uncompressed raw, 8-bit (unsigned char) gray scale image with a 5x5 Gaussian Kernel on a (AMD 64 3200+ 2GHz) system, normal convolution takes about 10.3 ms and separable convolution takes only 3.2 ms [2]. Separable convolution is faster compared to normal convolution. A convolution with a separable 15 x 15 kernel requires only 13 percent of the computation needed for when a non-separable kernel is used. According to [13], the improvement in performance achieved, when the kernel is separable, is significant. A 15 x 15 kernel takes about 8 seconds when it is not separable. On the other hand, it takes only about 1 second when it is separable. The Gaussian kernel is a classic example of a separable kernel. For any 2D discrete signal, convolution is defined as [13]

$$I' = I \otimes H \tag{3.3}$$

$$I'(u, v) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(u - i, v - j) \cdot H(i, j) \quad (3.4)$$

Where I is the original matrix, H is the filter kernel, and I' is the new matrix after the convolution. The \otimes is the convolution symbol. Now that the convolution equation is defined, its mathematical properties can be summarized as follow:

- Commutativity

$$I \otimes H = H \otimes I \quad (3.5)$$

- Linearity

$$(s.I) \otimes H = I \otimes (s.H) = s.(I \otimes H) \quad (3.6)$$

$$(I_1 + I_2) \otimes H = (I_1 \otimes H) + (I_2 \otimes H) \quad (3.7)$$

but

$$(b + I) \otimes H \neq b + (I \otimes H)$$

- Separability

$$H = H_1 \otimes H_2 \otimes \cdots * H_n \quad (3.8)$$

$$\begin{aligned}
H * I &= I \otimes (H_1 \otimes H_2 \otimes \cdots * H_n) \\
&= (\dots ((I \otimes H_1) \otimes H_2) \otimes \cdots * H_n)
\end{aligned}$$

An additional factor to mention is that the convolution is an associative operation so convolving in a horizontal direction first, then a vertical direction later will show the same result.

Convolution is significantly important in image processing. It is so important that specialized hardware exists to perform it in real time. In addition, high-performance parallel computing can be used to convolve images quicker as images can be split into small pieces and then each small piece can be assigned to a processor. Consequently, the convolution calculations can be performed faster in parallel way [31].

3.2.1.2 Low Pass Filters

Convolution can be used to carry out the linear filtering of an image. The nature of the filter is determined by the choice of kernel coefficients. Many image processing operations can be done using linear filtering such as image sharpening and image blurring. An image may be thought of as sound or radio waves as they all have frequencies. Frequencies that are present in an image are referred to the changes occurring in space. That is, how fast brightness or color changes as an image is traversed. A low pass filter allows low spatial frequencies to pass unchanged but not the high frequencies. The low pass filters are used for smoothing, averaging, or blurring. This reduces the noise but also make fine details unclear. Any convolution kernel that has positive values in all its elements can act similar to a low pass filter [13]. Here is a 3 x 3 low pass an example of a kernel.

$$\begin{bmatrix}
0.111 & 0.111 & 0.111 \\
0.111 & 0.111 & 0.111 \\
0.111 & 0.111 & 0.111
\end{bmatrix}$$

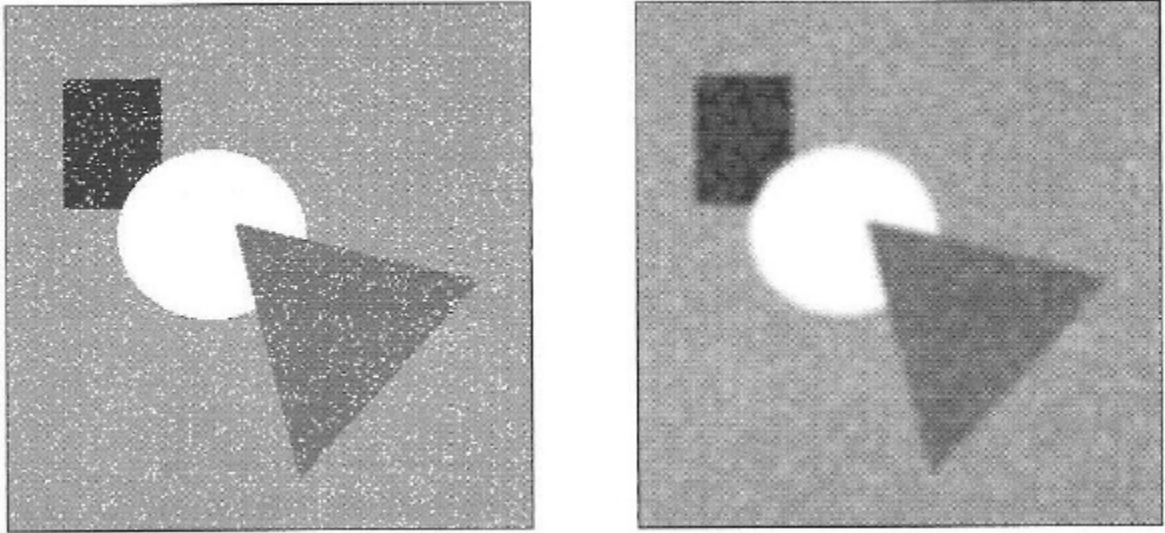


Figure 3.7: Applying $5 * 5$ mean low pass filter [13].

For simplicity, this is equal to

$$1/9 \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

This type of filter is called a mean or average filter. Values from the neighborhood are added together without being weighted, the number of pixels in the neighborhood then divides the sum. This is why it is called an average filter. By applying a mean filter, noise can be reduced which can make some images more clear. Figure 3.7 is an example that shows how useful is the mean filter. The image on the right side is the result of applying the filter. All the noise from the left image is now absent, however the fine details are less apparent. Using this type of filters is effective in many situations.

3.2.2 Gaussian Blur Low Pass Filter

The Gaussian kernel is named after Carl Friedrich Gaussian, a brilliant German mathematician who lived between the years 1777 and 1855. To show his important, his a picture shown along with the Gaussian kernel on a German banknote. These

banknotes are not in use nowadays as the Euro has replaced them [16]. Gaussian filters are a class of linear smoothing filters, with the weights chosen according to the shape of a Gaussian function. The Gaussian smoothing filter is a very favourable filter for removing noise and is drawn from the normal distribution. As shown in the previous section, blurring can be done using a uniform kernel that has equal coefficients. Also, non-uniform kernels may be used instead. Gaussian blur is a non-uniform kernel. It is similar to the mean filter, but it uses a different kernel. It represents the shape of a Gaussian bell shaped curve. Gaussian blur works very similarly to any other smoothing kernel, but the Gaussian blur produces a more natural looking blur. It is one of the most often used smoothing techniques.

Designing a Gaussian filter can be done using two ways. The first method is to use the coefficient of the binomial expansion to get an approximation to a Gaussian kernel [70] [27]

$$(1 + x)^n = \begin{bmatrix} n \\ 0 \end{bmatrix} + \begin{bmatrix} n \\ 1 \end{bmatrix} .x + \begin{bmatrix} n \\ 2 \end{bmatrix} .x^2 + \dots + \begin{bmatrix} n \\ n \end{bmatrix} .x^n, \quad (3.9)$$

which is the same as using the Pascals triangle [70] shown in Table 3.1.

Row n of Pascal's triangle is a one-dimensional approximation to a Gaussian filter. The filter will have n values. So, the five-point approximation is

$$\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix}.$$

This is the 5th row coefficient of the Pascals triangle. This Gaussian filter may be used to blur an image in the horizontal direction. If 2D Gaussian is needed, then

Table 3.1: Pascal's Triangle

Index N	Coefficients					Sum of coefficients = 2^N
0					1	1
1			1	1		2
2		1	2	1		4
3		1	3	3	1	8
4		1	4	6	4	16
5		1	5	10	10	32
6		1	6	15	20	64
7	1	7	21	35	35	128
8	1	8	28	56	70	256
9	1	9	36	84	126	512
10	1	10	45	120	210	1024
11	1	11	55	165	330	2048

another vertical convolution using

$$\begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix}$$

is needed. A convolution with the multiple of the previous two matrices will also give the same result but it is more expensive.

$$\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Performing the calculation for the 5 x 5 matrix is more expensive than the calculation of the 1D filter that is done vertically and then horizontally.

The second approach used to design the Gaussian kernel is to compute the kernel weights directly from the discrete Gaussian distribution functions [27]. The 1D Gaussian function is

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \quad (3.10)$$

σ is the standard deviation. When $\sigma = 1$ and the *mean* = 0. The distribution looks like Figure 3.8.

The 2 dimensions Gaussian formula is:

$$g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}. \quad (3.11)$$

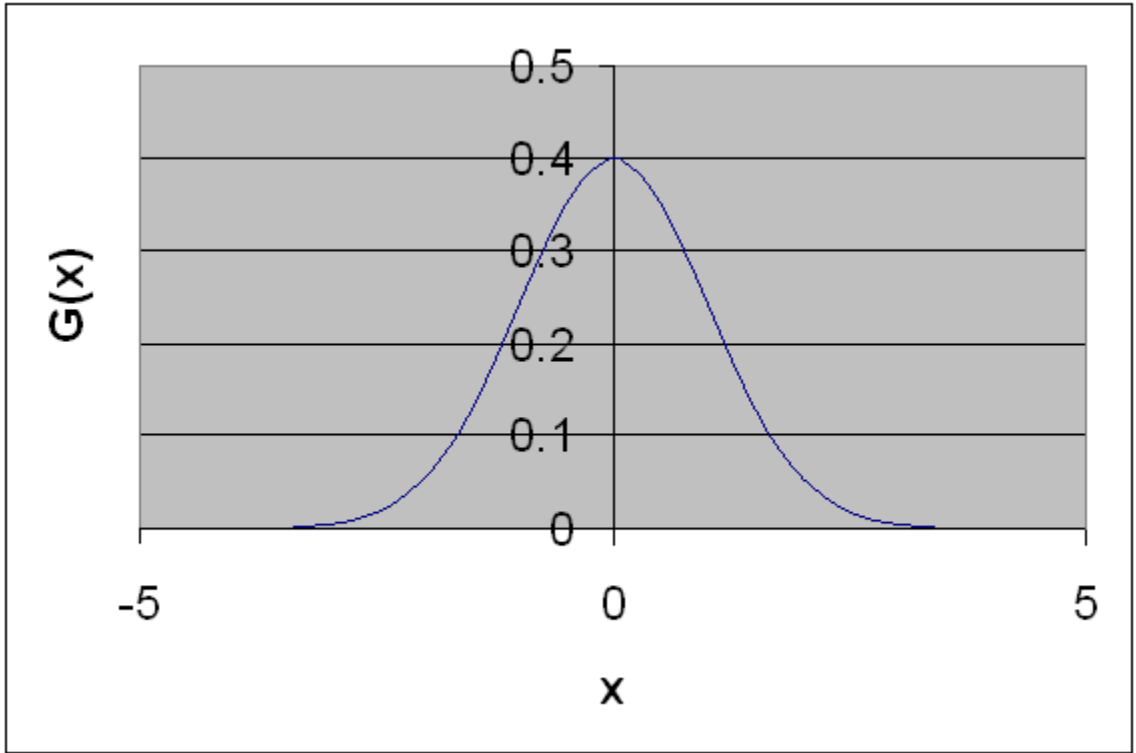


Figure 3.8: A 1D Gaussian distribution

Where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution. The following is a construction of a 3×3 Gaussian filter using the above formula. When $\sigma = 1$, the filter is as follows:

$$\begin{bmatrix} 0.0585498 & 0.0965324 & 0.0585498 \\ 0.0965324 & 0.159155 & 0.0965324 \\ 0.0585498 & 0.0965324 & 0.0585498 \end{bmatrix}$$

$$G(-1, -1) = 0.0585498, G(0, -1) = 0.0965324, G(1, -1) = 0.0585498,$$

$$G(-1, 0) = 0.0965324, G(0, 0) = 0.159155, G(1, 0) = 0.0965324,$$

$$G(-1, 1) = 0.0585498, G(0, 1) = 0.0965324, G(1, 1) = 0.0585498$$

$G(0,0)$ is the origin located in the center of the kernel. The values get diminish as the distance from the origin increases. Eventually, it will reach 0 when the distance from the centre is near 3σ Also, it can be noticed that many values are the identical

Table 3.2: A commonly used 15 x 15 Gaussian filter

2	2	3	4	5	5	6	6	6	5	5	4	3	2	2
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
3	4	6	7	9	10	10	11	10	10	9	7	6	4	4
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
6	8	11	13	16	18	19	20	19	18	16	13	11	8	6
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
2	2	3	4	5	5	6	6	6	5	5	4	3	2	2

because the Gaussian blur is circularly symmetric. Now, one can convolve an image with the Gaussian kernel in order to get the blurred smoothed image. Table 3.2 is an example of a commonly used 15 x 15 Gaussian filter with a noticeable distribution of the values [27]. The 2D Gaussian functions are rotationally symmetric. Therefore, all lines and edges in all directions are treated similarly. Gaussian kernels smooth by replacing each image pixel with a weighted average of the surrounding pixels and the weight decreases as the distance from the center increases. In addition, the Fourier transform of a Gaussian has a single lobe in the frequency spectrum and the Fourier transform of a Gaussian is itself a Gaussian. The degree of smoothing of Gaussian filter is σ and it is related to the wide of the Gaussian. When σ increases, the Gaussian kernel widens and the image gets smoother. In addition, convolution of a

Gaussian with itself is another Gaussian. This means that one can first smooth an image with a small Gaussian, and then convolve that smoothed image with another small Gaussian. The result is equivalent to smoothing the original image with a larger Gaussian. This allows large Gaussian kernels to be implemented very efficiently. Also, Gaussian functions are separable as mentioned earlier. By smoothing an image with a Gaussian blur, we make it “nicer”. In essence, each pixel is mapped into a weighted average of that pixel’s neighborhood. The highest Gaussian value (weight) is given to the original pixel while the neighboring pixels receive smaller weights since their distance to the original pixel increases.

3.2.2.1 Example

Here is an illustration example using a program, written in Java that applies a Gaussian blur to a randomly created shape. The Gaussian function part of the program is according to [7]. The kernel used for convolution is shown in the following.

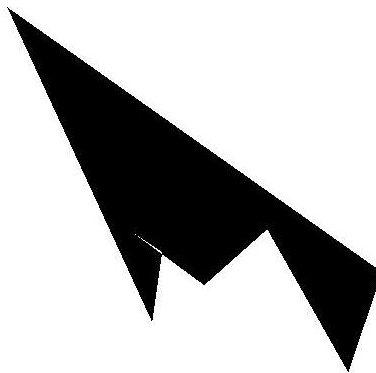


Figure 3.9: The shape before applying Gaussian blur

$$\begin{bmatrix} 0.0009 \\ 0.0016 \\ 0.0027 \\ 0.0045 \\ 0.0071 \\ 0.0108 \\ 0.0158 \\ 0.0222 \\ 0.0300 \\ 0.0389 \\ 0.0485 \\ 0.0581 \\ 0.0668 \\ 0.0739 \end{bmatrix} X \begin{bmatrix} 0.0009 \\ 0.0016 \\ 0.0027 \\ 0.0045 \\ 0.0071 \\ 0.0108 \\ 0.0158 \\ 0.0222 \\ 0.0300 \\ 0.0389 \\ 0.0485 \\ 0.0581 \\ 0.0668 \\ 0.0739 \end{bmatrix}^T$$

After applying this Gaussian blur kernel to Figure 3.9, the result smoothness image is apparent in Figure 3.10.



Figure 3.10: The shape after applying Gaussian blur

3.2.2.2 A More Efficient Gaussian Blur

When using larger size kernels, the execution time can be very long. Proper use of the Gaussian blur properties, as shown above, can help to reduce the cost of the computations. Waltza and Millerb suggested an even more efficient way for applying Gaussian blur [70]. They use two Gaussian blur properties in their more efficient method. The first property is that the Gaussian blur large kernel can be decomposed into the sequential application of small kernels. In addition, the Gaussian kernel is separable into row and column operations. The paper also suggests that the row and column operations can be formulated as finite-state machines (FSMs) to produce highly efficient code. It shows that it is necessary to write results to an intermediate images and then fetches these results for the next operation. The algorithm uses SKIPSM [69]. SKIPSM is a new way to carry out many standard image processing operations. In comparison with conventional hardware-based and software-based approaches, SKIPSM allows implementation at higher speeds and lower hardware cost. In SKIPSM, a large class of neighborhood image processing operations, which are generally considered not to be separable, is separated into row operation followed by a column operation. Also, the formulation of these row and column operations is compatible with pipelined operations. The implementation of the resulting operations is a simple finite-state machines. Here is a simple example [70] to illustrate the

efficient algorithm for Gaussian blur using finite-state machines. As a starting point, consider 2 x 2 kernel

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

By using the SKIPSM paradigm, the implementation requires one memory location for the state of the row machine $SR0$ and a column state buffer $SC0[i], i = 1, \dots, n$, where n is the size of the row. The column state buffer is set to zero at the start of the overall operation, and the row state buffer is set to zero at the start of each row. For each pixel, the main loop is executed once. The output is then written back onto the input image. Here is what the loop should look like along with the SKIPSM paradigm [70].

```
tmp1 = input[i][j]; // input pixel
tmp2 = SR0 +tmp1; //from row machine output
SR0 = tmp1; //update the row state buffer
output[i][j] = (2+SC0[i] + tmp)/4; //form and scale output
SC0[i] = tmp2; //update the column state buffer.
```

This 2 x 2 SKIPSM implementation requires five steps. The brute force algorithm implementation would use nine. Here, the SKIPSM result is not very impressive. The advantage increases as the size increases. To make things even more interesting, here is a 3 x 3 blur example.

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Here, $SR0$ and $SR1$ are needed. Also, column state buffers, $SC0[i]$ and $SC1[i]$. The output is written to $[j - 1][i - 1]$ because it is the center of the 3 x 3 neighbourhood [70].

```
//row machine
tmp1 = input[i][j]; // input pixel
```


Table 3.3: Comparison between the 3 different algorithms

Algorithm	Time
standard convolution	919 ms
separable convolution	450 ms
SKIPSM	108 ms

```

tmp2 = SR0 +tmp1; //from intermediate value
SR0 = tmp1; //update first row state buffer
tmp1 = SR1 + tmp 2; // new row machine output.
SR1 = tmp2; // update 2nd row state buffer
// column machine
tmp2= SC0[i] + tmp1;
SC0[i] = tmp1;
output[j-1][i-1] = (8+SC1[i]+tmp2))/16;
SC1[i] = tmp2;

```

Here, the calculation requires only 9 steps. The brute force approach for this case requires 24 steps. These are 9 fetches, 5 multiplications, 8 additions, one scaling step, and a write to the output image. The different approaches were all tested on a PC-compatible computer using a 166 MHz AMD K6 CPU with 512 KB of secondary cache and 64 MB of memory under Windows 95. The test was done using a $5 * 5$ kernel and the results is shown in Table 3.3 [70].

The best results were obtained using the SKIPSM. It is nine times faster than the standard convolution. The speed advantages of the SKIPSM implementation should be even greater for larger kernels.

3.3 The random shape heuristic algorithm

It is a natural temptation to draw a swirling line or a curve when attempting to generate a random but nice shape. However, designing an algorithm or a heuristic for such random line is not simple. Certainly, one can go left, right, up, or down randomly and can also choose random number of steps but such line will be very rigid and may cross itself many times. Table 3.4 shows a heuristic algorithm easy to implement.

Table 3.4: The placated random shape generation heuristic algorithm

Step 1 Generate random polygon.

Step 2 Apply Gaussian blur with randomly chosen parameter.

Step 3 Select random threshold to turning grey pixels to black

3.3.1 Generate Random Polygon

Java was chosen for implementation since Java applications are typically compiled to byte code (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. It was important since the results of this project was used for another project related to pairwise comparisons and data gathering took place over the Internet.

