Generation of Similar Images Using Bag Context Shape Grammars

Blessing Ogechi ***Ogbuokiri***

(1545654)

A thesis submitted to the
School of Computer Science and Applied Mathematics,
Faculty of Science, University of the Witwatersrand, Johannesburg,
in fulfilment of the requirements for the Degree of Doctor of Philosophy
in Computer Science.

Supervisors

Dr. Mpho Raborife

and

Prof. Raseelo Moitsheki

April 2020

Declaration

I, Blessing Ogechi **Ogbuokiri**, hereby declare the contents of this research thesis to be my work unless otherwise explicitly referenced. This research is submitted for the degree of Doctor of Philosophy in Computer Science at the University of the Witwatersrand, Johannesburg. This work has not been submitted to any other university, nor for any other degree.

Signed: 

Date: 10 April 2020

"Success is imparting the world with the investment of your personality."

Chris Oyakhilome

"If people did not do silly things, nothing intelligent would ever get done."

Ludwig Wittgenstein

Abstract

Picture grammars have become relevant in generating similar images, because of the increasing need for applications which require such images. Shape grammars are one form of picture grammars. Most shape grammar systems generate an infinite number of images without considering the similarity between them. This is mainly the result of failure to control the application of their rules during derivation. A shape grammar class is therefore imperative not only to generate similar images but also to control the rules applied during derivation. This research demonstrated new approach to generating an infinite number of images that are similar in a controlled manner with the use of a new shape grammar class, called Bag Context Shape Grammars. Bag Context Shape Grammar is also context free, but the application of a rule is controlled by a special vector of integers called the bag, which changes during a derivation. This research goes on to prove that every puzzle grammar, with permitting features, can be converted to Bag Context Shape Grammars. These were further demonstrated in the conversion process. Additionally, this research considered a set of images and demonstrated how Bag Context Shape Grammars can generate a set of images with fewer variable and rules. Then, the different methods these Bag Context Shape Grammars can generate an infinite number of similar images in a controlled manner were demonstrated. As proof of concept, the theories presented in Bag Context Shape Grammars, are implemented in a prototype software called Bag Context Shape Grammar Interpreter which is used to generate images. Images generated by Bag Context Shape Grammars are examined using a Spatial Colour Distribution Descriptor, a content based image retrieval method, used to mathematically check the similarity of images. Finally, the similar images generated using Bag Context Shape Grammars were implemented into a prototype visual password scheme as distractors. The outcome of this study shows that Bag Context Shape Grammars are suitable for the generation of an infinite number of similar images. These images can be useful in various application areas where similar images are needed, such as, distractors for visual password schemes.

Keywords: Image Generation, Formal Language, Shape Grammar, Bag Context.

Publications

Journals

1. **Ogbuokiri**, B.O. and Raporife, M. (2020). *Bag Context Shape Grammars.* IAENG International Journal of Computer Science, 47(1), 75–86. Scopus.

2. **Ogbuokiri**, B.O. and Raporife, M. (2020). *Bag Context Shape Grammar Interpreter: From Theory to Usable Software.* Computer-Aided Design & Applications, 17(3), 458-474. Taylor and Francis.

Conferences

1. **Ogbuokiri**, B.O. and Raborife, M. (2019). *Visual Password Scheme Using Bag Context Shape Grammars.* Presented at the 19th International Conference on Intelligent Systems Design and Applications (ISDA 2019), 3–5 December, Online, Springer Verleg.

2. **Ogbuokiri**, B.O. and Raborife, M. (2020). *The Similarity of Images Generated By Bag Context Shape Grammars.* In Proceedings of the 30th Annual Symposium of the Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech 2020), 29–31 January, Cape Town, South Africa, IEEE Xplore.

Posters

1. **Ogbuokiri**, B.O. (2016). *Visual Password Scheme Using Bag Context Shape Grammars.* Proceedings of the M & D Symposium of the South African Institute of Computer Scientists and Information Technologists (SAICSIT), 26–28 September, University of Johannesburg, South Africa.