A flat shape consisting of straight and non-intersecting sides that are joined to form closed path is called a simple polygon. If the sides intersect then the polygon is not

simple and is just called a polygon. By definition [20], the polygon boundary may or may not be part of the polygon itself. Whether the boundary is included or not, it does not effect our study. For the scope of this experiment, creating a polygon or a simple polygon are both valid. Therefore, shapes do not have to be simple polygons. To create a random polygon, the number of points needed for the polygon is randomly generated. A random number of points between 7 and 90 was used. This range is chosen because shapes with lower number of points can be very simple. On the other hand, shapes with very high number of points can be very complex and may have many wholes that the Gaussian blur may not overcome. Java's utility `Math.random()` function is used to generate the number of points to be used. The function returns a double value with a positive sign, ≥ 0.00 and < 1.00 . Returned values are chosen randomly with a uniform distribution from that range [52].

```
private int randomNumberOfPoints = numberOfPoints(min, max);
```

`numberOfPoints` takes the minimum and maximum number of points n and returns a number between the `min` and the `max` inclusively. This gives the ability to specify a minimum and a maximum bounds as desired. `Min + (int)(Math.random() * ((Max - Min) + 1))` is a trick used along with the random function to generate a number in between that range. It is used due to the limitation that the random function has since it only generate a number that is ≥ 0.00 and < 1.00 . After that the `RandomPoint` function is used to randomly create n points. After that, an array of type `Point` is generated.

```
Point[] points = new Point[randomNumberOfPoints];
```

Next, `RandomPoint(randomNumberOfPoints, x)` function is called. It takes both the number of points and the image size $x = 500$ where $imagesize = height * width = x * x$. The images used in this research all of the size of $500 * 500$. We took into consideration the image size so we can accommodate all part of the image after Gaussian blur is applied. Therefore, there is a 50 margin by the image border. To draw lines

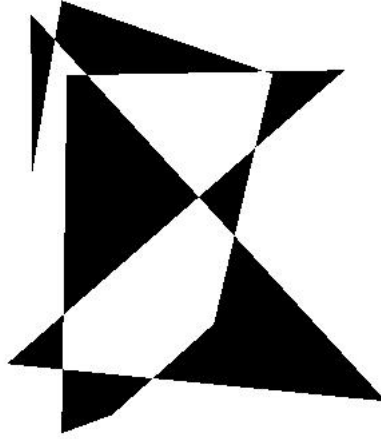


Figure 3.11: A 500 * 500 image with 11 points shape

connecting the randomly generated lines, `g.fillPolygon(x, y, points.length)` is used [50]. `fillPolygon` method fills a closed polygon defined by arrays of `x` and `y` coordinates. The third argument is the number of points that makes the polygon. It draws a line connecting the final point to the first point which automatically closes the figure. Figure 3.11 is a 500 * 500 image that uses the following 11 randomly generated points (x, y) . $(342, 387), (76, 97), (77, 216), (99, 87), (257, 142), (213, 329), (137, 397), (99, 411), (103, 143), (311, 139), (58, 359)$.

3.3.2 Applying the blur and the cut-off

In this subsection, applying Gaussian blur on a randomly generated shape and the threshold cut-off are explained. To apply Gaussian blur, we need to generate the Gaussian kernel using the Gaussian function first.

$$g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (3.12)$$

with the help of `Kernel` class and by using the following method.

```
public static Kernel makeKernel(radius)
```

The `Kernel` class defines a matrix that describes how a specified pixel and its surrounding pixels affect the value computed for the pixel's position in the output image of a filtering operation. [51]. `makeKernel` [25] takes the value of the radius and returns the kernel. The function code used is shown below. See Figure 3.12 for an example of the function output.

```
public static Kernel makeKernel( radius)
{
    int size=radius*2+1;
    float data[]=new float[size];
    float sigma=radius/ 3.0f; // raduis = 3*sigma
    float twoSigmaSquare=2*sigma*sigma;
    float root=(float)Math.sqrt(twoSigmaSquare*Math.PI);
    float total=0.0f;
    for(int i=-radius;i<radius;i++)
    {
        float distance=i*i;
        int index=i+radius;
        data[index]=(float)Math.exp(-distance/twoSigmaSquare)/root;
        total+=data[index];
    }
    for(int i= 0;i<data.length;i++)
    {
        data[i]/=total;
    }
    Kernel kernel=null;
```

```
0.0015[
0.0038
0.0088
0.0181
0.0333
0.0548
0.0809
0.1068
0.1262
Gaussian radius= 9
```

Figure 3.12: An output of a kernel with 9 radius

```
if(horizontal)
{
kernel=new Kernel(size,1,data);
}else
{
kernel=new Kernel(1,size,data);
for (int i = 0; i < size/2; i++)
{
vertical vector
System.out.printf("%9.4f ", data[i]);
System.out.println();
}
}
return kernel;
}
```

Applying the blur was done with the help of the `ConvolveOp` [49] class. This class operates with `BufferedImage`. It implements a convolution from the source to the destination. Convolution using a convolution kernel is a spatial operation that computes the output pixel from an input pixel by multiplying the kernel with the surround

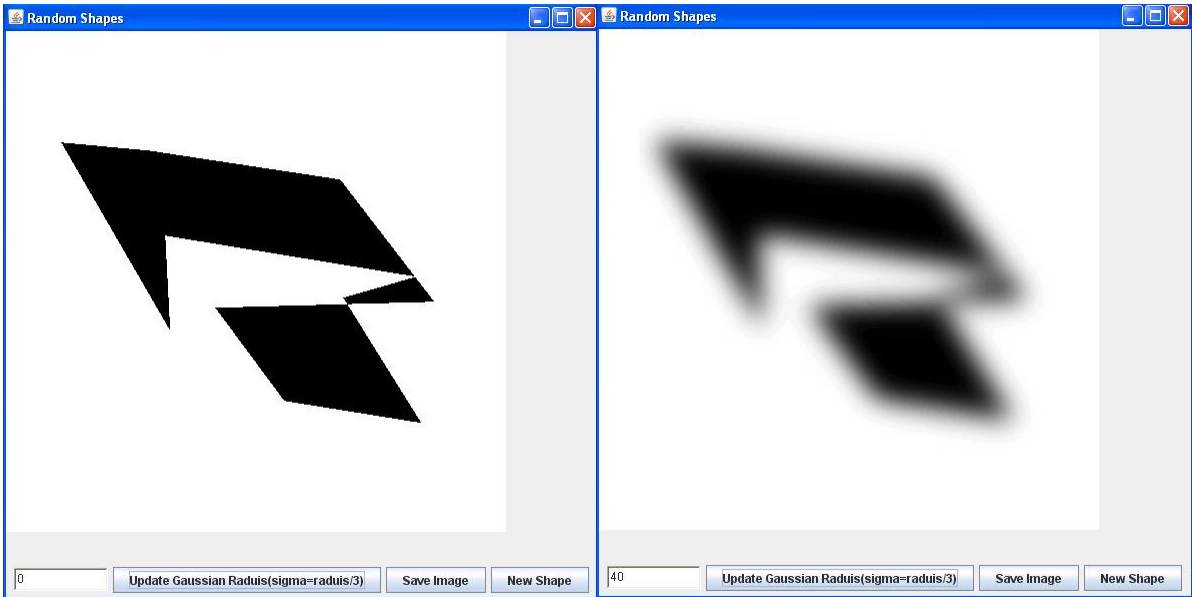


Figure 3.13: Blurring a random shape example

of the input pixel. This allows the output pixel to be affected by the immediate neighborhood in a way that can be mathematically specified with a kernel that was created earlier [49]. See Figure 3.13 for an example output.

```
ConvolveOp(kernel, ConvolveOp.EDGE_NO_OP, null);
```

The next step is to apply the cut-off by using the `ReplacePixel(blurredImage, threshold)` function. This function takes an array that represents the blurred image and a threshold. It returns the new cut-off image by going through each and every pixel in an image and checking its value. Pixels that are below the threshold value are replaced by the value of 0 or black. On the other hand, pixels that are greater than or equal than the threshold are switched to 255 or white. This will yield the desired smooth looking shape. Simultaneously, this function will also calculate the area of the shape by counting non white pixel.

```
public int[][] ReplacePixel(int[][] BlurredImage, int Threshold)
{
```

```

for(int i = 0 ; i < width; i++)
{
for(int j = 0 ; j < height ; j++)
{
if (newImage[i][j] >= Threshold)
{
newImage[i][j] = 255;
}
if (newImage[i][j] < Threshold )
{
newImage[i][j] = 0;
}
if(src[i][j] != 255)
{
area++;
}
}
}
}
}

```

Figure 3.14 shows what the smooth shape after the cut-off looks like. The output of calculating the area of the randomly generated, smooth looking shape is shown in Figure 3.15.

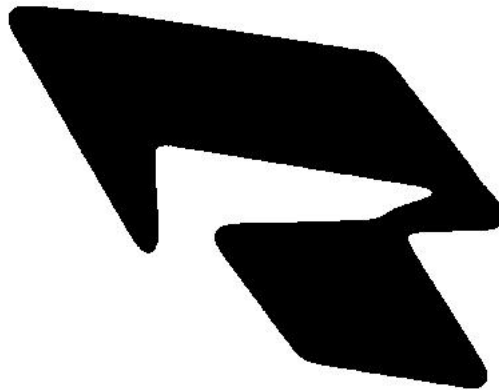


Figure 3.14: The smooth looking shape after applying the cut-off using a 200 threshold

```
<terminated> Image2Array [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (2013-06-05 10:56:53 PM)
image size: 500 x 500
That took 0 seconds to load image into the array
Threshold= 200
Area= 50073
That took 0 seconds to load from array to image file
That took 0 seconds to cut the shape and calculate the area
That took 0 seconds for TOTAL time
```

Figure 3.15: A terminal screenshot of the area of the shape output along with other information

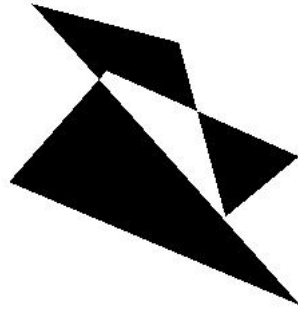


Figure 3.16: A 7 points shape



Figure 3.17: A 7 points shape with 20 Gaussian radius

3.3.3 Dealing with holes

Shapes that are randomly generated with the growing number of points N are more likely expected to have holes. To get the most desired nice looking shape with no holes, parameters adjustment should be performed. The parameters being the Gaussian blur radius and the threshold value. In the first example shown in Figure 3.16, the shape is randomly generated and constructed using 7 points.

As it can be seen, the shape does not look excessively nice. It seems to be three different but not completely connected shapes. Getting a nice smooth looking shape out of it requires extra process. First, apply the Gaussian blur of different radius values and then apply a threshold with a more appropriate value. Applying a Gaussian blur with a radius of 20 produces Figure 3.17.

In Figure 3.17, the shape is more connected now but there is still a large hole in the

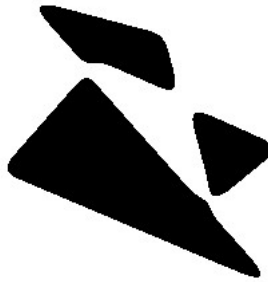


Figure 3.18: The shape after applying a 100 threshold

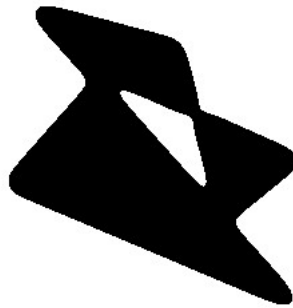


Figure 3.19: The shape after applying a 200 threshold

centrer. By applying a threshold of 100, the result can be seen in Figure 3.18.

The result in Figure 3.18 shows three separate shapes which are not what is desired. Increasing the threshold, makes the image move toward the desired shape. With the 200 threshold, Figure 3.19 shape shows that the hole is still there while a randomly generated nice looking shape with no holes is the goal that has not been reached yet.

Due to the fact that the whole is large, the threshold value should be considerably large and near the maximum to eliminate the hole. However, the threshold 254 is still not enough to remove the hole. Threshold value 255 would be acceptable and would finally remove the hole. However, the shape becomes much bigger and some part of the shape is not visible in the work area and this violates the niceness of the shape. This can be seen in Figure 3.20.

Processing the same shape with a larger Gaussian blur radius with an appropriate

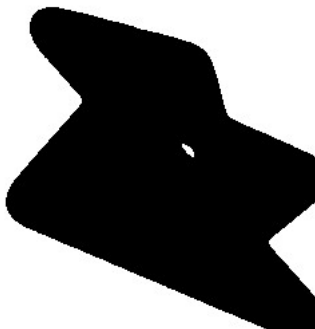


Figure 3.20: The shape after applying a 255 threshold



Figure 3.21: The shape after applying a 40 Gaussian kernel

threshold can remedy the problem. Figure 3.21 and Figure 3.22 demonstrates the processed images after applying a Gaussian radius of 40 and then applying a 127 threshold.

Figure 3.22 is all connected with no holes as desired but there are still two strange looking narrow nicks remaining. If they are not needed, a larger threshold should be used. Making the threshold reach as large as 217 is not enough as a tiny little hole is still there. The magic threshold number for this particular shape, that results the perfect shape we are looking for, is 218. Using a 40 Gaussian radius and a 218 threshold will produce the placated smooth looking shape that is randomly generated. This is shown in Figure 3.23.

In the next example, Figure 3.24, the randomly generated star looking shape also has also a large whole in the middle. It is noticeable that the area covered with black is smaller than the one in Figure 3.16 from the example shown earlier. The white whole



Figure 3.22: The shape after applying a 127 threshold



Figure 3.23: The shape after using a 40 Gaussian radius and a 218 threshold

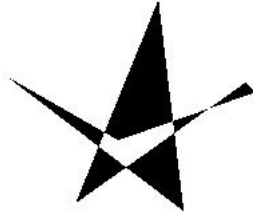


Figure 3.24: A star looking shape



Figure 3.25: A star looking shape with Gaussian blur of 10

here is smaller, which can make using Gaussian, with smaller radius, possible to get the preferred image. The value may be as low as 10, however the threshold has to be 254 for this to work. The blurred image looks like Figure 3.25 and Figure 3.26 after applying the threshold.

As it can be seen in Figure 3.26, the shape is not very nice. A larger Gaussian filter with a 35 radius will result the shape shown in Figure 3.27. Furthermore, a threshold of 240 will give Figure 3.27 which is the smooth placated looking shape that we are looking for.

In this third example, an 8 points shape with 2 holes is used (Figure 3.29). Applying a Gaussian blur filter with the value of 25, gives Figure 3.30. A 127 threshold gives the shape shown in Figure 3.31 or to be more accurate, 2 separate shapes, which is not ideal. Increasing the Gaussian blur radius to 45 and having the threshold value as 150 will give Figure 3.32 with no holes. Whenever the threshold increases, the 2



Figure 3.26: A star looking shape with Gaussian blur of 10 and a 254 threshold



Figure 3.27: A star looking shape with Gaussian blur of 35



Figure 3.28: A star looking shape with Gaussian blur of 35 and a threshold of 240

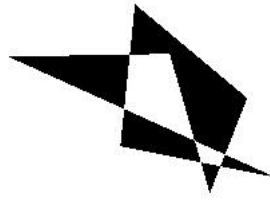


Figure 3.29: An 8 points shape



Figure 3.30: An 8 points shape and a Gaussian of size 25

“arms” of the shape come closer to each other until they meet and construct a new hole. Figure 3.33 is the result when the threshold is 157. Going towards a higher threshold, a 206 threshold, gives a placated looking shape with no holes shown in Figure 3.34.

The next example is more extreme. With too many holes in the randomly generated shape, a higher Gaussian blur radius value is needed in order to get the nice shape. Here, a 75 Gaussian radius is used with a 182 threshold and would result the shape shown in Figure 3.35.

In this last example, Figure 3.36, a 50 point random shape was generated. The number of holes here are very hard to count. By applying a Gaussian blur with a 50 radius and a 127 threshold, the right hand side and the middle shapes shown in Figure 3.36 are produced. However, a 150 Gaussian blur radius and a 127 threshold are used to get the desired nice shape as shown in Figure 3.37.



Figure 3.31: An 8 points shape and a Gaussian of size 25 and 127 threshold



Figure 3.32: An 8 points shape and a Gaussian of size 45 and 150 threshold



Figure 3.33: An 8 points shape and a Gaussian of size 45 and 157 threshold



Figure 3.34: An 8 points shape and a Gaussian of size 45 and 206 threshold



Figure 3.35: A more extreme example



Figure 3.36: A 50 points shape with 50 Gaussian radius and 127 threshold



Figure 3.37: A 50 points shape with 150 Gaussian blur radius and a 127 threshold

4 Survey

4.1 Background

Few experiments that are similar to the experiment implemented in this research were conducted in the past. When we decided to perform 2D Monte Carlo accuracy testing for the pairwise comparison method using random shapes, the main question that came to mind was: what is the best method to use? We looked at [55], [62], [1], and [34]. In [62], Saaty experimented five different in area geometric shapes. The objective was to compare the area of the shapes just by looking at them. The shapes used were a circle, a triangle, a square, a diamond, and a rectangle. They are shown in Figure 4.1. The comparisons' values are then inputted to a pairwise matrix as in Table 4.1. Notice how close the last two columns in the table are. The last two columns represent priorities derived from judgment and the real normalized relative sizes.

Table 4.1: A pairwise comparison matrix for the 5 geometric shapes [62]

Figure	Circle	Triangle	Square	Diamond	Rectangle	Priorities Eigenvector	Actual Relative Size
Circle	1	9	2	3	5	0.462	0.471
Triangle	1/9	1	1/5	1/3	1/2	0.049	0.050
Square	1/2	5	1	3/2	3	0.245	0.234
Diamond	1/3	3	2/3	1	3/2	0.151	0.149
Rectangle	1/5	2	1/3	2/3	1	0.093	0.096

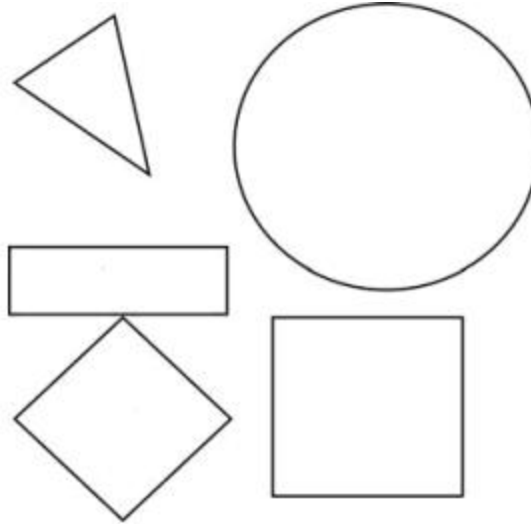


Figure 4.1: The 5 shapes to used for comparison [62]

In [1], pairwise comparisons and visual perceptions of equal in area polygons were also studied. Participants rated the areas of five randomly generated shapes of equal area. A reference unit area was also displayed along with the shapes. Respondents' average error when estimating the area using the unit square was 25.75%. Nevertheless, the error went down to 5.51% when the shapes were compared in pairs. It is a much better improvement percentage than the 1D case where bars were used [34]. The experiment demonstrated in [1] is the first statistically proven experiment for 2D shapes to improve accuracy of the pairwise comparisons method. In [1], a sample of 179 participated in the study. In the first part of that experiment, they were asked to estimate the area of 5 randomly generated shapes of equal areas in units. Of course, they were not told that the shapes were equal in area. The shapes and the unit used are shown in Figure 4.2. The shapes were presented in an overhead screen and participants took on average 10 to 15 seconds to estimate the area of each shape. In the second part, the shapes were shown in pairs. Ten pairs were shown and similarly it took 10 to 15 seconds to compare each pair. For each pair, participants were asked which shape is larger. They also had the option to respond if they believed that a pair was equal. The questionnaire is shown in Figure 4.3. Another study in [55] compared



Figure 4.2: The 5 Equal shapes with the square unit used in [1]

Your answers can be numbers with or without fractions (such as $1\frac{3}{4}$ or 2.3)

Slide number	1	2	3	4	5
Area estimation in units:					

Slide	The bigger shape is	<i>Answers: numbers with or without fractions (1 if equal)</i>
6	L = R	The larger figure is: _____ times bigger than the other figure
7	L = R	The larger figure is: _____ times bigger than the other figure
8	L = R	The larger figure is: _____ times bigger than the other figure
9	L = R	The larger figure is: _____ times bigger than the other figure
10	L = R	The larger figure is: _____ times bigger than the other figure
11	L = R	The larger figure is: _____ times bigger than the other figure
12	L = R	The larger figure is: _____ times bigger than the other figure
13	L = R	The larger figure is: _____ times bigger than the other figure
14	L = R	The larger figure is: _____ times bigger than the other figure
15	L = R	The larger figure is: _____ times bigger than the other figure

Figure 4.3: The questionnaire used in [1]

direct and pairwise area estimation of physical shapes through, not only vision, but also by physically touching the shapes. Participants were asked to estimate areas of five physical rigid shapes by viewing and touching the shapes using the direct and pairwise methods. For the direct method, participants held one shape at a time and estimated the area based on a unit square. The average error was 14.4%. However, the error decreased to 0.3% when the shapes were touched by hand and compared in pairs. That is considerably lower than the previous average error rates made, when viewing with an overhead projector. Our research replicates the experiment in [1] but this time with using shapes that were randomly generated with different area.