2. **Ogbuokiri**, B.O. (2018). *Graphical Password Scheme Using Bag Context Shape Grammars.* 9th Post Graduate Cross Faculty Symposium, 29–30 October, University of the Witwatersrand, Johannesburg, South Africa.

Acknowledgements

"Worthy are you, O Lord our God, to receive glory and honour and power. . . ."

(Revelation 4:11)

I would like to give my sincere gratitude to my supervisors, Dr. Mpho Raborife and Prof. Roseelo Moitsheki for accepting to work with me as their student and for agreeing to continue to finish this research with me. To Prof. Sigrid Ewert and Prof. Ian Sanders for suggesting this topic and for their editorial comments that have made me a better writer. I am grateful to Prof. Sigrid Ewert for financial support during my first year of this research.

To the authors whose materials I used in this research. Your works provided me with a whole lot of understanding of the research area.

To my entire family members for their unalloyed support towards my career. In particular, my lovely mother, Lolo Ugoeze Harriet Ogbuokiri. Thank you all.

To my colleagues, George Obaido, Kehinde Arubela, Nuru Jingili, Liemiso Mpota, Francis Egbelowo, Modupe Abiodu, Senyo Cudjoe, Francisca Nweze, Okechi Achilonu, Zandile Sityata, Sindisiwe Zulu, and Mbulelo Ubisi who in one way or another supported and motivated me during this research.

For their support during this PhD programme, I am grateful to the following people, in no particular order, Prof. and Dr. (Mrs) Chukwuedozie Ajaero, Prof. Benjamin Rosman, Prof. Abejide Ade–Ibijola, Dr. Pravesh Ranchod, Dr. Ikechukwu Achilonu, Marike Kluyts, Michelle van Heerden, Audrey Magagula, Nkaba Senne, Henry Ugwu, Kizito Anyanwu, and Michael Azu.

I want to thank all the administrative staff of the Faculty of Science, as well as the School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg, especially and in no particular order, Mpumi Mnqapu, Pulane Ditlhake, Mohsin Desai, Kgomotso Monyepote, Kebadiretse Mosiane and Precious Shabalala for their support.

My hearty thanks to the organisations who have provided me with funding for this research as follows:

- CSIR–DST scholarship for the year 2017–2019
- University of the Witwatersrand, Johannesburg, Postgraduate Merit Award (PMA) for the year 2017–2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Image generation in data visualization can be defined as the process of representing data or information in visual form. Image generation, a key subject in computer science and mathematics, has been applied to various fields, such as medicine, document processing, generating of clothing, as well as in video games [30, 41]. Moreover, images can be generated syntactically or non-syntactically. Due to the relevance of image generation in computer science, new methods of syntactically generating images have emerged in the past few years. The use of formal grammars, in generating syntactic or abstract images, is one example of the new methods of image generation [33, 42, 163].

Formal grammars can be classified into four main classes: unrestricted, context sensitive, context free and regular grammars [24]. The class of formal grammars that is of interest to this research is the context free grammars. In formal languages, grammar is considered context free when its rules can be applied regardless of the context (circumstance or condition). The context free grammar is used for generating string languages. However, the approach of context free grammar has been applied in other grammar developments for the generation of image galleries. These classes of context free image generating grammars are generally called picture grammars.

Of equal interest to this research is a form of picture grammar called the shape grammar. A shape grammar is a class of context free grammars, used in formal languages, that uses spatial rules to generate images. Shape grammars have been studied for over two decades [85, 86, 143, 144, 145, 146, 148, 149], and have been applied mostly in architectural building plans, decorative art, engineering and product designs. The purpose of shape grammars is to transform or to produce shapes (designs) from the premise of an

existing one [100]. Moreover, there are three processes of transforming shape grammars [88]. They are: addition, subtraction, and substitution. Addition in shape grammars is where rules are added to the grammars. Subtraction is where a rule is deleted from the grammars, while substitution involves changing of the constructive mechanism of the grammar rule [87]. These shape grammar transformation processes can be used to define different design styles during a generation process [4, 87].

The use of shape grammars for image generation does not always produce images in a regulated manner. Some generate an infinite number of images without considering their similarities. As a result, these grammars may generate images that are entirely different from each other, or they may generate images, that are too small in some parts, using the same grammar. This is because they are context free and as such, result in the application of a rule that is not controlled. Furthermore, shape grammar rules are designed with a specific shape in mind and it may not be possible to use any other shape to implement images during rendering and after derivation.

However, research has shown that control or regulation can be added to the context free grammar rules and to the derivation process using different techniques such as bag context [35], random context [42], etc. This research focuses on bag context which is essentially a notion to control when a context free grammar rule can be applied during derivation. This type of technique when added to a shape grammar rule is called Bag Context Shape Grammar (BCSG).

We finally define BCSG which is used for the generation of an infinite number of similar images in a controlled manner. Bag Context Shape Grammars (BCSGs) belongs to the class of grammars which make use of context to regulate the application of context free rules. In BCSGs, the application of a context free, shape grammar rule, is controlled by a special vector of integers called the bag that changes during the derivation of an image. A rule in the grammar can only be applied if the bag at that point is within the range defined by the lower and upper limits of the rule. When a rule is applied, it causes the bag's values to change by adding the bag adjustment, which is part of the rule.

As a result, this research has proven that every puzzle grammar with permitting features can be converted to a BCSGs. These were demonstrated through the conversion process which was illustrated with examples. A set of images were considered which demonstrated how BCSGs can generate a set of images with fewer variables and rules. Thereafter, it was demonstrated that these BCSGs can generate an infinite number of images which are similar, but not identical in a controlled manner. As proof of concept, the theories presented in BCSGs are implemented in a prototype software called Bag Context Shape Grammar Interpreter (BCSGI) which was used to generate images.

The similarity of the images generated by BCSGs is examined using the Spatial Colour Distribution Descriptor (SpCD). The SpCD is a content based image retrieval method used to mathematically check the similarity of images. Finally, the similar images generated using BCSGs were implemented into a prototype visual password scheme. The outcome of this research shows that BCSGs are suitable for the generation of an infinite number of similar images. These can be useful where similar images are needed, such as visual password schemes, decorative arts, etc.

1.2 Research Motivation

Similar image generation has become relevant as the need for its use in areas like visual password schemes as distractors have increased. A distractor image can be defined as a similar image, in a visual password scheme that diverts the attention of a user from the selected image during login, or provides incorrect image choices that looks similar to the selected image [109].

Research has shown that picture grammars, such as shape grammars, can generate similar images [43, 76, 77]. Most shape grammar systems fail to generate only similar images in a controlled manner because they require users to generate different images and to manually select similar images. Therefore, it is difficult to determine if the image to be generated will be similar to a selected image.

Existing research using shape grammars are not able to regulate the ways in which similar images can be generated in a controlled manner. Consequently, some of these grammars generate images that are entirely different from each other or, they generate images where some parts are too small for the human eye to see. Therefore, a regulation can be added to the shape grammar rules. This will result in the derivation of the generation of an infinite number of images that are similar to a selected image. This research used bag context shape grammars to generate similar images.

1.3 Research Objectives

The main objective of this research was to propose a new approach of generating an infinite number of similar images in a controlled manner using BCSGs. Therefore, this research accomplished the following specific objectives:

1. defined the theoretical concept and implementation of a new additive shape grammar type that can allow any shape to be used to implement the images during rendering and after a derivation,

2. defined bag context shape grammars and proved that every puzzle grammar with permitting features can be converted to a bag context shape grammar,

3. determined different methods to use bag context shape grammars to make similar images,

4. implemented the theories in bag context shape grammars into a prototype software called bag context shape grammar interpreter,

5. checked the capability of the bag context shape grammars to generate similar images using a mathematical model called SpCD,

6. implemented the similar images generated by bag context shape grammars in a prototype visual password scheme as distractors.