4.2 Preparation

Redoing [1] but with randomly generated shapes not equal in area, using the same method of data collection was not an option. Using paper questionnaires and an overhead projector has many limitations as it lacks the flexibility and the ability to show real time random shapes with different area. Therefore, we had to think of with a more sophisticated idea. We first had to generate the shapes to be used in the experiment. We used the algorithm shown in Chapter 3 to generate the shapes and calculate all the area of the shapes. For building the questionnaire, PHP is used. PHP, or Personal Homepage tool [21], is a web scripting language that works as a tool to process HTML forms and create web pages. It is a server side scripting language that interprets the PHP code at the webserver and generates HTML. PHP is the engine behind millions of dynamic web applications and can speak to many databases such as MySQL [63]. It is the most well known web platform in the world as it operates in more than 33% of the web servers across the world. Many companies trust PHP as the language to use their applications. This includes Fortune companies [21]. PHP strengths include performance, stability, built-in libraries for common web tasks, low cost, and portability. Combining the use of PHP and MySQL is very favorable because they work with almost all operating systems, web servers, and hardware [71]. After

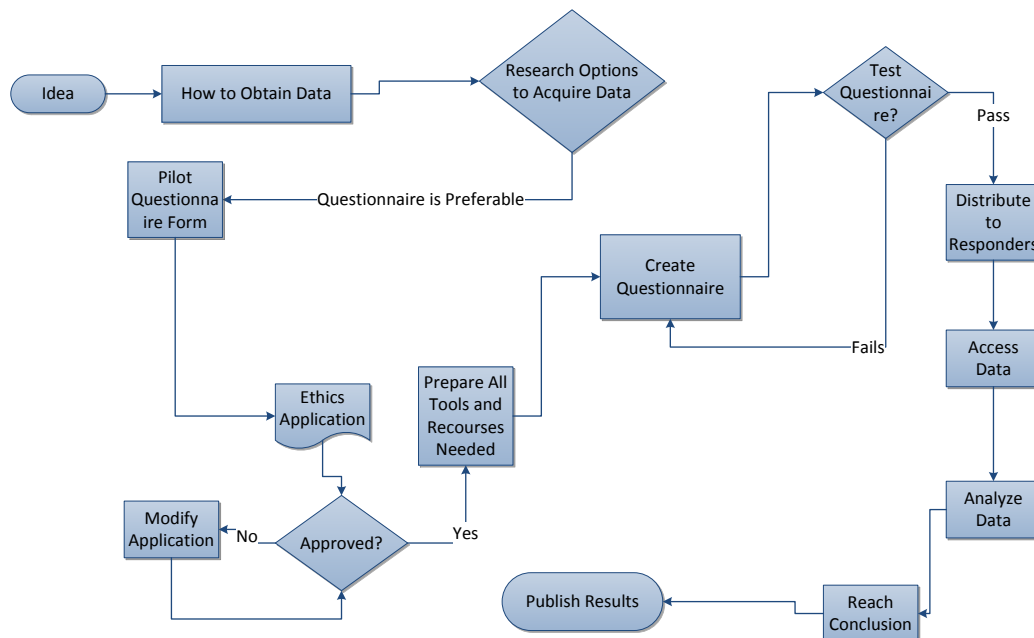


Figure 4.4: A flowchart describing the step taken to perform the survey

programming the questionnaire but before circulating the survey, an application had to be submit for the University Research Ethics Board to approve our study. The University Research Ethics Board is responsible for all research that includes human subjects conducted by faculty, students and staff at the University to make sure that the rights and the welfare of the human subjects involved are not harmed in any way. The Ethical Bored has approved study and we were able attempt the next step of the research. Every participant in the study had to read and electronically date and sign the consent form prepared with the help of the University Research Ethics Board before being able to start the survey. See Figure 4.5 for the consent form used. Figure 4.4 shoe all steps taken for the preparation of the experiments.



Consent Form

Generating random data for 2D accuracy testing of pairwise comparisons questionnaire

Please read the following in regards to your participation in the study entitled "*Generating random data for 2D accuracy testing of pairwise comparisons.*"

My name is Abdullah Almowanes. I am an MSc student in the Department of Math and Computer Science at Laurentian University. I'm conducting a research on how to compare and estimate the area of smooth looking random shapes. Benefits of this study include testing the accuracy of the pairwise comparison method. Pairwise comparison is used to compare entities in pairs to judge which of each entity is preferred as an alternative of comparing them individually. The study will also help me complete my MSc thesis. There should be no risk to you performing the tasks. If you get frustrated, you can withdraw from the experiment. The study will take approximately 10 minutes of your time and will involve filling out a questionnaire by performing few tasks that include comparing and estimating the area of some shapes. **Tasks to be performed include:** (1) Choose 5 shapes according to your liking from a set of shapes (2) Estimate the area of the shapes using a reference unit that will be displayed. (3) Compare the area of the shapes in pairs. Your participation in this study is strictly voluntary. You have the right to withdraw without completing the questionnaire at any time without any consequences. If you have any questions or concerns about the study or about being a subject, you can call my supervisor Prof. Waldemar W. Koczkodaj in his office at 705-675-1151, ext. 2311 or toll free at 1-800-461-4030. You also can contact me through my supervisor. Your identity will not be revealed at any time. The results from this study will be part of my MSc thesis. We will take all the measures to ensure anonymity of the respondents. No IPs or any personal Identification is stored. Data will be kept for at most a year after the finishing the study. Only group, not individual information will be reported and confidentiality is ensured. If you wish, you can provide an email were you can be informed with the results of the study. For questions or concerns regarding the ethical conduct of this study please contact: Research Ethics Officer, Laurentian University Research Office, telephone: 705-675-1151 ext 2436 or toll free at 1-800-461-4030 or email: ethics@laurentian.ca **By providing checking the "I agree" box and providing today's date you are giving your digital signature that you have read the above statements and freely consent to participate in this research**

Date:

Do you wish to be informed with the results of the study? (provide email if yes)

Yes Email

No

I agree

Figure 4.5: The consent form used

4.3 Experiment 1

After preparing all the steps necessary to complete the questionnaire, it was ready for distribution and we started the data collection process. The following is a detailed description of experiment 1.

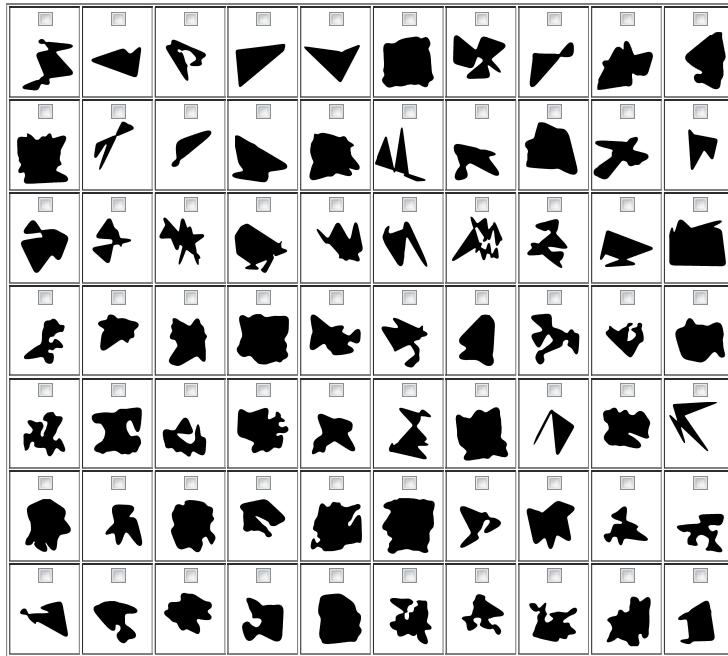
4.3.1 Sample

There were 65 participants who contributed in this study. There was no particular procedure for selecting participants. Any person could participate and no one was excluded as long as that person is an adult. Students around campus were asked to participate and we were asking them for the sake of helping with the research. Only the date, time, and participants' answers were recorded. The email was also recorded only if the participant asked for the results to be sent to them when the study will be completed. No IPs or any personal identification were stored.

4.3.2 Measures

In the first part of the experiment and only after the consent form was electronically signed, participants were asked to choose 5 shapes from a pool of 70 shapes. These shapes were previously randomly generated and have different areas. The 70 images were all of size $500 * 500$. They were rescaled to a smaller size ($63*63$) to make all 70 shapes fit the screen (see Figure 4.6). The PHP code validated the participant's responses and made sure that the user chooses only 5 shapes. If not, a visible error message was shown and the user was able to edit the shape selection to ensure that only 5 were chosen. Users were encouraged to select the shapes based on what is considered being easy for area estimation and comparisons. After the user has chosen the 5 shapes, he/she may start performing the second task. In the second part, an example will be displayed to help understand how this part should be done. The following message is shown: "Task 2 (0/5): please estimate the area of the 5 shapes

Task 1: please select 5 different shapes. Selection is based on what you consider to be easy for the area estimation and comparisons. Selecting shapes of different sizes is highly advisable.



Submit

Figure 4.6: The select 5 shapes page

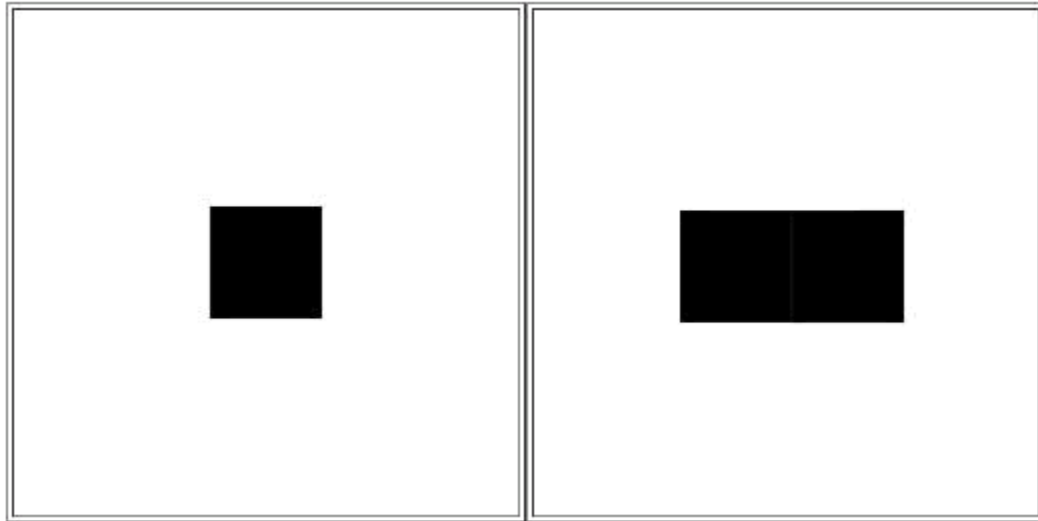


Figure 4.7: How to use the unit to estimate the area example

you have just selected in units (a black square) i.e. how many black squares you need to fill up the shape. The area estimation in units for the shape on the right is 2 units square. That is because you can use 2 squares to fill up the shape on the right” (See Figure 4.7). After some discussion, we decided for the square unit to be of size 3136 pixels. That is $56 * 56$ or 56 unit square. This size was chosen because it is not too small to make it almost impossible to estimate the area of the shape. Also, it is not too large to make the unit lose its properties and become as large as one of the shapes. It was just the right size for our purpose. After that, the user can click on the start task 2 button, which will allow the user to start the area estimation process for the 5 shapes that the user have selected. Shapes will be shown once at a time to ensure that the user only concentrates on one shape at a time. You may see Figure 4.8 for an example of what the area estimation in units page looks like. The user can only input valid numeric values. No negative numbers, zeros, spaces, or even excursively large numbers are allowed. If the user inputs an invalid value, an appropriate error message will appear. If a value is valid and the submit button is clicked, the user will be taken to the next page where the next shape estimation will take place. Users

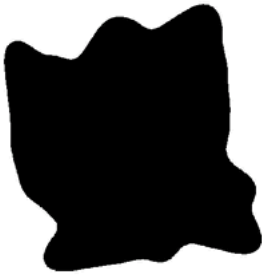

Shape	Unit	Area estimation in units:
 <p data-bbox="613 630 630 651">4</p>		<p data-bbox="1055 472 1266 493">Area = <input type="text"/> Units <input type="button" value="Submit"/></p>

Figure 4.8: Taks 2: estimate area of a shape in units

have to estimate the area of all 5 shapes.

In the last part of the experiment, selected shapes were shown in pairs side by side. There are 10 unique pairs that can be formed from the five shapes, therefore, 10 comparisons were performed. Respondents were asked to choose if the pair is approximately equal, if shape 1 is [(value from user)] times larger than shape 2, or shape 2 is [(value from user)] times larger than shape 1. The same three options are repeated for each of the ten pairs. See Figure 4.9 for an example of a comparison question. For example, if a participants answered a question by choosing “shape 1 is [1.5] times larger than shape 2”, this means that shape 1 is 1 and a half larger than shape 2. After completing all 10 comparisons, an appreciation page is displayed. A detailed flowchart of experiment 1 PHP code is shown in Figure 4.10. Some respondents answered the questionnaire inconsistently, therefore, their questionnaire was not taken into consideration. The reason for that is that they probably were not able to fully comprehend what should be done or being confused which shape is larger than the other.

Task 3 (5/10): Compare the following 2 shapes according to their areas

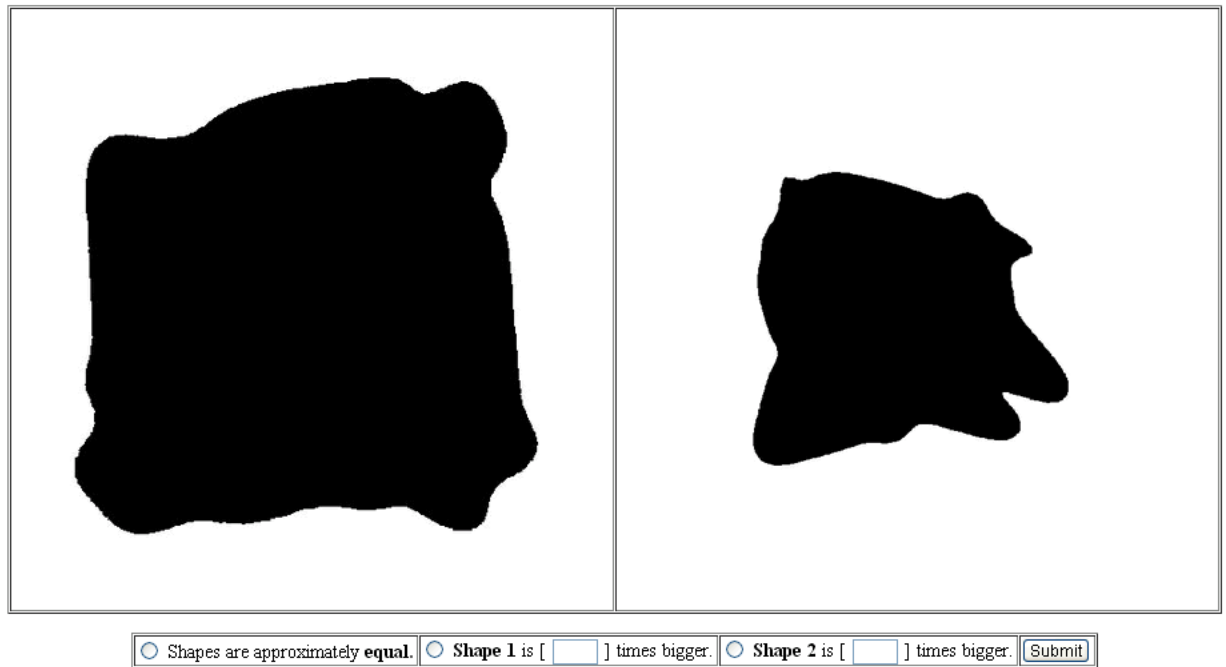


Figure 4.9: Taks 3: Compare the two shapes

4.3.3 Stimuli

A pool of 70 random shapes that were randomly generated with different area sizes was used. Respondents would then choose 5 of them to perform the area estimation and comparison. They were asked to choose the 5 shapes that are easiest to estimate for their area. This procedure gives more randomness to the experiment since probably not all participants would use the same shapes, which produces a variety of shape combinations. In addition, this will give an idea on the type of shapes that are easier to use for area estimation according to the participant's selection. After creating around 100 shapes using the algorithm in Chapter 3, we decided to choose only the ones shown in Figure 4.6. Random shapes should not be too simple for area estimation such as randomly generated squares or circles [1]. Therefore, some of them were eliminated. To add even more randomness to the experiment, the PHP code randomly rescaled the 5 shapes after the user selection.

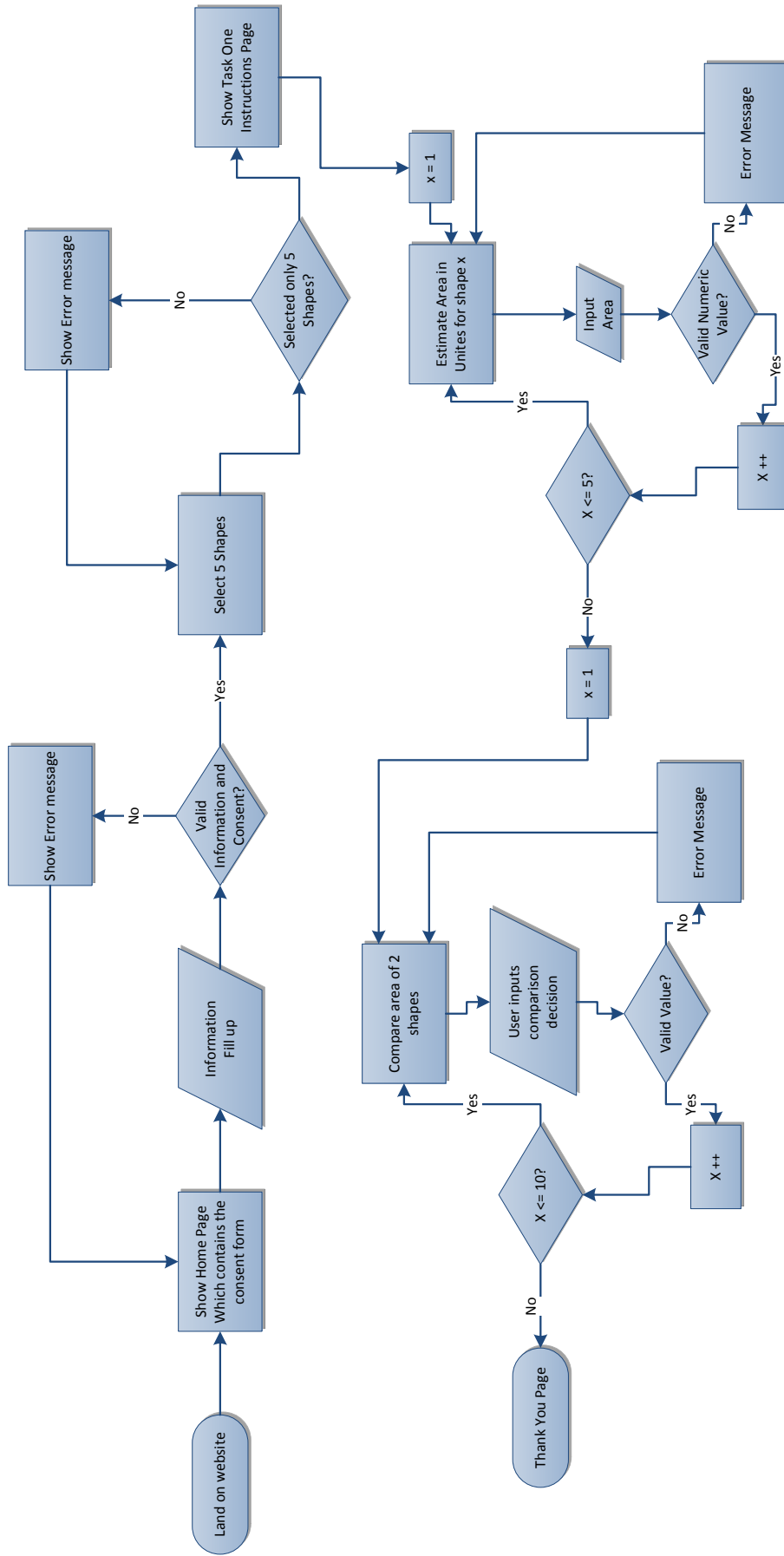


Figure 4.10: Experiment 1 flowchart

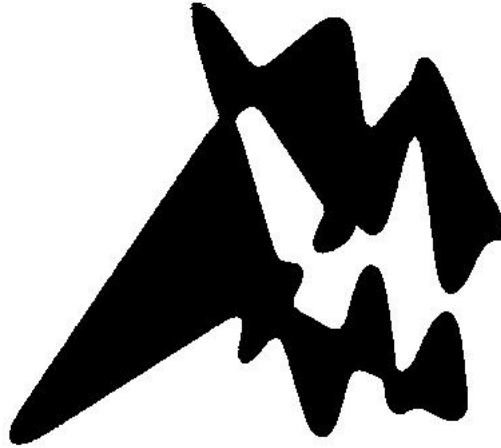


Figure 4.11: Not too simple shape

4.3.4 Procedure

All of the shapes were polygons with a randomly chosen number of points between 5 and 90. The random numbers were generated using `Math.Rand()` function in Java. Polygons are then generated and a Gaussian blur is applied to make rough edges smooth. Afterwards, a threshold is set to turn grey pixels, which are a result of the blurring, to black or white. As mentioned earlier, after generating over 100 shapes, we went through all the shapes and decided to choose 70 to ensure that random shapes are not very simple such as randomly generated squares or circles. Also, not too complicated shape similar to Figure 4.11. You can notice that this shape is included in the 70 shapes pool used. This is only to test the behavior of the participants towards these kind of shapes. Next, the areas of all the shapes were calculated using a Java method, introduced in Chapter 3. The method counts all non white pixel in the image and displays the area of the shape in pixels. Obviously, it was the simplest way to calculate the area of an irregular random shape. The areas of the

shapes are then recorded and saved to a MySQL database for easy access through PHP. The largest shape area size is 126539 pixels and the smallest area size is 23752 pixels. That is about 1-5 ratio. When the user selects the 5 shapes, they will be rescaled to add more randomness to the experiment. It is done by manipulating how the image is displayed on the browser by randomly modifying the HTML `img` tag height and width attributes. ``. `height` and `width` attributes should have the same values to ensure that the image is a square. All 70 shapes are initially represented as 500 * 500 image size. Because of the rescaling, the area of the shape would definitely change. Therefore, the new area in pixels should be calculated by PHP and saved in MySQL. Area of the shapes dramatically change when the shape is rescaled and the 1 to 5 ratio is lost. Let us take the shapes shown in Figure 4.12 and the one in Figure 4.13 as an example. The original area of the first one, before scaling is 121787 pixels and 71899 pixels for the second one. The first shape is about 1.7 times larger than the second one.

The rescaling process is shown here. An array of 5 numbers that represent the new height and width is initialized, then each element of the array is assigned a value. One of these values is 500 and the other 4 values are randomly assigned. The array is then shuffled to make sure that it is in a random order. Next, new areas are calculated according to the formula 4.1.

$$newarea = \frac{newheight * newwidth}{500 * 500} * oldarea \quad (4.1)$$

From the example shown earlier, Figure 4.13, and with a new height and width values of 128 and by applying 4.1, we get 4711 pixels which is the new area size of the rescaled shape.

$$newarea = \frac{128 * 128}{500 * 500} * 71899 \approx 4711pixels \quad (4.2)$$

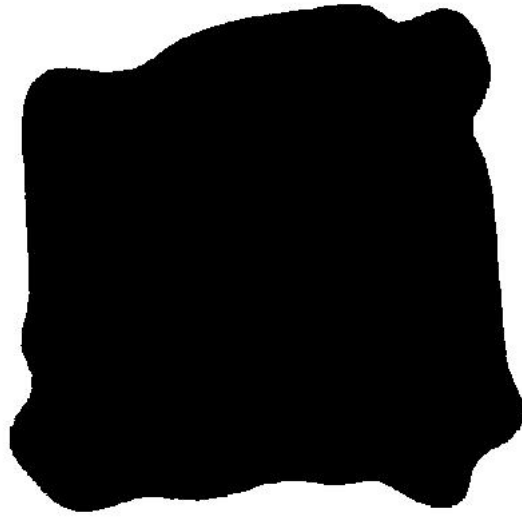


Figure 4.12: A shape with original area 121787 pixels before scaling



Figure 4.13: A shape with original area 71895 before scaling

32928	31360	78400	3763.2	10662.4
40217	40482	121787	4716	19823
0.181242	0.225335	0.356253	0.202036	0.46212

Figure 4.14: Analysing and calculating the relative error for area estimation in units

4.3.5 Analysis

PHP code is used to extract the data. The values are then saved in several `.txt` files. Along with the serial number for each record, the first file contains the users' area estimations for the 5 shapes and the actual area size in pixels. The values are then copied to MS Excel where this part of the analysis will take place. For each record, the value of the user estimated area in units is multiplied by the unit size in pixels and then compared with the actual area of that shape. For example, in Figure 4.14 the first line represents the user's area after multiplying the unit size by the user input, and the second line contains the actual area. The third line is the error rate, which is calculated using the following formula.

$$RelativeError = |ActualArea - EstimatedArea| / ActualArea \quad (4.3)$$

The relative error for shape 1 in Figure 4.14 is 18.14%. The average error for all 5 shapes is then calculated. To analyze the data for the second part, values of the relative comparison coefficients are entered into the pairwise comparison matrix. For 5 shapes, only 10 values are needed and these values fill the upper triangle of the matrix. We do not have to worry about the diagonal since all values on the diagonal are ones, because comparing the same shape with itself gives 1 on the main diagonal. This is a reciprocal matrix that satisfies the following formula.

$$a_{ij} = \frac{1}{a_{ji}} \quad (4.4)$$

This would take care of the lower triangle matrix. Table 4.2 is an example that shows the pairwise comparison matrix for comparing 5 shapes. Afterwards, JConcluder [72]

Table 4.2: A pairwise comparison matrix example for one of the records

	<i>shape</i> ₁	<i>shape</i> ₂	<i>shape</i> ₃	<i>shape</i> ₄	<i>shape</i> ₅
<i>shape</i> ₁	1	1	0.4	0.87	0.125
<i>shape</i> ₂	1	1	0.3	1	.15
<i>shape</i> ₃	2.5	3	1	2	0.3
<i>shape</i> ₄	1.1	1	0.5	1	0.2
<i>shape</i> ₅	8	0.65	3	5	1

is used for further data analysis.

The JConcluder is a program that is developed by Ding Xu, a fellow student at Laurentian University, and is part of his thesis. The program takes the 10 upper triangle values of the pairwise comparisons matrix and performs the analysis needed to calculate the weight vector and the inconsistency values. The real weight vector and the estimated weight vector are compared and the relative error average is calculated. Furthermore, the time spent to complete each task was analyzed and served to demonstrate which method takes less time and the average time spent to complete each task. Records with incomplete or apparent phony fields were removed and not considered.

4.3.6 Results

The average error for estimating the area by the direct method was 41.82% which is much larger than the 25.75% reported in [1] using the same method for random, but equal in area shapes, which is pretty high. In addition, it is higher than the 15.42% error reported in [33] for the 1D case using bars. Figure 4.15 shows the average error rate for participants when the direct estimating the area in units method is used. On the other hand, the average estimation error for the pairwise comparisons

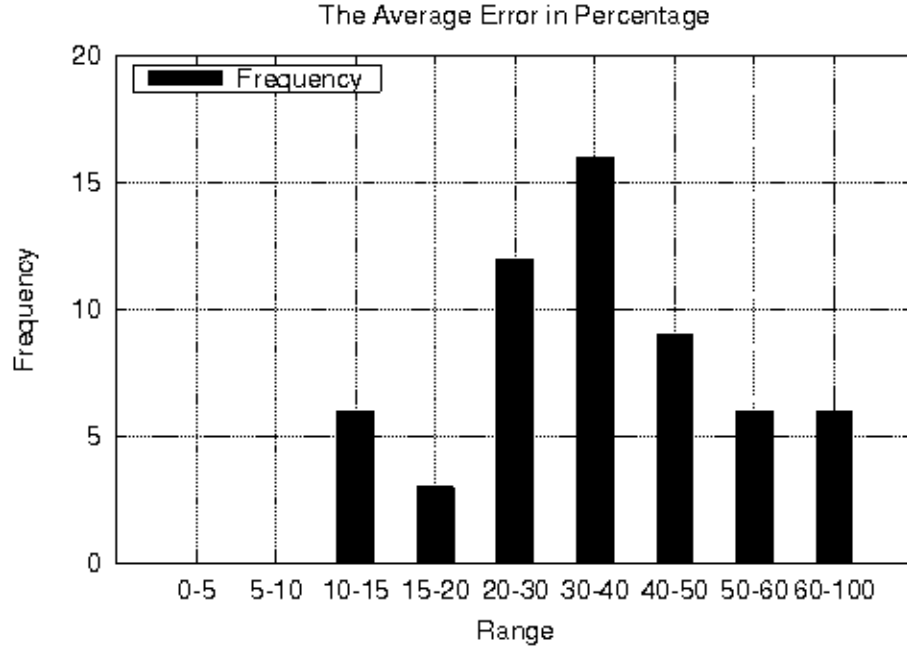


Figure 4.15: Histogram showing the average error when using the direct method

method is only 24.85%, which is higher than the ones reported in [1] and [33] using the same method. Figure 4.16 shows the average error rate when using the pairwise comparisons method. Comparing the average error rate when the direct method (41.82%) and the pairwise methods (24.85%) shows that, the accuracy rate improves when the pairwise comparison method is used. The difference between the errors derived from the direct method and pairwise comparisons method is called the gain of accuracy [1]. The gain of accuracy (23.97%) is greater this time. This is a tremendous improvement, decreasing from 48.82% to 24.85%.

In this experiment, the average inconsistency is 0.43. Some inconsistency rates were unacceptably very high. Figure 4.17 shows the level of inconsistency for participants in experiment 1 (More to come on experiment 1 inconsistency in the next subsection).

The average total time that it took participants to complete all the tasks was about 7.6 minutes. Figure 4.18 shows the average time of each task in details. The average time needed to estimate the area of the 5 shapes in units, was 3.09 minutes. On the other hand, the time needed to compare all the shapes using the pairwise

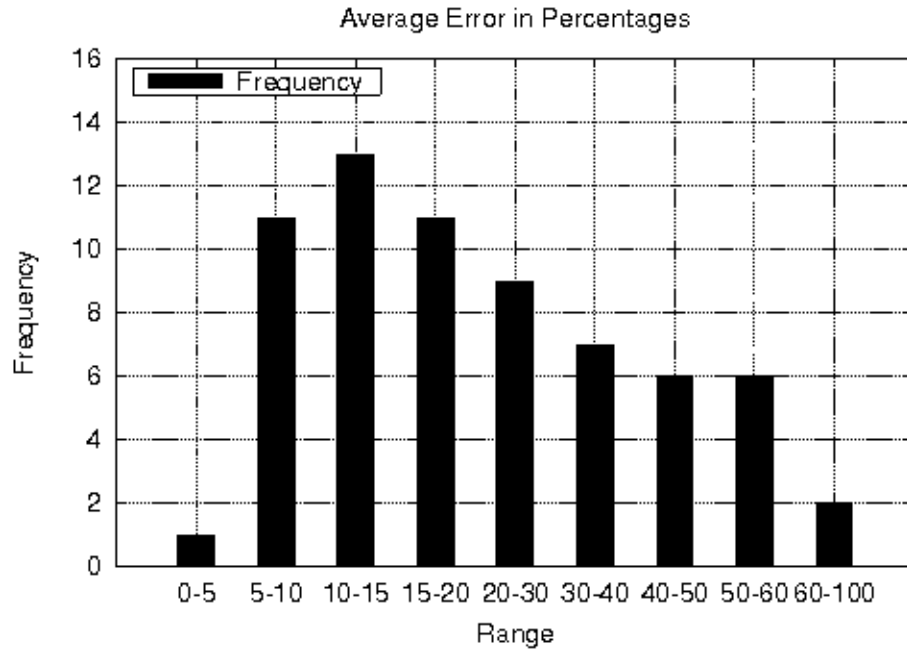


Figure 4.16: Histogram showing the average error when using pairwise comparisons method

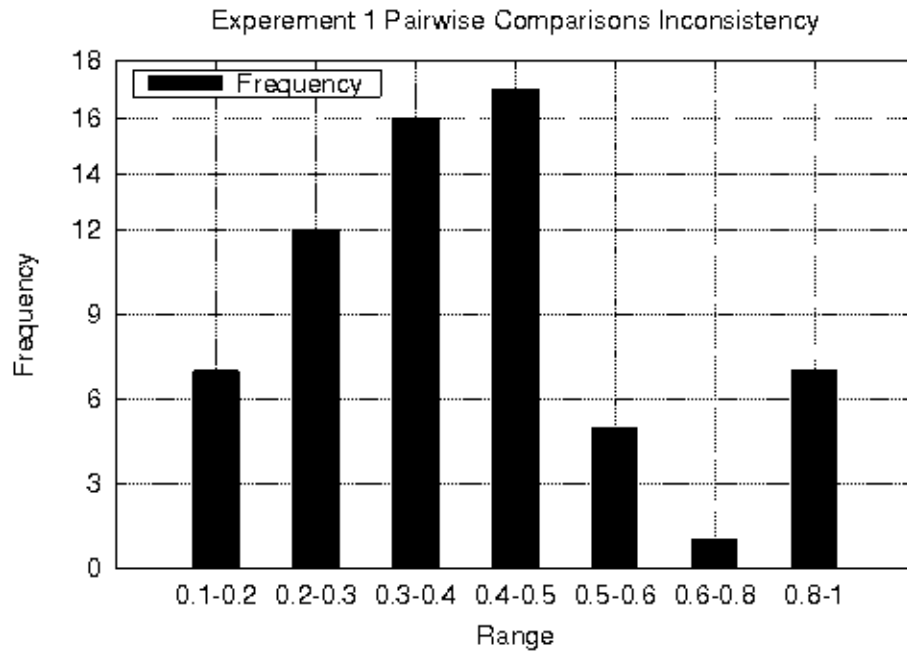


Figure 4.17: Histogram showing the inconsistency when using pairwise comparisons method

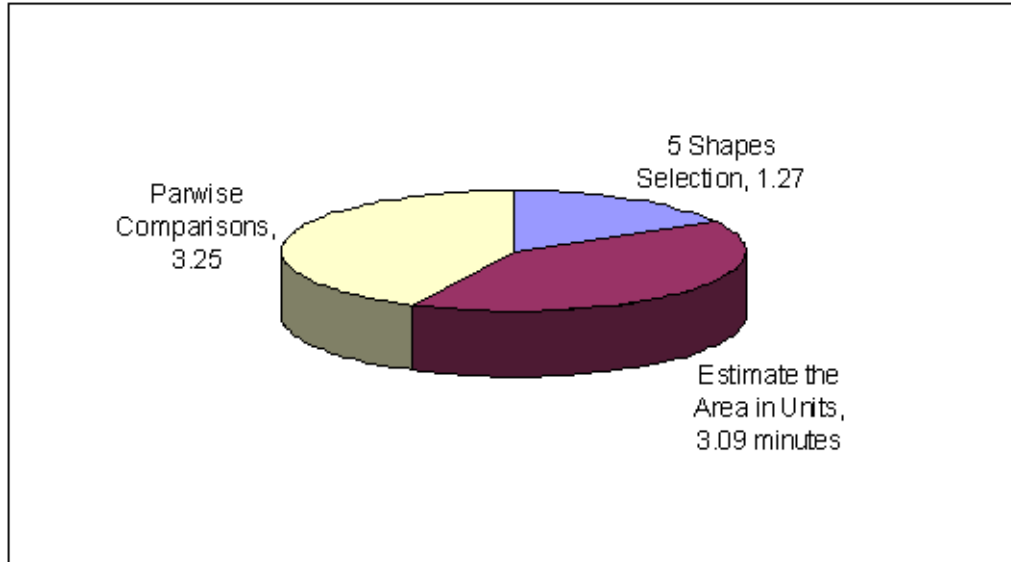


Figure 4.18: The average time needed to complete each task in experiment 1

comparisons method was 3.25 minutes. The process of estimating the area of the shapes in units involves going through 5 pages. However, the process that uses the pairwise comparisons method involves visiting 10 different pages. Therefore, the time needed to complete one page using the pairwise comparison method is 0.325 minutes, which is much better than the time needed to submit an area estimation in units page, which is 0.618 minutes. Although the total average time needed to complete the area estimation using the direct method is marginally lower, the accuracy or average error rate dramatically decreases when the pairwise comparisons method is used.

4.3.7 Problems encountered in Experiment 1

Because of the higher inconsistency rates in many records, we suspected that there should be something wrong in our experiment. We decided to investigate this matter in depth. First, we went over the PHP code to check for any bugs but none were found. We went over the data analyzing techniques used and things looked fine. We also checked our pairwise comparisons matrices and the way we populated them for all records but we found nothing wrong. Eventually, we found out that the main

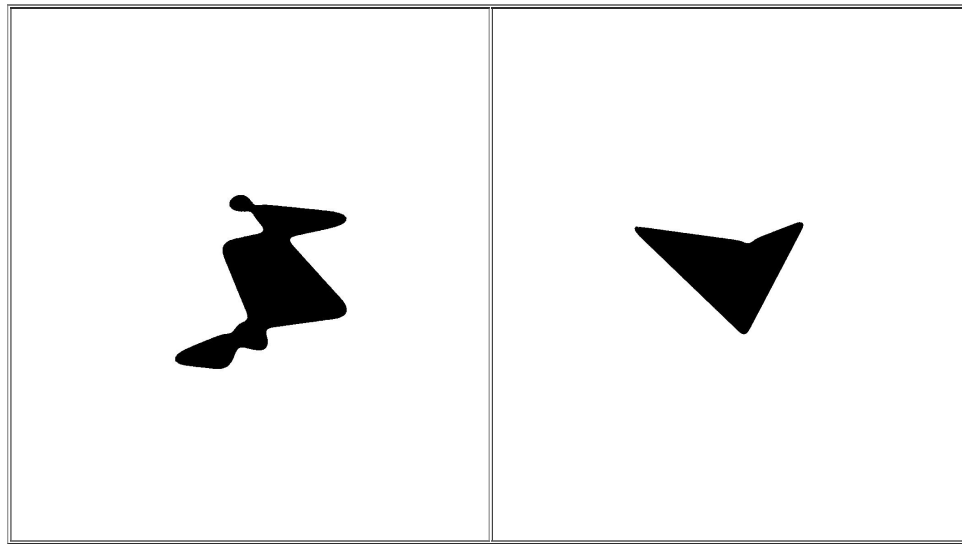
problem that influenced the high inconsistency in many records was the not so good user interface design. Many participants were confused when they were asked to perform task 3 (See Figure 4.19). Task 3 shows 2 shapes and asks the user to compare them. There are three possible options to choose from. Users can choose that the shapes are equal, shape 1 is larger than shape 2, or shape 2 is larger than shape 1. The user needed to answer that question 10 times for 10 different pairs. Consequently, many users were confused or lost concentration during the process. Some of them choose to select that shape 2 is larger then shape 1, but meant the opposite. When doing so, the pairwise comparison matrix for that record completely went wrong. For example, if the user wanted to enter that shape 1 is [3] times larger than shape 2 but Shape 2 is [3] times larger shape 1 is selected, this ruined the pairwise comparison matrix. Instead of the value of 3 to be saved in corresponding matrix position, $\frac{1}{3}$ is saved. This will severely influence the inconsistency of the pairwise comparisons matrix. Therefore, other measures had to be implemented regarding the user interface design of the questionnaire. Another issue that we faced in experiment 1 was a problem with the ratio of the displayed shapes areas. In other words, the area for 2 or more shapes were close to each other for many records which resulted in redundancy. It had an impact on the pairwise comparisons matrix and of course the inconsistency. Problems encountered in experiment 1 influenced the design of experiment 2 to get the best results possible.

4.4 Experiment 2

To overcome all the glitches and drawbacks in experiment 1, the user interface redesign was necessary. Good user interface design is very important for productivity and accuracy improvement. It is one of the most important parts in any system. The software code is invisible and users do not want to worry about the way the code is written. What users find important is the simplicity and visual aesthetics of the interface. Nowadays, the amount of programming effort devoted to the user interface

Task 3 (1/10): Compare the following 2 shapes according to their areas

You may use numbers with or without fractions (such as 1, 3, 1.40, or 5.75) when comparing



Shapes are approximately equal. Shape 1 is [] times bigger. Shape 2 is [] times bigger.

Figure 4.19: Task 3: pairwise comparison page (experiment 1)

Table 4.3: The impact of bad screen design [15]

Seconds required per screen	Years required to process 4.8 million screens
1	0.7
5	3.6
10	7.1
20	14.2

is more than 50% of the entire time needed to complete the system [15]. Creating a good user interface is not too simple for a software programmer. User interface software should be prototyped, tested, and modified repeatedly [47]. According to [15], “User interface design is a subset of a field of study called human-computer interaction (HCI). Human-computer interaction is the study, planning, and design of how people and computers work together so that a person’s needs are satisfied in most effective way.” For HCI designers, many factors must be considered: What people want and what people expect? What physical limitations do people have? How do the perceptual and information processing system work? Besides, designers should consider the technical characteristics and limitation of the computer hardware and software. Once an interface has been constructed, how humans react to it and interact with it should be measured [22]. It can be done by extreme testing procedure such as the try to break test. Building a good design is very beneficial to booth the accuracy and the productivity of the system. A research found out that by processing 4.8 million screens per year, poor clarity made users spend an extra second per screen. That is almost an additional one year to process all screens [15]. Table 4.3 shows the impact of inefficient screen design on processing time.

4.4.1 Sample

There are 93 recorded observations used in this experiment. As in experiment 1, any adult could participate as there was no particular procedure for selecting the participants. Some participants who engaged in this experiment participated on the previous one also. Others participated solely in this one. As in experiment 1, only the date, time, and participants' answers were recorded. The email was also recorded only if the participant asked for the results to be sent to him/her when the study is complete. No IPs or any personal identification was stored.

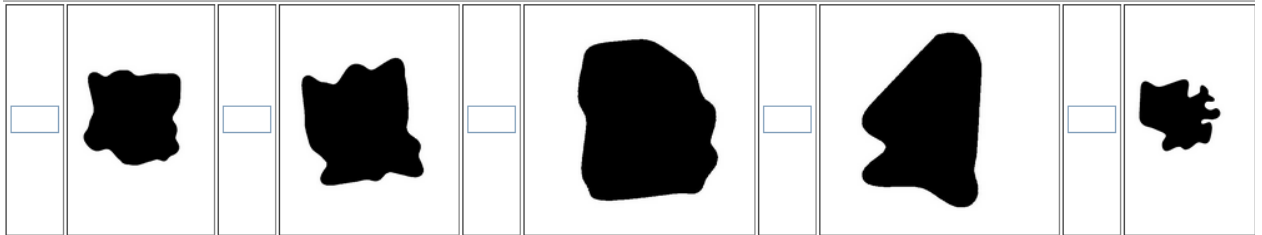
4.4.2 Measures

In the first part of experiment 2 and similar to experiment 1, participants were asked to choose 5 shapes from a pool of 70 shapes. These shapes were previously randomly generated and have different areas. The 70 images are all of size 500 * 500. They were rescaled to a smaller size of 63 * 63 to make all 70 shapes fit the screen for simpler user selection (Figure 4.6). The PHP code validates the participant's response and makes sure that the user chooses only 5 shapes. Otherwise, a visible error message will be shown and the user will be able to edit the shape selection to make sure only 5 are chosen. Users are encouraged to select shapes based on what they consider being easy for the area estimation and comparisons. Now that the user has chosen the 5 shapes, he/she can start performing the second task. An extra step was added to the experiment to overcome the user interface design problem that occurred in experiment 1. This extra step asks the user to put in order the 5 randomly generated shapes from the largest to the smallest, where the largest gets the value of 1 and the smallest gets 5. This screen can be seen in Figure 4.20. This is to ensure that the user is able to distinguish the visible size difference among the shapes. In addition, it gives the ability to be consistent in the way the pair of shapes is displayed on the 10 pairwise comparisons screens. The larger shape is displayed on the left and the

It is very important to make sure to input accurate data as much as possible otherwise inconsistent data will be produced. Thank you for your patience and understanding.