1.4 Research Methodology

This research was conducted as follows:

1. We studied and reviewed related work to identify the gap as regards to picture grammars and shape grammars in particular.

2. We also defined the methods and theoretical concepts of additive shape grammars and BCSGs in relation to similar image generation.

3. Then we proved that every puzzle grammar with permitting features can be converted to a BCSG.

4. The BCSGs methods of generating similar images were implemented.

5. The methods and theoretical concepts of BCSGs were implemented into a prototype software called BCSG interpreter for the generation of similar images.

6. The similarity of the images generated by BCSGs were checked using the mathematical measure of image similarity, SpCD.

7. Finally, the similar images generated using BCSGs were implemented into a prototype visual password scheme as distractors.

1.5 Research Result

The result of this research validated the claim that BCSGs can generate similar images in a regulated manner, see Chapters 3 and 4. These similar images generated by BCSGs were implemented in a prototype visual password scheme as distractors, see Chapter 7.

A usability study of the prototype visual password scheme, to ascertain if users can remember and recognise their selected image password in the midst of distractor images, immediately after enrolment and one week later revealed that, 72 (64.9%) of the 111 users were able to login immediately, while 65 (58.6%) of the total number of users were able to log into the system successfully after one week.

The Chi-square (X^2) test statistic was used to compute the relationship or an association between a user's ability to log into the system immediately after enrolment and one week later. The results therefore revealed that there is a statistically significant ($X^2 = 10.18675$, p-value=0.00156) relationship, or an association between a user's ability to log into the system immediately after enrolment and one week later. This research therefore concludes, based on these results that there is an association between the two variables. That is to say, those who remembered their image passwords immediately as well as one week later was not by chance. A more detailed explanation can be found in Chapter 7.

1.6 Research Contribution

The main contribution of this research to the broader body of knowledge is through the introduction and implementation of a new approach to similar image generation using Bag Context Shape Grammars (see Chapters 3 and 4). Although, existing shape grammars can generate similar images, many lacked methods for generating only images similar to a given image.

Another contribution of this research is the proof of concept of the theories presented in BCSGs being implemented in a prototype software called BCSG interpreter (see Chapter 5). This tool can aid in the teaching of shape grammar or BCSG implementation in formal language classes or, in higher learning in general. It can also be a framework for researchers or developers who wish to contribute new ideas to the study of similar image generation, or to improve upon the tool for similar image generation.

This research also recommends a mathematical similarity model to be used for checking the similarity between syntactically generated images, particularly BCSGs (see Chapter 6).

Finally, this research implemented the similar images generated using BCSGs into a prototype visual password scheme as distractors (see Chapter 7).

1.7 Road Map

The remaining part of this research consists of the following relevant chapters:

1. Chapter 2 provides an overview of works relating to formal grammars, picture grammars with emphasis on shape grammars, bag context grammars, shape grammar interpreters, models of image similarity, and visual password schemes.
2. Chapter 3 presents the theoretical definition of bag context shape grammars with examples.
3. Chapter 4 demonstrates the power of Bag Context Shape Grammars. Here different BCSG methods that generate structurally similar images are considered.
4. Chapter 5 demonstrates the theories presented in BCSGs in a prototype software called Bag Context Shape Grammar Interpreter which was used to generate similar images.
5. Chapter 6 checks the similarity of these images generated by BCSGs using the SpCD.
6. Chapter 7 implemented the similar images generated using BCSGs in a prototype visual password scheme.
7. Finally, Chapter 8 is the conclusion which summarises what was discussed in all other chapters.