Here are the shapes that you have just selected.

Rank largest to smallest from 1 to 5, where the largest shape gets the value of 1, the smallest gets the value of 5 and the shapes in between get the values of (2, 3, 4) respectively



Rank from largest to smallest, from 1 to 5. Where the largest shape gets the value of 1, the smallest gets the value of 5

Submit

Figure 4.20: Order shapes from largest to smallest screen

smaller shape on the right all the time. The system will allow the user to proceed to the area estimation in units page only if the ordering is correct. Otherwise, they would need to select 5 new shapes. After that and exactly the same as experiment 1, the user will be shown an example to help understand how the area estimation in units should be done. The black square example will be shown as in Figure 4.7. This time we decided for the square unit to be of size 1600 pixels. That is a $40 * 40$ unit square, which is smaller than the one used in experiment 1 due to the fact that shapes now are in rather fixed scale ratio size. On the help page, the user can click on the start task 2 button which will allow to start the area estimation process for the 5 shapes. As in experiment 1 shapes will be shown one by one to ensure full concentration on one shape at a time. Figure 4.8 shows how the direct method page is displayed. The user can only input valid numeric values. No negative numbers, zeros, spaces, or even very large numbers are allowed. If the user inputs an invalid value, an appropriate error message will be shown. If a value is valid and the submit button is clicked, the user will be taken to the next page, where the next shape estimation will take place. Users have to estimate the area of all 5 shapes. In the last part of the experiment,



participants were shown two of the five random shapes side by side. Now that the user have ranked them from largest to smallest, the larger shape is always displayed on the left side, as shape 1 and the smaller shape is displayed on the right as shape 2. There were 10 unique pairs that can be formed from the five shapes. So, 10 comparisons were performed. This time, respondents were allowed to choose only one option which is “Shape 1 is [(value from user)] times larger” for each of the ten pairs of shapes. A screenshot of that page is shown in Figure 4.21. If the user feels that shape 1 is not larger than shape 2, a “Disapprove” button is provided and can be clicked. Users can click the “Disapprove” button only if they think that shape 1 is not larger than shape 2. Even though this is not possible to happen since the ordering already took place and the larger shape is always displayed on the left side, the option of disapproving was provided since different people may have different visual perception of shapes. People see with their brains but the eyes take in light and then projects it to retina. Cells carry the signal along a pathway to the brain, then interpret the projection of the retina. Afterwards, light is turned into information that can be interpreted by the brain [44]. People have different brain capabilities. Thus, they perceive what they see differently and that is why the disapprove button is provided. If the disapprove button is clicked, users would need to select 5 new shapes. After completing all 10 comparisons, a thank you page is displayed. Please see Figure 4.22 for a detailed flowchart for the PHP code.

4.4.3 Stimuli

Pretty much like experiment 1, a pool of 70 random shapes that was randomly generated with different area sizes was used. Respondents were asked to choose 5 of those shapes to perform the area estimation and comparison. This gives more randomness to the experiment due to the fact that probably not all participants would use the same shapes. In addition, this will give an idea on what type of shapes are easier to use for area estimation according to the participant’s selection. Only the shapes

Task 3 (1/10): Compare the following 2 shapes according to their areas. You may select one option only. You may use numbers with or without fractions (such as 1, 3, 1.40, or 5.75) when comparing

Click "Disapprove" button only if you think that shape 1 is not larger than shape 2

Shape 1	Shape 2
	

Shape 1 is [] times larger. 1 of 10

Figure 4.21: Pairwise comparisons used in experiment 2

shown in Figure 4.6 were used. This is because random shapes should not be very simple such as randomly generated squares or circles [1]. To add even more randomness to the experiment, the PHP code randomly rescales the 5 shapes after the user selection.

4.4.4 Procedure

As mentioned in experiment 1, all shapes were originally polygons with a randomly chosen number of points between 5 and 90. The random numbers were generated using `Math.Rand()` function in Java. Polygons are then generated and filled with black and a Gaussian blur is applied to make rough edges smooth. Afterwards, a threshold to transform grey pixels to black or white is used. The next step was to scale all 70 shapes to make them equal in area with $< 0.1\%$ margin at most. Next, we saved the new area of all the shapes. The largest size after rescaling is 23834 pixels and the smallest is 23664 pixels. The difference between the largest and the smallest is 170 pixels, which is too small to be noticed. Therefore, all of the shapes became

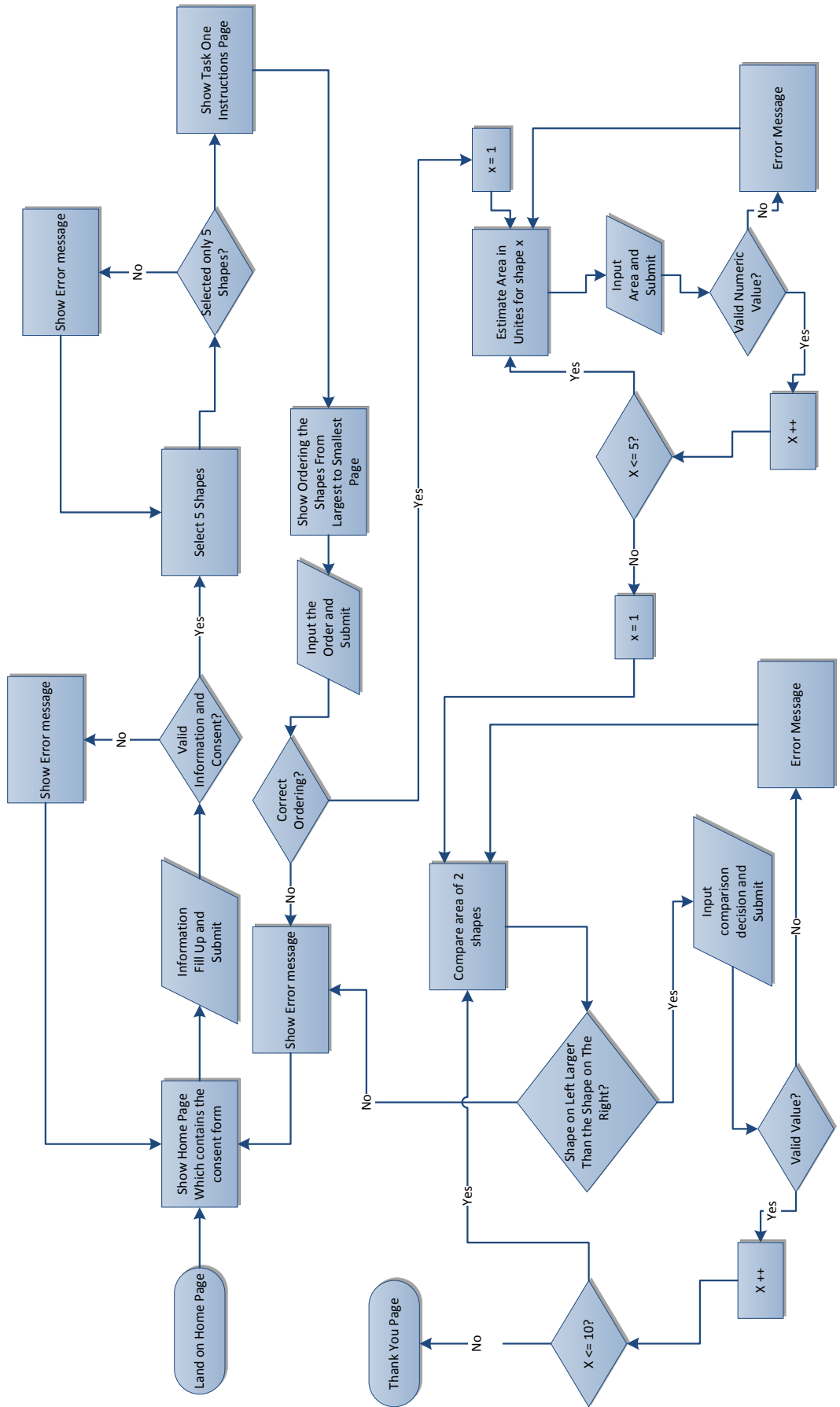


Figure 4.22: Experiment 2 Flowchart

approxitally equal in area. The areas are then recorded and saved to a MySQL database for easy access through PHP. To overcome the problem of experiment 1 consistency, we needed to be sure that the 5 selected shapes are displayed to the user in 1-5 ratio from largest to smallest. That is why we performed the previous step of rescaling all shapes to approximately equal in area shapes and then apply a new random scale to have the 5 shapes in a 1-5 ratio. This can be done by manipulating how the image is displayed on the browser. Next, Shapes are displayed on the ranking screen in no particular order. The user then orders them from largest to smallest. Table 4.4 shows the steps used for experiment 2 to have the shapes in a 1-5 ratio.

Table 4.4: Experiment 2 steps

Step 1 Scale all 70 shapes to make them equal in area.

Step 2 Show shapes for user selection.

Step 3 Randomly re-Scaling shapes to have areas in 1-5 ratio.

Step 4 Show shapes ordering page.

Step 5 If correct ordering, user inputs area in units for 5 shapes. If not, select 5 new shapes.

Step 6 Pairwise comparison method for the shapes having the larger shape on the left.

Step 7 If the larger shape is not on the left, disapprove and start again.

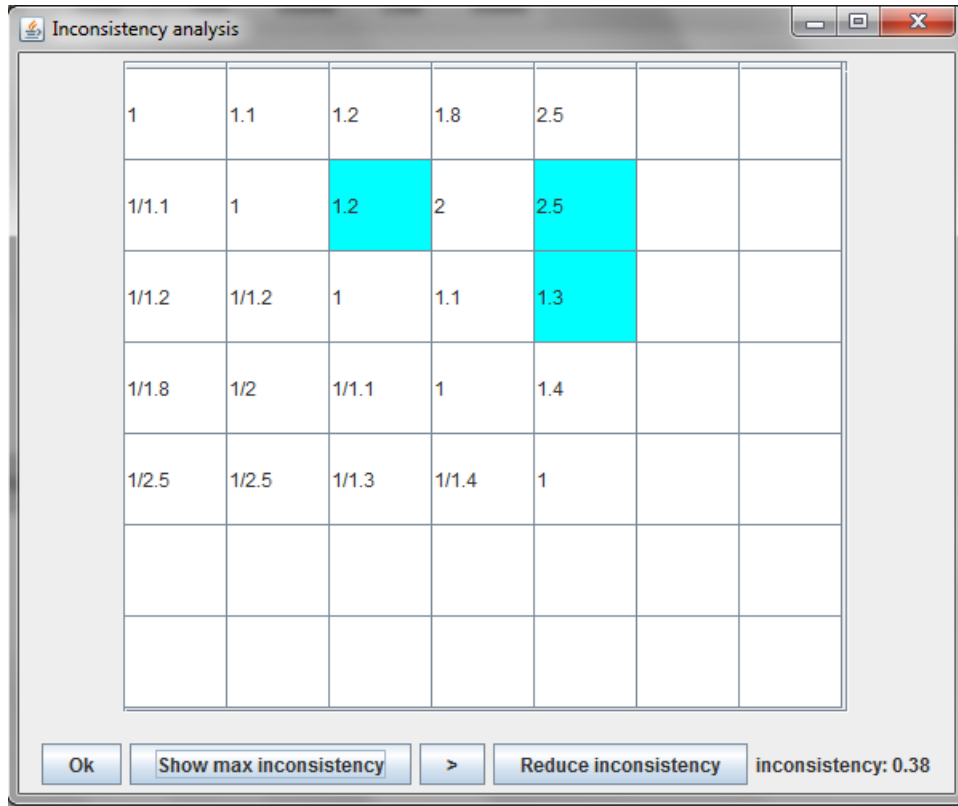


Figure 4.23: A pairwise comparisons matrix in JConcluder

4.4.5 Analysis

Similarly to the analysis procedure performed in experiment 1, the PHP code is used to extract the data from the MySQL database. Excel and JConcluder [72] are used, as in the previous experiment as well. Figure 4.23 demonstrates a pairwise comparisons matrix inputted in JConcluder, and the inconsistency is shown as well. The weight for each criteria is demonstrated in Figure 4.24.

4.4.6 Results

The average error rate when estimating the area of random shapes in units (direct method), is 30.3% for the 93 observations. Figure 4.25 shows the error rates for all observations when the direct method is used. On the other hand, the average error rate is only 11.96% when the pairwise comparison method is used, and this can be

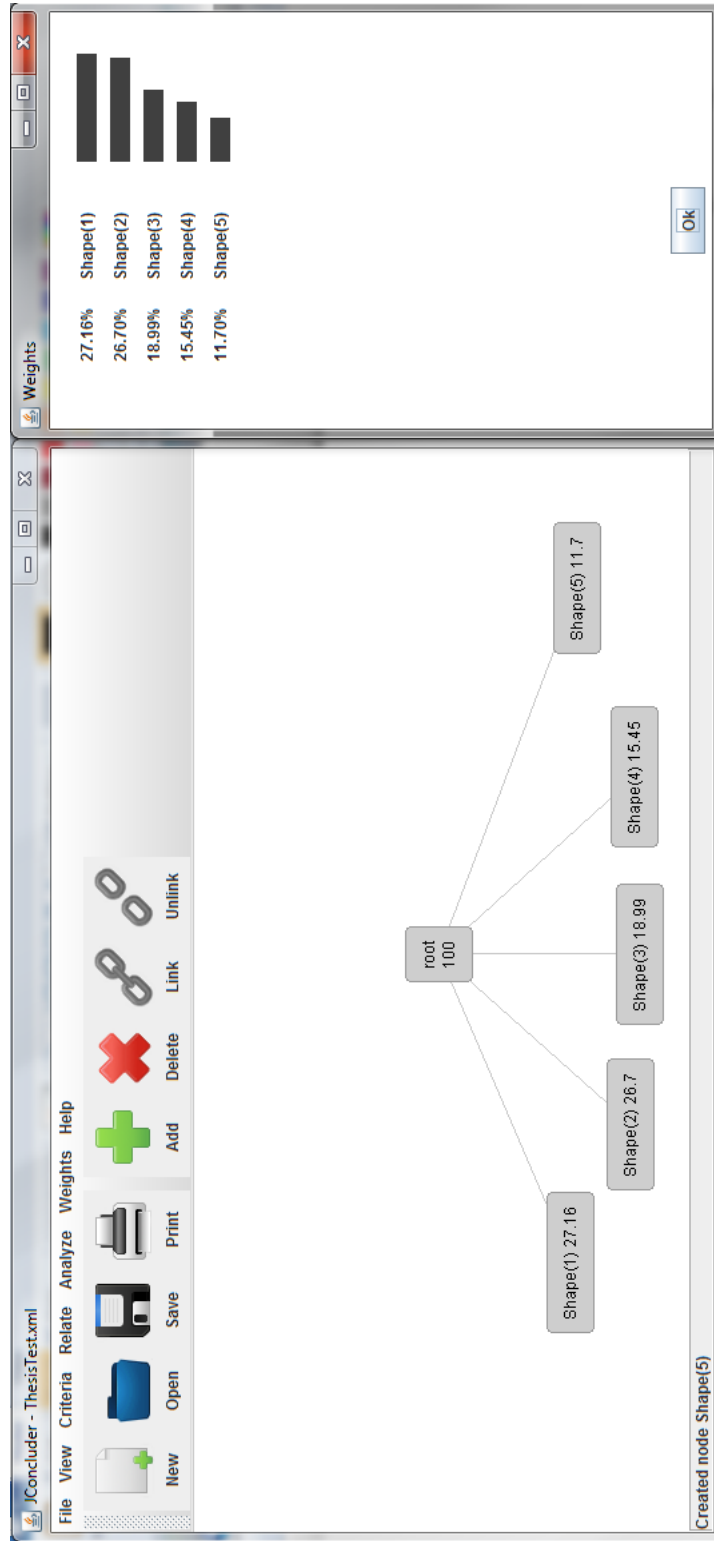


Figure 4.24: Weights of criterias example in JConcluder

seen in Figure 4.27. The gain of accuracy here is approximately 18.4%. The results are highly encouraging. The drop of estimation error, from 30.3% to 11.96% (See Figure 4.28), is even more spectacular than the 1D case reported in [33]. It is evident that the accuracy improves when random shapes' area estimation using the pairwise comparison method is enforced. In this experiment, the average inconsistency is only 0.27, which is reasonably acceptable. It is more appealing than the average inconsistency (0.43) reported in experiment 1. Figure 4.26 demonstrates the level of inconsistency recorded in this experiment, when the pairwise comparisons method is used.

The total average time that the participants needed to complete all tasks, is approximately 9 minutes. Figure 4.29 demonstrates the average time spent on each task. Although the average time taken to complete both the direct and pairwise comparison methods are close, the accuracy improves dramatically when the pairwise comparisons method is used.

Out of the pool of 70 shapes, participants were asked to choose 5 shapes to use for area comparisons and estimation. Users were encouraged to select the shapes based on what is considered simple for an area estimation and comparison. Some shapes were popular, while others were not selected at all. The most popular shape is shown in Figure 4.30, which was selected 9.20% of the time. This may be due to the fact that the shape looks like a "triangle," which is easier for participants to estimate its area. The second most popular shape is shown in Figure 4.31 and was selected 6.59% of the time. Figure 4.32 is the third most shape selected and it was chosen 6.48% of the time. These three shapes are the easiest for area estimation and comparisons, according to the observations.

The only shape, out of the 70 shapes, that was not selected at all can be seen in Figure 4.33. The second and third least selected shapes, that were selected a few times, appears in Figure 4.34. It appears that some shapes are not easy to do area estimation but others were overlooked for no logical reason known.

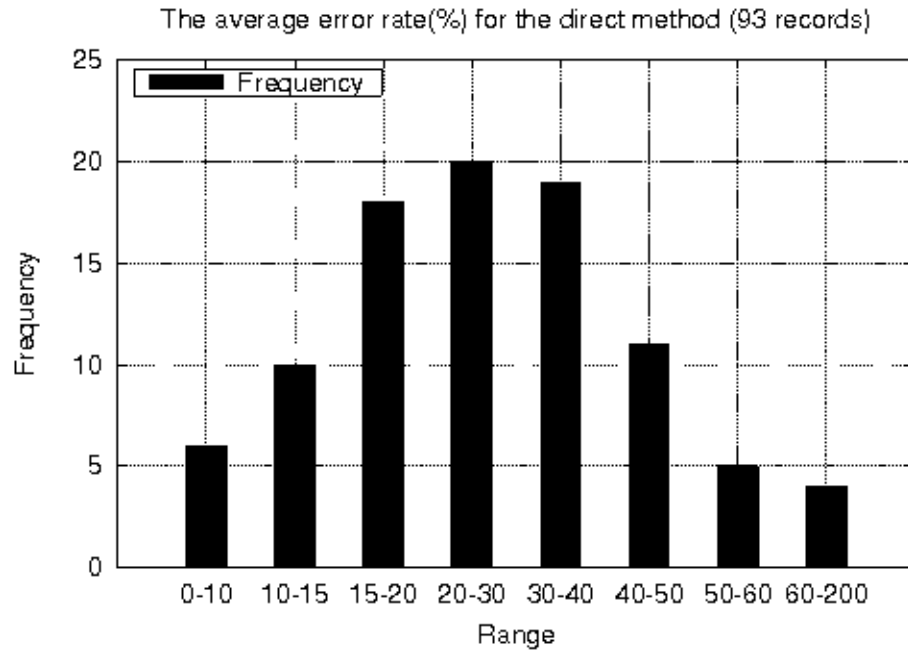


Figure 4.25: Histogram showing the average error when using the direct method in experiment 2

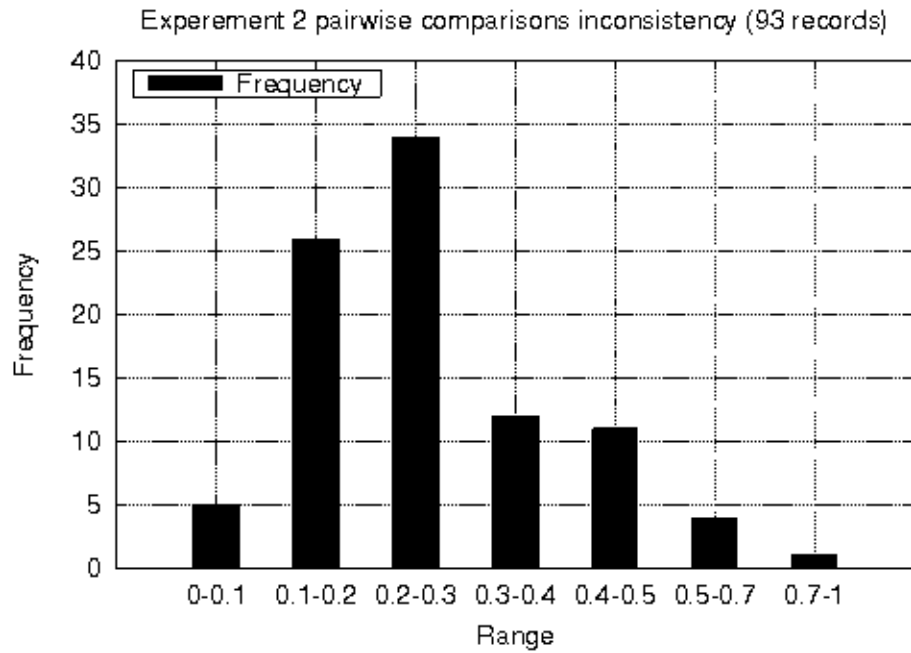


Figure 4.26: Histogram showing the inconsistency in the pairwise comparisons in experiment 2

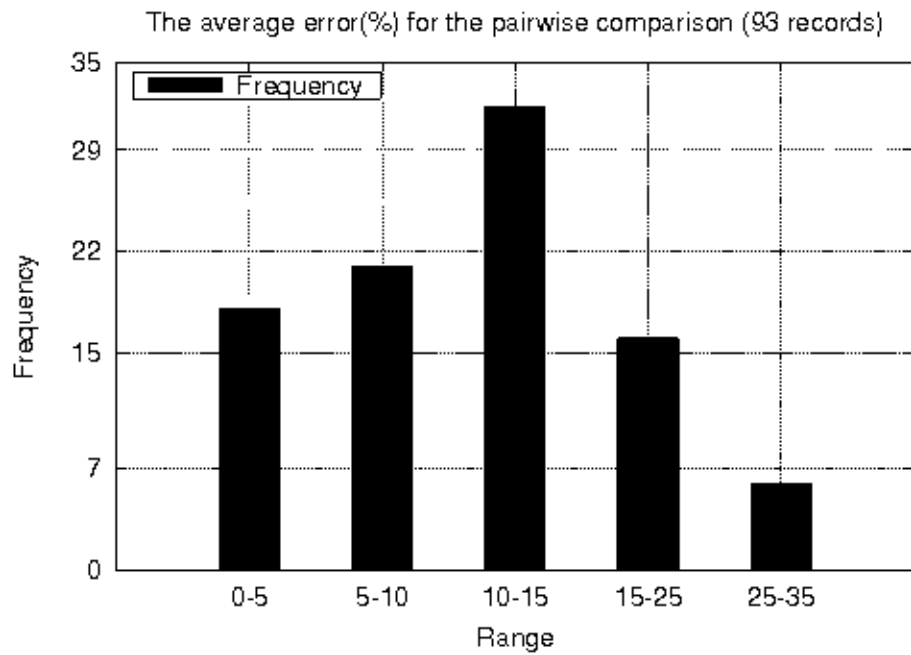


Figure 4.27: Histogram showing the average error when using the pairwise comparisons method in experiment 2

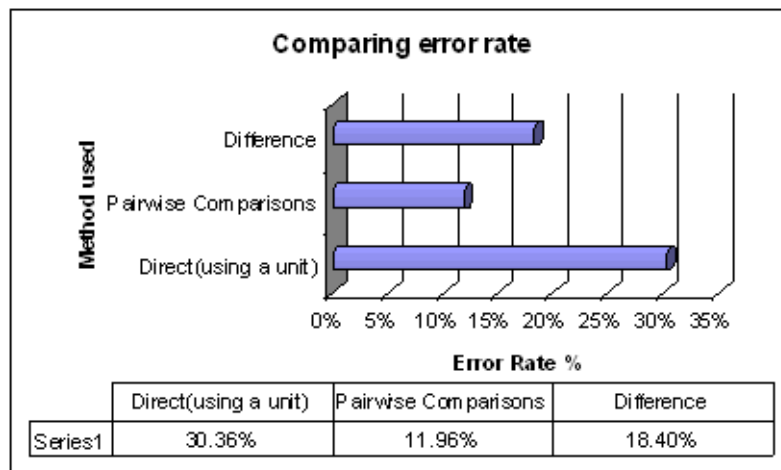


Figure 4.28: Comparing the average error rate when using the pairwise comparisons and the direct method for area estimation of random shapes

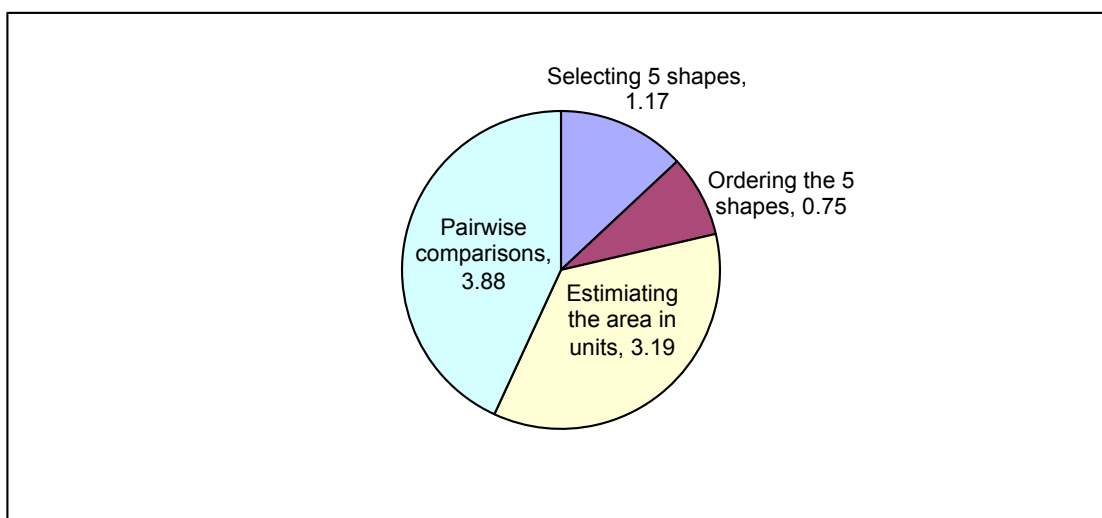


Figure 4.29: The time taken to complete each task in experiment 2 in minutes

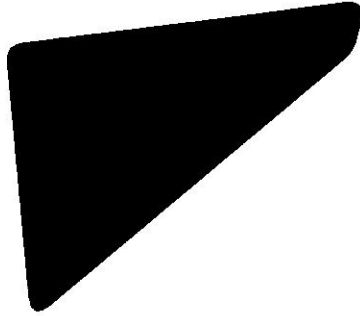


Figure 4.30: The most popular shape selected



Figure 4.31: The second most popular shape selected

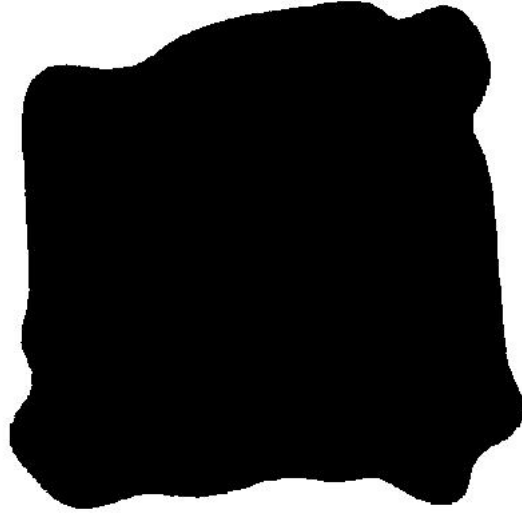


Figure 4.32: The third most popular shape selected

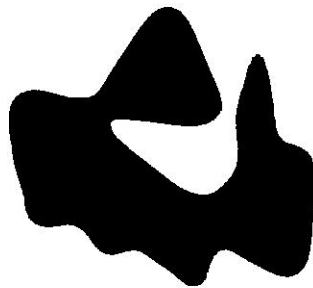


Figure 4.33: The shapes that was not selected not even once

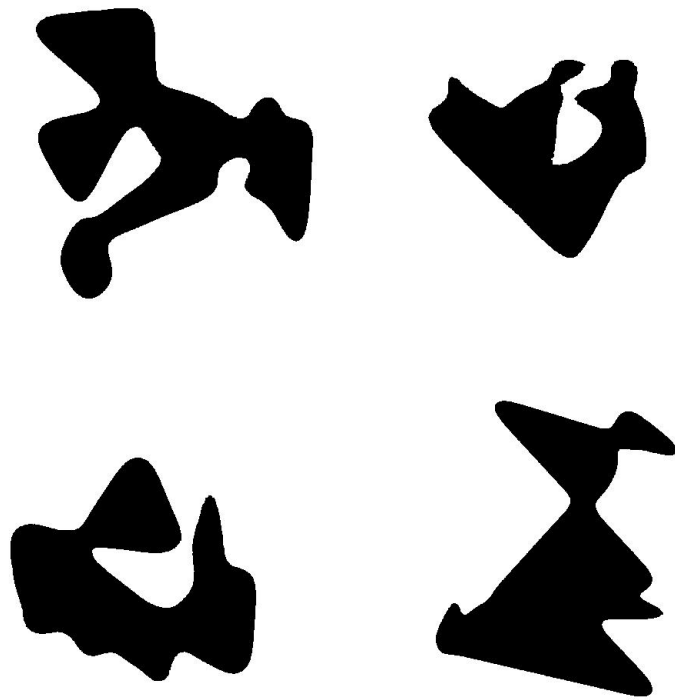


Figure 4.34: Shapes selected only once or twice

5 Conclusions

We have demonstrated that the Gaussian blur can be used for the generation of placated random shapes (accepted for publication in a journal [3]). The beauty of the random generation of placated but random shapes method is its simplicity. However, it is worth noticing that the method is simple yet not simplistic. Unlike in [1], the generated 2D random, but not equal in area, shapes were used to test the accuracy of pairwise comparisons.

The results of this experiment are in favor of the pairwise comparisons method over the direct method. The average error for the pairwise comparisons was nearly 11.96% versus 30.3% when the direct method is used. The gain of accuracy, which is the difference between the errors derived from the direct method and pairwise comparisons method, is around 18.4%. It is even more impressive than the 1D case reported in [33]. While the direct rating method may be straightforward to use and there is no need to compute the weights, there are trade offs in terms of accuracy and reliability. The pairwise comparison method gives more firmness for the final outcomes. The low average inconsistency value implies that the resulting error between the actual area of the shape and the area estimated by the observer is low. It is also worth mentioning that the average time taken to complete both the direct and pairwise comparison methods were close, but the accuracy improves dramatically when the pairwise comparisons method is used.

To our own knowledge and based on an intensive Internet search of academic databases, such as Web of Knowledge, this is the first study in the world for Monte Carlo 2D accuracy testing of pairwise comparisons.

In the future, a more sophisticated algorithm to generate placated random shapes is envisioned. The Gaussian value and the cut-off threshold may be assigned automatically to generate the desired placated nice looking shape without holes. This can save time and hassle in the shape generation process. Furthermore, the attempt will be made to use it as a plug-in for open source image processing systems (such as GIMP) when the software is perfected and extended to other filtering methods. Also, the bump function may be used.

Bibliography

- [1] P. Adamic, T. Kakiashvili, W. W. Koczkodaj, V. Babiy, R. Janicki, and R. Tadeusiewicz. Pairwise comparisons and visual perceptions of equal area polygons. *Perceptual and Motor Skills*, 108(1):37–42, 2013/05/24 2009. URL <http://dx.doi.org/10.2466/pms.108.1.37-42>.
- [2] Song Ho Ahn. Convolution. URL <http://www.songho.ca/dsp/convolution/convolution.html>.
- [3] A Almowanes, T Kakiashvili, and W W Koczkodaj. Generating placated random shapes for an area estimation study (accepted). *Journal of Applied Mathematics and Computational Mechanics*, 2013.
- [4] V Babiy, R Janicki, A Wassying, AD Bogobowicz, and W W Koczkodaj. Selecting the best strategy in a software certification process. In *Computer Science and Information Technology (IMCSIT), Proceedings of the 2010 International Multiconference on*, pages 53–58. IEEE, 2010.
- [5] Sandor Bozoki and Tamas Rapcsak. On saaty’s and koczkodaj’s inconsistencies of pairwise comparison matrices. *Journal of Global Optimization*, 42(2):157–175, 2008.
- [6] Sándor Bozóki, János Fülöp, and Waldemar W Koczkodaj. An lp-based inconsistency monitoring of pairwise comparison matrices. *Mathematical and Computer Modelling*, 54(1):789–793, 2011.

- [7] Google Code. `Effectserrodisplay.java`. URL <http://scalalab.googlecode.com/hg/scalalab281Src/scalaExec/gui/EffectsErrorDisplay.java>.
- [8] Josep Colomer. Ramon llull: from ars electionis to social choice theory. *Social Choice and Welfare*, 40(2):317–328, February 2013. URL <http://ideas.repec.org/a/spr/sochwe/v40y2013i2p317-328.html>.
- [9] Marquis de Condorcet. Essay on the application of analysis to the probability of majority decisions. *Paris: Imprimerie Royale*, 1785.
- [10] David Dailey and Deborah Whitfield. Constructing random polygons. In *Proceedings of the 9th ACM SIGITE conference on Information technology education*, pages 119–124. ACM, 2008.
- [11] Herbert Aron David. *The method of paired comparisons*, volume 12. DTIC Document, 1963.
- [12] Hepu Deng. Multicriteria analysis with fuzzy pairwise comparison. In *Fuzzy Systems Conference Proceedings, 1999. FUZZ-IEEE'99. 1999 IEEE International*, volume 2, pages 726–731. IEEE, 1999.
- [13] Nick Efford. *Digital Image Processing: A Practical Introduction Using Java (with CD-ROM)*. Addison-Wesley Longman Publishing Co., Inc., 2000.
- [14] Georg Frobenius, Ferdinand Georg Frobenius, Ferdinand Georg Frobenius, and Ferdinand Georg Frobenius. *Über Matrizen aus nicht negativen Elementen*. Königliche Akademie der Wissenschaften Sitzungsber, Kon, 1912.
- [15] Wilbert O Galitz. *The essential guide to user interface design: an introduction to GUI design principles and techniques*. Wiley, 2007.
- [16] Gaussiankernel.nb. The gaussian kernel. URL <http://www.stat.wisc.edu/~mchung/teaching/MIA/reading/diffusion.gaussian.kernel.pdf.pdf>.

- [17] GNU. *GNU Image Manipulation Program User Manual*. URL <http://docs.gimp.org/en/index.html>.
- [18] Rafael C. Gonzalez and R.E. Woods. *Digital image processing*. World Student Series. Addison-Wesley, 1992. URL <http://books.google.ca/books?id=CfQeAQAAIAAJ>.
- [19] R.C. Gonzalez and P.A. Wintz. *Digital image processing*. Applied mathematics and computation. Addison-Wesley Pub. Co., Advanced Book Program, 1977. ISBN 9780201030440. URL <http://books.google.ca/books?id=UQhEAQAAIAAJ>.
- [20] Branko Grünbaum. *Convex Polytopes*, volume Series: Graduate Texts in Mathematics, Vol. 221. First Edition Prepared with the Cooperation of Victor Klee, Micha Perles, and Geoffrey C. Shephardt, 1967.
- [21] Andi Gutmans, Stig Bakken, and Derick Rethans. *PHP 5 Power Programming (Bruce Perens' Open Source Series)*. Prentice Hall PTR, 2004.
- [22] Wilfred J Hansen. Introduction to user interface systems for hci developers and researchers. In *Conference Companion on Human Factors in Computing Systems*, pages 377–378. ACM, 1994.
- [23] Michael W Herman and Waldemar W Koczkodaj. A monte carlo study of pairwise comparison. *Information Processing Letters*, 57(1):25–29, 1996.
- [24] Włodzimierz Holsztynski and Waldemar W Koczkodaj. Convergence of inconsistency algorithms for the pairwise comparisons. *Information processing letters*, 59(4):197–202, 1996.
- [25] Huxtable.com. Class graphics2d. URL <http://groups.inf.ed.ac.uk/vision/STAVRAKAKIS/skinSpotTool2/skinSpotTool/GaussianFilter.html>.

- [26] Carlo Jacoboni and Lino Reggiani. The monte carlo method for the solution of charge transport in semiconductors with applications to covalent materials. *Reviews of Modern Physics*, 55(3):645, 1983.
- [27] Ramesh Jain, Rangachar Kasturi, and Brian G Schunck. *Machine vision*, volume 5. McGraw-Hill New York, 1995.
- [28] Kevin G Jamieson and Robert D Nowak. Active ranking using pairwise comparisons. *arXiv preprint arXiv:1109.3701*, 2011.
- [29] Tamar Kakiashvili, Waldemar W Koczkodaj, Phyllis Montgomery, Kalpdrum Passi, and Ryszard Tadeusiewicz. Assessing the properties of the world health organization's quality of life index. In *Computer Science and Information Technology, 2008. IMCSIT 2008. International Multiconference on*, pages 151–154. IEEE, 2008.
- [30] Tamar Kakiashvili, W W Koczkodaj, and Marc Woodbury-Smith. Improving the medical scale predictability by the pairwise comparisons method: Evidence from a clinical data study. *Computer methods and programs in biomedicine*, 105(3):210–216, 2012.
- [31] Pavel Karas and David Svoboda. *Algorithms for Efficient Computation of Convolution*, pages 179–208. InTech, Rijeka (CRO), 1st ed. edition, 2013. ISBN 978-953-51-0874-0. doi: <http://dx.doi.org/10.5772/3456>. URL <http://www.intechopen.com/books/design-and-architectures-for-digital-signal-processing/algorithms-for-efficient-computation-of-convolution>.
- [32] W W Koczkodaj. A new definition of consistency of pairwise comparisons. *Mathematical and computer modelling*, 18(7):79–84, 1993.
- [33] W W Koczkodaj. Statistically accurate evidence of improved error rate by pair-

- wise comparisons. *Perceptual and Motor Skills*, 82(1):43–48, 2013/05/24 1996. URL <http://dx.doi.org/10.2466/pms.1996.82.1.43>.
- [34] W W Koczkodaj. Statistically accurate evidence of improved error rate by pairwise comparisons. *Perceptual and motor skills*, 82(1):43–48, 1996.
- [35] W W Koczkodaj. Testing the accuracy enhancement of pairwise comparisons by a monte carlo experiment. *Journal of statistical planning and inference*, 69(1):21–31, 1998.
- [36] W W Koczkodaj and S J Szarek. On distance-based inconsistency reduction algorithms for pairwise comparisons. *Logic Journal of IGPL*, 18(6):859–869, 2010.
- [37] W W Koczkodaj and Wojciech Trochymiak. An expert system for construction tendering process. *ESDA 1996: Expert systems and AI; Neural networks*, 7:79, 1996.
- [38] Waldemar W Koczkodaj, Michael W Herman, and Marian Orłowski. Using consistency-driven pairwise comparisons in knowledge-based systems. In *Proceedings of the sixth international conference on information and knowledge management*, pages 91–96. ACM, 1997.
- [39] Waldemar W. Koczkodaj, Nicolas Robidoux, and Ryszard Tadeusiewicz. Classifying visual objects with the consistency-driven pairwise comparisons method. *MG&V*, 18(2):143–154, January 2009. ISSN 1230-0535. URL <http://dl.acm.org/citation.cfm?id=1643375.1643378>.
- [40] Waldemar W Koczkodaj, Artur Przelaskowski, and Kazimierz T Szopinski. Medical knowledge mining from image data: synthesis of medical image assessments for early stroke detection. *Machine Graphics & Vision International Journal*, 19(3):283–298, 2010.

- [41] Ming-Shin Kuo, Gin-Shuh Liang, and Wen-Chih Huang. Extensions of the multicriteria analysis with pairwise comparison under a fuzzy environment. *International Journal of Approximate Reasoning*, 43(3):268–285, 2006.
- [42] Tatsumi Kurosawa. Monte carlo calculation of hot electron problems. *J. Phys. Soc. Jpn.*, 21:424, 1966.
- [43] Joan Landes. The history of feminism: Marie-jean-antoine-nicolas de caritat, marquis de condorcet. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2010 edition, 2010. URL <http://plato.stanford.edu/archives/win2010/entries/histfem-condorcet/>.
- [44] Peter Loken, Amy Voytilla, Matt Bach, and Sivika Sirisanthana. The world of visual art and aesthetics: Its functions and limitations. URL <http://www.macalester.edu/academics/psychology/whathap/ubnrp/aesthetics/home.html>.
- [45] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.
- [46] L Mikhailov. Deriving priorities from fuzzy pairwise comparison judgements. *Fuzzy sets and systems*, 134(3):365–385, 2003.
- [47] Brad A. Myers. User-interface tools: Introduction and survey. *Software, IEEE*, 6(1):15–23, 1989.
- [48] Peter Nijkamp, Piet Rietveld, Henk Voogd, et al. *Multicriteria evaluation in physical planning*. North-Holland Amsterdam, 1990.
- [49] Oracle. Class convolveop, . URL <http://docs.oracle.com/javase/6/docs/api/java/awt/image/ConvolveOp.html>.
- [50] Oracle. Class graphics2d, . URL <http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/Graphics2D.html>.

- [51] Oracle. Class kernel, . URL <http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/image/Kernel.html>.
- [52] Oracle. Class math, . URL <http://docs.oracle.com/javase/6/docs/api/java/lang/Math.html>.
- [53] JI Pelaez and MT Lamata. A new measure of consistency for positive reciprocal matrices. *Computers & Mathematics with Applications*, 46(12):1839–1845, 2003.
- [54] Oskar Perron. Zur theorie der matrices. *Mathematische Annalen*, 64(2):248–263, 1907. ISSN 0025-5831. doi: 10.1007/BF01449896. URL <http://dx.doi.org/10.1007/BF01449896>.
- [55] John G Raiti and JD Daniels. Direct and pairwise area estimation of physical shapes through vision and touch 1, 2. *Perceptual and Motor Skills*, 114(2):391–396, 2012.
- [56] L. Saaty. *The analytic hierarchy process*. McGraw-Hill, 1980.
- [57] Thomas Saaty. A scaling method for priorities in hierarchical structures. *Journal of Mathematical Psychology*, 15(3):234 – 281, 1977. ISSN 0022-2496. doi: 10.1016/0022-2496(77)90033-5. URL <http://www.sciencedirect.com/science/article/pii/0022249677900335>.
- [58] Thomas Saaty. Scenarios and priorities in transport planning: Application to the sudan. *Transportation Research*, 11(5):343–350, 1977.
- [59] Thomas Saaty. How to make a decision: the analytic hierarchy process. *European journal of operational research*, 48(1):9–26, 1990.
- [60] Thomas Saaty. Fundamentals of multiple criteria decision making with the analytic hierarchy process, 1994.

- [61] Thomas Saaty. Decision making with the analytic hierarchy process. *International Journal of Services Sciences*, 1(1):83–98, 2008.
- [62] Thomas Saaty. Relative measurement and its generalization in decision making why pairwise comparisons are central in mathematics for the measurement of intangible factors the analytic hierarchy/network process. *RACSAM-Revista de la Real Academia de Ciencias Exactas, Fisicas y Naturales. Serie A. Matematicas*, 102(2):251–318, 2008.
- [63] David Sklar and Adam Trachtenberg. *PHP cookbook*. O’Reilly Media, Inc., 2003.
- [64] Stanley Smith Stevens et al. On the theory of scales of measurement, 1946.
- [65] Davida Teller and url = <http://ai.atoms.MITECS/Entry/teller.html> publisher = MIT John Palme, title = Psychophysics.
- [66] A Thomas. Heuristics for the generation of random polygons.
- [67] L L Thurstone. A law of comparative judgment. *Psychological Review*, 34(4): 273–286, 1927. ISSN 1939-1471(Electronic);0033-295X(Print). doi: 10.1037/h0070288.
- [68] Kristi Tsukida and Maya R Gupta. How to analyze paired comparison data. Technical report, DTIC Document, 2011.
- [69] Frederick M Waltz. Skipsm: separated-kernel image processing using finite-state machines. In *Proc. SPIE Conf. on Machine Vision Applications, Architectures, and Systems Integration III*, volume 2347, 1994.
- [70] Frederick M Waltz and John WV Miller. Efficient algorithm for gaussian blur using finite-state machines. In *Photonics East (ISAM, VVDC, IEMB)*, pages 334–341. International Society for Optics and Photonics, 1998.

- [71] Luke Welling and Laura Thomson. *PHP and MySQL Web development*. Sams Publishing, 2003.
- [72] Ding Xu. Jconcluder. URL <http://sourceforge.net/projects/concluder/>.
- [73] Bieke Zaman. Introducing a pairwise comparison scale for ux evaluations with preschoolers. In *Human-Computer Interaction–INTERACT 2009*, pages 634–637. Springer, 2009.
- [74] Y Zhai and R Janicki. On consistency in pairwise comparisons based numerical and non-numerical ranking. In *Proceedings of the International Conference on Foundations of Computer Science, FCS*, volume 2010, pages 183–186, 2010.

A Appendix

A.1 Java Code

A.1.1 Point Class

```
package Gaussian;

public class Point {
    private int x;
    private int y;

    public Point (int x, int y) //contstructor
    {
        this.x=x;
        this.y=y;
    }
    public int getX()
    {
        return x;
    }
    public void setX(int x)
    {
        this.x=x;
    }
    public int getY()
    {
        return y;
    }
    public void setY(int y)
```

```

    {
        this.y=y;
    }
    public String toString()
    {
        String context="";
        context="x="+x+" y="+y;
        return context;
    }
}

```

A.1.2 Pointlist Class

```

package Gaussian;
import java.util.ArrayList;
import java.util.Iterator;

public class PointList {

    private ArrayList<Point> pointList;
    private int pointNumber;

    public PointList()
    {
        pointList=new ArrayList<Point>();
    }
    public ArrayList<Point> getPointList()
    {
        return pointList;
    }
    public void setPointList(ArrayList<Point> pointList)
    {
        this.pointList=pointList;
    }

    public int getPointNumber()
    {
        return pointNumber;
    }
}

```

```

public void setPointNumber(int pointNumber)
{
    this.pointNumber=pointNumber;
}
public int[] getX()
{
    int[] x=new int[this.pointNumber];
    Iterator iter=this.pointList.iterator();
    Point temp; //container
    int i=0;
    while(iter.hasNext())
    {
        temp=(Point)iter.next();
        x[i]=temp.getX();
        i++;
    }
    return x;
}
public void addPoint(Point p){
    this.pointList.add(p);
}
public int[] getY()
{
    int[] y=new int[this.pointNumber];
    Iterator iter=this.pointList.iterator();
    Point temp; //container
    int i=0;
    while(iter.hasNext())
    {
        temp=(Point)iter.next();
        y[i]=temp.getY();
        i++;
    }
    return y;
}
}

```

A.1.3 RandomPoint Class

```

package Gaussian;

```

```

import java.util.Random;

public class RandomPoint {
    Point p[];
    int panelSize;

    public RandomPoint(int numberOfPoints, int panelSize
        /*height or width as it should be square*/)
    { //constructor
        p = new Point[numberOfPoints];
        this.panelSize=panelSize;
    }
    public Point[] getRandomPoints(){
        int x;
        int y;
        Random r = new Random();
        for (int i = 0; i<p.length; i++)
        {
            x = r.nextInt((panelSize-50)-50) + 50;
            y = r.nextInt((panelSize-50)-50) + 50;
            p[i] = new Point(x,y);
            System.out.println(p[i].toString());
        }
        return p;
    }
    //returns the values of x in an array
    public int[] getX() {
        int[] x;
        x = new int [p.length];
        for (int i = 0; i<x.length; i++)
        {
            x[i] = p[i].getX();
        }
        return x;
    }
    //returns the values of y in an array
    public int[] getY() {
        int[] y;
        y = new int [p.length];
        for (int i = 0; i<y.length; i++)
        {
            y[i] = p[i].getY();
        }
    }
}

```

```

        return y;
    }
}

```

A.1.4 Gaussian Class

```

package Gaussian;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.awt.image.ConvolveOp;
import java.awt.image.Kernel;
import java.io.File;
import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class Gaussian extends JFrame {

    private BlurPanel bPanel;
    private TextField tField;
    private JButton Gaussian;
    private JButton SaveImg;
    private JButton NewShape;
    static int max;
    static int i = 0;

    public Gaussian() {

        bPanel= new BlurPanel();
        this.add(bPanel);
    }
}

```



```

this.tField = new TextField(10);
Gaussian=new JButton("Update Gaussian Raduis(sigma=raduis/3)");
SaveImg=new JButton("Save Image");
NewShape=new JButton("New Shape");
Gaussian.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent arg0) {
            bPanel.setRadius(Integer.parseInt
            (tField.getText()));
        }
});

NewShape.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent arg0) {
        bPanel.NewShape();
    }
});

SaveImg.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae){
        try{
            if(bPanel.getRadius()<=0){

                SaveScreenShot(bPanel,"NoBlur.jpg");
            }
            if(bPanel.getRadius()>0){
                SaveScreenShot(bPanel,"WithBlur.jpg");
            }
        }
        catch(Exception e){
        }
    }
});

JPanel control1=new JPanel(new FlowLayout(FlowLayout.CENTER));
control1.add(tField);
control1.add(Gaussian);
control1.add(SaveImg);
control1.add(NewShape);
this.add(control1, BorderLayout.SOUTH);
this.setSize(new Dimension(600,600));// JFrame size
this.setLocationRelativeTo(null);
this.setTitle("Random Shapes");
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

```

```

}

//capture the image
public static BufferedImage getScreenShot(Component component){

    BufferedImage bi = new BufferedImage
        (500, 500, BufferedImage.TYPE_INT_BGR);
    component.paint(bi.getGraphics());

    return bi;
}

// save the image
public static void SaveScreenShot(Component component,
String filename) throws Exception{
    BufferedImage bi = getScreenShot(component);
    ImageIO.write(bi, "jpg", new File(filename));
}
}

/*-----BlurPanel-----*/

class BlurPanel extends JPanel{
    private int r=0;
    int min = 5;
    int max = 11;
    private BufferedImage image = null;

    private int randomNumberOfPoints = numberOfPoints(min, max);
    Point[] points = new Point[randomNumberOfPoints];
    RandomPoint rp = new RandomPoint(randomNumberOfPoints, 500
/*image size*/);
    int ix[];
    int iy[];
    public BlurPanel(){
        ix = new int[4];
        iy = new int[4];

        //white background
        ix[0]=0; iy[0]=0;

```

```

ix[1]=0; iy[1]=500;
ix[2]=500; iy[2]=500;
ix[3]=500; iy[3]=0;

rp.getRandomPoints();
}
int numberOfPoints(int min2, int max2) {

    return randomNumberOfPoints = min2 + (int)(Math.random()
        * ((max2 - min2) + 1));
    // Min + (int)(Math.random() * ((Max - Min) + 1));
}

@Override //paint and repaint
protected void paintComponent(Graphics mg) {
    super.paintComponent(mg);
    int[] x= rp.getX();
    int[] y= rp.getY();
    this.image= new BufferedImage
        (500,500,BufferedImage.TYPE_BYTE_GRAY);
    Graphics2D g=this.image.createGraphics();
    g.setColor(Color.WHITE);
    g.fillPolygon(ix, iy, 4);
    g.setColor(Color.BLACK);
    System.out.print("Number of points = ");
    System.out.println(randomNumberOfPoints);
    g.fillPolygon(x, y, points.length);
    if(this.r>0){
        image=this.gaussianFilter(this.r, true).filter(image, null);
        image=this.gaussianFilter(this.r, false).filter(image, null);
        System.out.print("Gaussian raduis= ");
        System.out.println(r);
    }
    g.dispose();
    mg.drawImage(image,0,0,null);
}

public void setRadius(int radius){
    this.r=radius;
    image=null;
    this.repaint();
}

public int getRadius()

```

```

    {

        return r;
    }
    public void NewShape(){
        image=null;
        repaint();
    }

    /**
     * Make a Gaussian blur kernel
     */

    public static ConvolveOp gaussianFilter
        (int radius, boolean horizontal){
        int size=radius*2+1;
        float data[]=new float[size];

        float sigma=radius/ 3.0f; // radius = 3*sigma
        float twoSigmaSquare=2*sigma*sigma;
        float root=(float)Math.sqrt(twoSigmaSquare*Math.PI);
        float total=0.0f;

        for(int i=-radius;i<radius;i++){
            float distance=i*i;
            int index=i+radius;
            data[index]=(float)Math.exp
                (-distance/twoSigmaSquare)/root;
            total+=data[index];
        }

        for(int i= 0;i<data.length;i++){
            data[i]/=total;
        }

        Kernel kernel=null;
        if(horizontal){
            kernel=new Kernel(size,1,data);
        }else{
            kernel=new Kernel(1,size,data);
            for (int i = 0; i < size/2; i++) {
                //to print the kernel vertical vector
            }
        }
    }
}

```

```

        System.out.printf("%9.4f ", data[i]);
        System.out.println();
    }
}
return new ConvolveOp(kernel,ConvolveOp.EDGE_NO_OP,null);
}

```

A.1.5 Image2Array Class

```

package Gaussian;

import java.awt.image.BufferedImage;
import java.awt.image.Raster;
import java.io.IOException;
import javax.imageio.ImageIO;
public class Image2Array
{
    int height, width;
    public Image2Array() {

//read image
public int [][] compress() throws IOException
{
    File file = new File("WithBlur.jpg");
    BufferedImage image = ImageIO.read(file);
    Raster image_raster = image.getData();
        int[][] original; // where we'll put the image

//get pixel by pixel
int[] pixel = new int[1];
int[] buffer = new int[1000];

original = new int[image_raster.getWidth()]
                [image_raster.getHeight()];

width = image_raster.getWidth();
height = image_raster.getHeight();
System.out.print("image size: ");
System.out.print(width);

```

```

System.out.print(" x ");
System.out.println(height);

for(int i = 0 ; i < image_raster.getWidth() ; i++)
for(int j = 0 ; j < image_raster.getHeight() ; j++)
{
    pixel = image_raster.getPixel(i, j, buffer);
    original[i][j] = pixel[0];
}

ReplacePixel(original, 200); //cutoff with threshold 200
return original;
}

int area=0;
// calculate the area
public int[][] ReplacePixel(int[][] src, int oldValue)
throws IOException
{
for(int i = 0 ; i < width; i++){
for(int j = 0 ; j < height ; j++)
{
    if (src[i][j] >= oldValue)
    {
        src[i][j] = 255;
    }

    if (src[i][j] < oldValue )
    {
        src[i][j] = 0;
    }
    if(src[i][j] != 255)
        // not all black pixels pure white
    {
        area++;
    }
}
}

}

System.out.println("Threshold= " + oldValue);
System.out.println("Area= " +area);

```

```

        Array2Image(src);
            return src;
    }

    public void Array2Image(int[][] src) throws IOException
    {
        BufferedImage theImage = new BufferedImage
            (width, height, BufferedImage.TYPE_BYTE_GRAY);
        for(int y = 0; y<width; y++){
            for(int x = 0; x<height; x++){

                int grayValue = src[x][y] << 16 |
                    src[x][y] << 8 | src[x][y];
                theImage.setRGB(x, y, grayValue);

            }
        }
        File outputfile = new File("afterCut.jpg");
        ImageIO.write(theImage, "jpg", outputfile);
    }

    public static void main(String[] args) throws IOException {
        Image2Array i2a = new Image2Array();
        i2a.compress();
    }
}

```

A.1.6 SquareUnit Class

```

package SquareUnit;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

```

```

import java.awt.image.BufferedImage;
import java.io.File;
import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
public class SquareUnit extends JFrame {

    private BlurPanel bPanel;
    private JButton SaveImg;
    static int max;
    static int i = 0;

    public SquareUnit() {
        bPanel= new BlurPanel();
        this.add(bPanel);
        SaveImg=new JButton("Save Image");
        SaveImg.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent ae){
                try{
                    SaveScreenShot(bPanel,"SquareUnit.jpg");
                }
                catch(Exception e){
                }}
        });

        JPanel control1=new JPanel(new FlowLayout
        (FlowLayout.CENTER));
        control1.add(SaveImg);

        this.add(control1, BorderLayout.SOUTH);
        this.setSize(new Dimension(600,600)); // jframe size
        this.setLocationRelativeTo(null);
        this.setTitle("Square Unit");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    //capture the image
    public static BufferedImage getScreenShot(Component component){

        BufferedImage bi = new BufferedImage (500, 500, BufferedImage.TYPE_INT_BGR);
        component.paint(bi.getGraphics());
        return bi;
    }
}

```



```

    }

    // save the image
    public static void SaveScreenShot(Component component, String filename) throws Exception{
        BufferedImage bi = getScreenShot(component);
        ImageIO.write(bi, "jpg", new File(filename));
    }
}

/*-----BlurPanel-----*/
class BlurPanel extends JPanel{
    private int r=0;
    private BufferedImage image = null;
    int ix[];
    int iy[];

    public BlurPanel(){

        ix = new int[4];
        iy = new int[4];

        //white background
        ix[0]=0; iy[0]=0;
        ix[1]=0; iy[1]=500;
        ix[2]=500; iy[2]=500;
        ix[3]=500; iy[3]=0;
    }

    @Override
    protected void paintComponent(Graphics mg) {

        super.paintComponent(mg);
        this.image= new BufferedImage(500,500,BufferedImage.TYPE_BYTE_GRAY);
        Graphics2D g=this.image.createGraphics();
        g.setColor(Color.WHITE);
        g.fillPolygon(ix, iy, 4);
        g.setColor(Color.BLACK);

        // unit square size is 1600 pixels or 40*40

        g.fillRect(230, 230, 40, 40);
        g.dispose();
        mg.drawImage(image,0,0,null);
    }
}

```

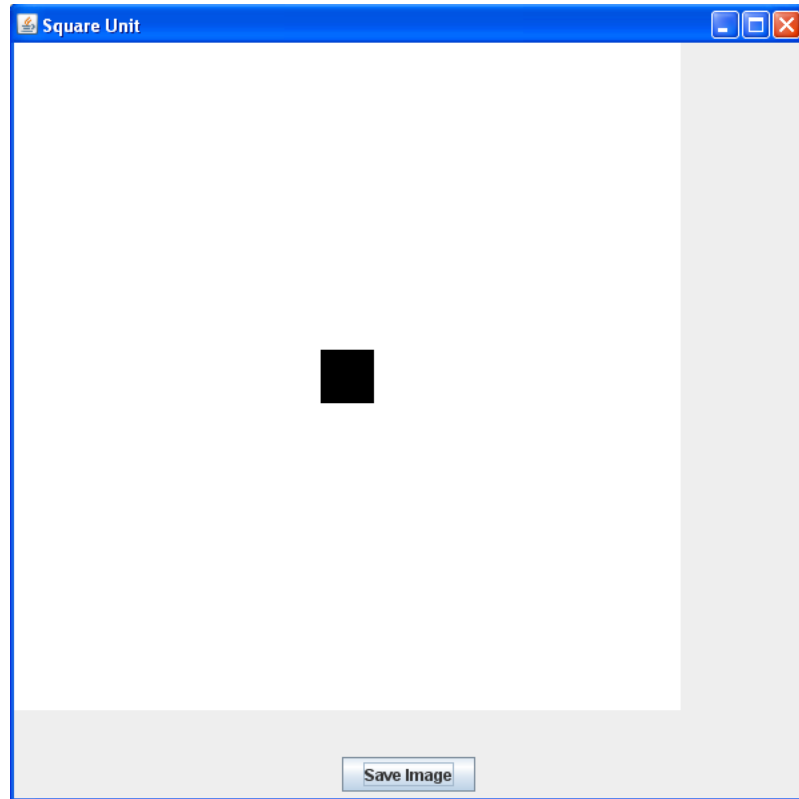


Figure A.1: Creating the unit square interface

```
}  
}
```

Generating the unit square example can be seen in Figure A.1.

A.2 PHP Code

In this section, I show parts of my php code for designing the questionnaire.

A.2.1 Consent form error checking and validating PHP code

This makes sure that the user can not get access to the questionnaire unless the consent form is filled up and signed properly.

```
function validate_page_one()  
{  
    $error = "";
```

```

if (!filter_var($_SESSION['email'], FILTER_VALIDATE_EMAIL) and $_SESSION['send_email'] === 'Yes')
{
    $error .=
    "<span class=\"error\">Email is invalid</span><br>";
}

if ($_SESSION['agree'] != 'agree')
{
    $error .= "<span class=\"error\">You must check the \"I agree\" checkbox to be able to continue</span>
    <br>";
}

if ($_SESSION['day'] === 'Day' or $_SESSION['month'] === 'Month' or $_SESSION['year'] === 'Year')
{
    $error .= "<span class=\"error\">You must select a valid \"Date\" to be able to continue</span><br>"
    ;
}

if (strlen($_SESSION['send_email']) == 0)
{
    $error .= "<span class=\"error\">Please answer the question \"Do you wish to be informed with the
    results of the study? (provide email)\"</span><br>";
}

return $error;
}

```

A.2.2 Experiment 1

A.2.2.1 Display the 70 shapes to allow the selection of the 5 shapes PHP code

```

<?php
    // show all shapes
    $file_number = 1;
    $file_name = '';
    echo "<table border=\"1\" align=\"center\">";
    echo "<tr>";
    for ( $i = 1; $i <= 7; $i++) //rows
    {
        for( $j = 1; $j <= 10; $j++) //columns

```

```

        {
            $file_name = "shapes/"
            . $file_number
            . "-WithCut.jpg";

            echo "<td align=\"center\">";
            echo "<input type=\"checkbox\" name=\"selected[]\" value=" ;
            echo $file_name;

            if (is_array($_SESSION['selected']) and in_array($file_name, $_SESSION['selected']))
            {
                echo ' checked = "checked"';
            }

            echo "><br /></td>";
            $file_number++;

        }

        echo "</tr>";
    }
    echo "</table>";

?>

```

A.2.2.2 Estimating the area in units PHP code

The following displays 1 of of the 5 pages where the user is asked to estimate the area of 1 of the selected shapes in units.

```

<html>
  <head><title>Random Shapes Study</title>
  <style type="text/css">
.error { color: #ff0000 }
.error { font-size:250% }
</style>
</head>
  <body>

```

```

<h3>Task 2 (1/5): please estimate the area of the shape you have just selected in units <b>(a black
square)</b> i.e. how many black
squares you need to fill up the shape. Note: Area can be a number with or without a decimal point. (Eg.
1, 6, 0.5, 1.25, 7.75, 11.39, ....)</h3>
<?php
echo "<table border='1' align='center'>";
echo "<tr>";
echo "<td>";

echo "<table border='1' align='center'>"; //table on the left side
echo "<tr>";
echo "<th>Shape</th>";
echo "<th>Unit</th>";
echo "</tr>";

$file_name = $_SESSION['5shapes'][0];
$shape_number = 0 + 1;

echo "<tr>";
echo "<td align='center' height='500' >";
echo "<img src='\$file_name' width='\$height_width[0]' height='\$height_width[0]' alt='
\$file_name' /><br />
\$shape_number</td>";
echo "<td><img src='blacksquare.jpg' alt='blacksquare.jpg' width='250' height
='250' /></td>";
echo "</tr>";

echo "</table>";
echo "</td>";
?>

<form action="<?php echo \$self ?>" method="post">
<td>
<!-- Table on right side -->
<table border="1" align="center">
<tr>
<th>Area estimation in units:</th>
</tr>
<tr align="center"> <td height="500">Area = <input class="input" type="text" name="area1"
value="<?php echo $_SESSION['area1'] ?>" size = 5> Units <input type="submit" name="page3b" value=
"Submit" />
</td>
</tr>

```

```

</table>
<!-- END -->
</td>
</tr>
</table>

<!-- END OF MAIN TABLE -->
    </form>
</body>
</html>

<?php
}

```

To check for errors, I used the following PHP code.

```

function validate_page_three_f()
{
$error = '';

    if ( !($_SESSION['area5'] > 0 and $_SESSION['area5']<=100) and (is_numeric ( $_SESSION['area5'])))
    {
        $error .= "<span class=\"error\">Area of shape is too large or too small to be true</span></br>";
    }

    if ( ($_SESSION['area5'] == '') or !(is_numeric ( $_SESSION['area5'])))
    {
        $error .= "<span class=\"error\">Area of shape should be a number</span></br>";
    }

    return $error;
}

```

A.2.2.3 Displaying shapes in pair for comparisons PHP code

```
<html>
```

```

<head><title>Random Shapes Study</title>
<style type="text/css">
    .error { color: #ff0000 }
.error { font-size:250% }
</style>
</head>

<body>
<h3><u>Task 3 (1/10):</u> Compare the following 2 shapes according to their areas</h3>
You may use numbers with or without fractions (such as 1, 3, 1.40, or 5.75) when comparing

<form action="<?php echo $self ?>" method="post">

<?php
    echo "<table border=\"1\" align=\"center\">";
    echo "<tr>";
    echo("&<p>");
    $file_name = $_SESSION['5shapes'][0];
    $shape_number = 0 + 1;
    echo "<td align=\"center\" height=\"500\" width=\"500\">";
    echo "<img src=\"$file_name\" width=\"$height_width1\" height=\"$height_width1\" alt=\"$file_name
        \"/><br />";

    </td>";
    $file_name = $_SESSION['5shapes'][1];
    $shape_number = 1 + 1;
    echo "<td align=\"center\" height=\"500\" width=\"500\" >";
    echo "<img src=\"$file_name\" width=\"$height_width2\" height=\"$height_width2\" alt=\"$file_name
        \"/><br />";
    </td>";
    echo("&<p>");
    echo "</tr>";
    echo "</table>";
    echo "</form>";
?>

<?php
    if ($error)
    {
        echo "<p>$error</p>\n";
    }
?>

<form action="<?php echo $self ?>" method="post">

```

```

<table border="1" align="center">
<tr>
<td>
<input type="radio" name="compare" value="a" <?php if ($_SESSION['compare'] === "a")
    {
        echo 'checked = "checked"';
    }?>>
Shapes are approximately <b>equal.</b>

</td><td> <input type="radio" name="compare" value="b" <?php if ($_SESSION['compare'] === "b")
    {
        echo 'checked = "checked"';
    }?>> <b>Shape 1</b> is [ <input class="input" type="text" name="compare_area_b"
    size = 1 > ] times bigger.
</td><td> <input type="radio" name="compare" value="c" <?php if ($_SESSION['compare'] === "c")
    {
        echo 'checked = "checked"';
    }?>> <b>Shape 2</b> is [ <input class="input" type="text" name="compare_area_c"
    size = 1 > ] times bigger.
</td><td>
<input type="submit" name="page4" value="Submit" />
</td>
</tr>
</table>
</body>
</html>
<?php
}

```

To validate and check for user input errors I used the following PHP code

```

function validate_page_four()
{
    $error = "";
    if (strlen($_SESSION['compare']) == 0)
    {
        $error .= "<span class=\"error\">Please select a, b, or c</span></br>";
    }
    if (($_SESSION['compare_area_b'] == '' and $_SESSION['compare'] == 'b'))
    {

```



```

        $error .= "<span class=\"error\">Please input approximately how many times shape 1 is larger
                than shape 2</span></br>";
    }
    if ( ($_SESSION['compare_area_c'] == '' and $_SESSION['compare']=='c'))
    {
        $error .= "<span class=\"error\">Please input approximately how many times shape 2 is larger
                than shape 1</span></br>";
    }
    if ( ($_SESSION['compare'] == 'c') and ($_SESSION['compare_area_c'] < 0 or $_SESSION['compare_area_c']
        ]>=25))
    {
        $error .= "<span class=\"error\">Estimating how many times is one shape larger than the other is too large
                or too small to be true</span></br>";
    }
    if ( $_SESSION['compare'] == 'b' and $_SESSION['compare_area_b'] < 0 or $_SESSION['compare_area_b']
        ]>=25)
    {
        $error .= "<span class=\"error\">Estimating how many times is one shape larger than the other
                is too large or too small to be true</span></br>";
    }
    return $error;
}

```

A.2.3 Experiment 2

Most the php code used for Experiment 2 is the same as the one used in experiment 1 except for the ordering or ranking from largest to smallest page and the pairwise comparisons 10 pages.

A.2.3.1 Shape ranking PHP code

Here is part of the php code I use to display the "order from largest to smallest" page.

```

function display_page_rank($error)
{

    $self = $_SERVER['PHP_SELF'];
?>

```

```

<html>
<head><title>Random Shapes Study</title>
<style type="text/css">
.error { color: #ff0000 }
.error { font-size:250% }
</style>
</head>
<body>
<h2> It is very important to make sure to input accurate data as much as possible otherwise inconsistent
data will be produced. Thank you for your patience and understanding.</h2>
<h3> Here are the shapes that you have just selected. <!--ranked from the largest to the smallest. Do
you approve or disapprove the shapes and thier ranking?--></h3>
<h3> Rank largest to smallest from 1 to 5, where the largest shape gets the value of 1, the smallest gets
the value of 5 and the shapes in between get the values of (2, 3, 4) respectively</h3>

<form action="<?php echo $self ?>" method="post">
<?php
// to sort the area
$to_sort= array($_SESSION['scaled_area1'], $_SESSION['scaled_area2'], $_SESSION['scaled_area3'],
$_SESSION['scaled_area4'], $_SESSION['scaled_area5']);
asort($to_sort);

\\shuffle to randomise the order
$shuffle_to_rank=array("<td width="250"></td>",
" <td width="250"></td>",
" <td width="250"></td>",
" <td width="250"></td>",
" <td width="250"></td>");

$array_to_shuffle=array("<td><input class="input" type="text" name="rank1" size = 1></
td>$shuffle_to_rank[0]",
" <td><input class="input" type="text" name="
$rank2\" size = 1></td>$shuffle.to_rank[1]",

```

```

        " <td><input class=\"input\" type=\"text\" name=\"
            $rank3\" size = 1></td>$shuffle_to_rank[2]\",
        " <td><input class=\"input\" type=\"text\" name=\"
            $rank4\" size = 1></td>$shuffle_to_rank[3]\",
    " <td><input class=\"input\" type=\"text\" name=\"$rank5\"
        size = 1></td>$shuffle_to_rank[4]");

    shuffle($array_to_shuffle);

?>
<table border="1">
<tr>
<?php echo $array_to_shuffle[$randomis[0]] . $array_to_shuffle[$randomis[1]] . $array_to_shuffle[$randomis[2]]
    . $array_to_shuffle[$randomis[3]] . $array_to_shuffle[$randomis[4]];?>
</tr>
</table>
<p align="center">
Rank from largest to smallest, from 1 to 5. Where the largest shape gets the value of 1, the smallest gets the
    value of 5</p>
    <p align="center">
        <input type="submit" name="pagerank1" value="Submit" /></p>

<style type="text/css">
.error { color: #ff0000 }
.error { font-size:250% }
</style>
<?php //to show errors.
if ($error)
{
    echo "<p align=\"center\">$error</p>\n";
}
?>

</form>

</body>
</html>

<?php
}

```

To check if the user has the correct ordering, I used the following function that will return an error if the ordering is wrong.

```
function validate_page_rank()
{
$error="";
if (( $_SESSION['rank1'] <1 or $_SESSION['rank1']>5 or is_numeric( $_SESSION['rank1'] or $_SESSION['rank1'] == ''
))
or ( $_SESSION['rank2'] <1 or $_SESSION['rank2']>5 or is_numeric( $_SESSION['rank2'] or $_SESSION['
rank2'] == ''))
or ( $_SESSION['rank3'] <1 or $_SESSION['rank3']>5 or is_numeric( $_SESSION['rank3'] or $_SESSION['
rank3'] == ''))
or ( $_SESSION['rank4'] <1 or $_SESSION['rank4']>5 or is_numeric( $_SESSION['rank4'] or $_SESSION['
rank4'] == ''))
or ( $_SESSION['rank5'] <1 or $_SESSION['rank5']>5 or is_numeric( $_SESSION['rank5'] or $_SESSION['
rank5'] == '')))
{
$error .= "<span class=\"error\">You have inputted invalid values; please make sure that you
input numbers between 1 and 5. Rank largest to smallest from 1 to 5, where the largest shape
gets the value of 1, the smallest gets the value of 5 and the shapes in between get the values
of (2, 3, 4) respectively</span></br>";
return $error;
}

if( ($_SESSION['rank1'] != 1)
or ($_SESSION['rank2'] != 2)
or ($_SESSION['rank3'] != 3)
or ($_SESSION['rank4'] != 4)
or ($_SESSION['rank5'] != 5))
{
$error .= "<span class=\"error\">Shape are not ranked correctly, please start over again by
clicking the link \"Start over again\" below. Rank largest to smallest from 1 to 5, where the
largest shape gets the value of 1, the smallest gets the value of 5 and the shapes in between
get the values of (2, 3, 4) respectively</span></br>";
}

return $error;
}
```

A.2.3.2 Displaying shapes in pair for comparisons PHP code

As the design of the pairwise comparisons 10 pages in experiment 2 have changed, I used the following code to display it.

```
<html>
  <head><title>Random Shapes Study</title>
  <style type="text/css">
    .error { color: #ff0000 }

  </style>
  </head>
  <body>

  <h3>Task 3 (1/10): Compare the following 2 shapes according to their areas. You may select one option only.
  You may use numbers with or without fractions (such as 1, 3, 1.40, or 5.75) when comparing</h3>

  <form action="<?php echo $self ?>" method="post">
    <p align="center">
      Click "Disapprove" button only if you think that shape 1 is not larger than shape 2
    <input type="submit" name="pagerank2" value="Disapprove"/></p>
  </p>

  <?php
  echo "<table border=\"1\" align=\"center\">";
  echo "<tr>";
  echo "<td align=\"center\">";
  echo "<b>Shape 1</b>";
  echo "</td>";
  echo "<td align=\"center\">";
  echo "<b>Shape 2</b>";
  echo "</td>";
  echo "<tr>";
  echo("</p>");
  $file_name = $_SESSION['5shapes'][0];
  $shape_number = 0 + 1;
  echo "<td align=\"center\" height=\"500\" width=\"500\">";
  echo "<img src=\"$file_name\" width=\"$height_width1\" height=\"$height_width1\" alt=\"$file_name
  \"/> <br />
  </td>";
  $file_name = $_SESSION['5shapes'][1];
```

```

    $shape_number = 1 + 1;
    echo "<td align=\"center\" height=\"500\" width=\"500\" >";
    echo "<img src=\"\$file_name\" width=\"\$height_width2\" height=\"\$height_width2\" alt=\"\$file_name
        \"/> <br />
    </td>";
    echo("</p>");
    echo "</tr>";
    echo "</table>";
    echo "</form>";
?>
<?php
    if ($error)
    {
        echo "<p align=\"center\">$error</p>\n";
    }
?>
<form action="<?php echo \$self ?>" method="post" >
<table border="1" align="center" >
<tr>
<!--<td>
<input type="checkbox" name="compare" value="a" >
Shapes are approximately <b>equal.</b>

    </td>--><td> <b>Shape 1</b> is [ <input class="input" type="text" name="compare_area_b"
    size = 1 > ] times <b>larger.</b>

    </td><td>
<input type="submit" name="page4" value="Submit" />
<b> 1 of 10</b></td>
</tr>
</table>
</body>
</html>

<?php
}

```

A.3 MySQL code

Figure A.2 shows a bird-eye view of my database.

Server: localhost ▶ Database: graddb

Structure SQL Search Query Export Import Operations

Table	Action				Records	Type	Collation	Size	Overhead
<input type="checkbox"/> COMPARE_SHAPES					66	MyISAM	latin1_swedish_ci	99.7 KIB	16.3 KIB
<input type="checkbox"/> exp2COMPARE_SHAPES					52	MyISAM	latin1_swedish_ci	45.1 KIB	-
<input type="checkbox"/> exp2height_width					46	MyISAM	latin1_swedish_ci	4.1 KIB	-
<input type="checkbox"/> exp2randomis					0	MyISAM	latin1_swedish_ci	1.0 KIB	-
<input type="checkbox"/> exp2selected					46	MyISAM	latin1_swedish_ci	7.6 KIB	-
<input type="checkbox"/> exp2SHAPES					140	MyISAM	latin1_swedish_ci	5.4 KIB	-
<input type="checkbox"/> exp2sort_key					46	MyISAM	latin1_swedish_ci	3.2 KIB	-
<input type="checkbox"/> height_width					87	MyISAM	latin1_swedish_ci	7.1 KIB	-
<input type="checkbox"/> NEW_AREA					94	MyISAM	latin1_swedish_ci	10.7 KIB	-
<input type="checkbox"/> Scale_to_Equal					70	MyISAM	latin1_swedish_ci	3.2 KIB	-
<input type="checkbox"/> selected					89	MyISAM	latin1_swedish_ci	13.8 KIB	-
<input type="checkbox"/> SHAPES					70	MyISAM	latin1_swedish_ci	3.2 KIB	-
12 table(s)					806	MyISAM	latin1_swedish_ci	204.0 KIB	16.3 KIB

Check All / Uncheck All / Check tables having overhead With selected: ▼

Print view Data Dictionary

Database: graddb (12)

- COMPARE_SHAPES
- exp2COMPARE_SHAPES
- exp2height_width
- exp2randomis
- exp2selected
- exp2SHAPES
- exp2sort_key
- height_width
- NEW_AREA
- Scale_to_Equal
- selected
- SHAPES

Figure A.2: The MySQL overall view

A.3.1 Experiment 1 MySQL code

```
--  
-- Table structure for table `SHAPES`  
--  
  
CREATE TABLE IF NOT EXISTS `SHAPES` (  
  `number` varchar(100) NOT NULL,  
  `area` varchar(100) NOT NULL  
) ENGINE=MyISAM DEFAULT CHARSET=latin1;  
  
-----  
  
--  
-- Table structure for table `COMPARE_SHAPES`  
--  
  
CREATE TABLE IF NOT EXISTS `COMPARE_SHAPES` (  
  `Serial_Number` int(25) NOT NULL AUTO_INCREMENT,  
  `Agree` varchar(10) NOT NULL,  
  `Server_Time_Date` varchar(100) NOT NULL,  
  `Session` varchar(100) NOT NULL,  
  `User_Date` varchar(50) NOT NULL,  
  `Send_Result` varchar(100) NOT NULL,  
  `Shape_1_Area` varchar(100) NOT NULL,  
  `Shape_2_Area` varchar(100) NOT NULL,  
  `Shape_3_Area` varchar(100) NOT NULL,  
  `Shape_4_Area` varchar(100) NOT NULL,  
  `Shape_5_Area` varchar(100) NOT NULL,  
  `scaled_area1` varchar(100) NOT NULL,  
  `scaled_area2` varchar(100) NOT NULL,  
  `scaled_area3` varchar(100) NOT NULL,  
  `scaled_area4` varchar(100) NOT NULL,  
  `scaled_area5` varchar(100) NOT NULL,  
  `Compare_1` varchar(100) NOT NULL,  
  `Compare_2` varchar(100) NOT NULL,  
  `Compare_3` varchar(100) NOT NULL,  
  `Compare_4` varchar(100) NOT NULL,  
  `Compare_5` varchar(100) NOT NULL,  
  `Compare_6` varchar(100) NOT NULL,
```



```

`Compare_7` varchar(100) NOT NULL,
`Compare_8` varchar(100) NOT NULL,
`Compare_9` varchar(100) NOT NULL,
`Compare_10` varchar(100) NOT NULL,
`time1` varchar(100) NOT NULL,
`time2` varchar(100) NOT NULL,
`time3` varchar(100) NOT NULL,
`total_time` varchar(100) NOT NULL,
PRIMARY KEY (`Serial_Number`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=103 ;

```

```

--
-- Table structure for table `height_width`
--

```

```

CREATE TABLE IF NOT EXISTS `height_width` (
  `Session` varchar(100) NOT NULL,
  `height_width1` varchar(100) NOT NULL,
  `height_width2` varchar(100) NOT NULL,
  `height_width3` varchar(100) NOT NULL,
  `height_width4` varchar(100) NOT NULL,
  `height_width5` varchar(100) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

```

A.3.2 Experiment 2 MySQL code

```

--
-- Table structure for table `exp2SHAPES`
--

```

```

CREATE TABLE IF NOT EXISTS `exp2SHAPES` (
  `number` varchar(100) NOT NULL,
  `area` varchar(100) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

```

--

```

-- Table structure for table `exp2COMPARE_SHAPES`
--

CREATE TABLE IF NOT EXISTS `exp2COMPARE_SHAPES` (
  `Serial_Number` int(25) NOT NULL AUTO_INCREMENT,
  `Agree` varchar(10) NOT NULL,
  `Server_Time_Date` varchar(100) NOT NULL,
  `Session` varchar(100) NOT NULL,
  `User_Date` varchar(50) NOT NULL,
  `Send_Result` varchar(100) NOT NULL,
  `Shape_1_Area` varchar(100) NOT NULL,
  `Shape_2_Area` varchar(100) NOT NULL,
  `Shape_3_Area` varchar(100) NOT NULL,
  `Shape_4_Area` varchar(100) NOT NULL,
  `Shape_5_Area` varchar(100) NOT NULL,
  `scaled_area1` varchar(100) NOT NULL,
  `scaled_area2` varchar(100) NOT NULL,
  `scaled_area3` varchar(100) NOT NULL,
  `scaled_area4` varchar(100) NOT NULL,
  `scaled_area5` varchar(100) NOT NULL,
  `Compare_1` varchar(100) NOT NULL,
  `Compare_2` varchar(100) NOT NULL,
  `Compare_3` varchar(100) NOT NULL,
  `Compare_4` varchar(100) NOT NULL,
  `Compare_5` varchar(100) NOT NULL,
  `Compare_6` varchar(100) NOT NULL,
  `Compare_7` varchar(100) NOT NULL,
  `Compare_8` varchar(100) NOT NULL,
  `Compare_9` varchar(100) NOT NULL,
  `Compare_10` varchar(100) NOT NULL,
  `time0` varchar(100) NOT NULL,
  `time1` varchar(100) NOT NULL,
  `time2` varchar(100) NOT NULL,
  `time3` varchar(100) NOT NULL,
  `total_time` varchar(100) NOT NULL,
  `Disapprove` varchar(100) NOT NULL DEFAULT 'FALSE',
  PRIMARY KEY (`Serial_Number`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=53 ;

```

--

```
-- Table structure for table `exp2height_width`  
--
```

```
CREATE TABLE IF NOT EXISTS `exp2height_width` (  
  `Session` varchar(100) NOT NULL,  
  `height_width1` varchar(100) NOT NULL,  
  `height_width2` varchar(100) NOT NULL,  
  `height_width3` varchar(100) NOT NULL,  
  `height_width4` varchar(100) NOT NULL,  
  `height_width5` varchar(100) NOT NULL  
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```