The research presents the review of other work in the area of formal grammars, picture grammars with emphasis on shape grammars, bag context grammars, shape grammar interpreters, models of image similarity, and visual password schemes in Chapter 2.

Chapter 2

Review of Related Work

2.1 Introduction

A general overview of formal grammars with regards to image generation was presented in the last chapter. Hence, we look into formal grammars, particularly context free grammars and how they work. We also look at picture grammars, a subclass of context free grammars with regards to similar image generation, their methods, implementation and how they work. We look into the use and workability of bag context in grammars. Then, different mathematical models for the measure of image similarity were also considered as a way to determine which is best to check the similarity of images and why. Then, existing works in the area of visual password schemes were also considered. We conclude this chapter with a summary of all the literature.

2.2 Formal Grammars

2.2.1 Introduction

Formal grammars were first introduced by Post [115] and later improved by Thue and other researchers [75]. However, full use of formal grammars and languages did not start until the mid 1950s [22]. The general representation of a language is based on the formal notion of grammars. A formal grammar (grammar for short), consists of a finite set of rules, operating over a vocabulary of terminal and nonterminal symbols, and include a unique nonterminal called the initial or start symbol that starts the generative or derivation process [55, 75, 101]. Grammars are classified into four classes; type–0, type–1, type–2, and type–3, this is also known as Chomsky hierarchy (see Table 2.1) [22, 23, 24, 25].

TABLE 2.1: Chomsky hierarchy

Class	**Grammars**	**Automata**	**Language**
Type–0	Unrestricted	Turing machine	Recursively enumerable
Type–1	Context sensitive	Linear bounded automaton	Context sensitive
Type–2	Context free	Pushdown automaton	Context free
Type–3	Regular	Finite state automaton	Regular

In this research, interest is on type–2 class of grammars also known as context free grammars. Next, we present context free grammars in Section 2.2.2.

2.2.2 Context Free Grammars

The formal definition of context free grammars was adapted from Hopcorft et al [67]. First, we define some terms and notations.

2.2.2.1 Definition of Terms and Notations

- A symbol is a character or an abstract entity that has no meaning on its own.
- An alphabet is a finite set of symbols.
- A string is a finite ordered sequence of symbols taken from an alphabet.
- The symbol ϵ denotes a string with no symbol or a string with length zero. This is also called the empty string.
- If Σ is an alphabet, then the Kleene closure of Σ written as Σ^*, denotes all possible finite combinations of finite symbols taken from Σ, including the empty string ϵ.

2.2.2.2 Formal Definition of Context Free Grammars

Definition 2.1. A context free grammar (CFG) is a 4-tuple (V_M, V_T, R, S), where:

- V_M is a finite nonempty set of nonterminal symbols.
- V_T is a finite nonempty set of terminal symbols.
- R is a finite set of rules of the form $A \longrightarrow \beta$, where $A \in V_M$ and $\beta \in (V_T \cup V_M)$,
- S is a unique nonterminal, the start symbol [22, 24, 67, 73].

We now present the definitions of the sentential form of grammars, a direct derivation, a derivation and the language of grammar. The definition of the sentential forms of grammars is adapted from Hopcorft et al [67].

Definition 2.2. A sentential form of a grammar G is any string of terminals, V_T and nonterminals, V_M derivable from the start symbol, S. That is, a string over $(V_T \cup V_M)$ derivable from S.

Next, we define the direct derivation of grammars.

Definition 2.3. Let Υ_1 and Υ_2 be two sentential forms (see Definition 2.2). If $A \longrightarrow \beta$ is a rule in R, and $\Upsilon_1 = B_1AB_2$, $\Upsilon_2 = B_1\beta B_2$, where B_1 and B_2 are substrings. Then, Υ_1 directly derives Υ_2, denoted by $\Upsilon_1 \Rightarrow \Upsilon_2$.

Then, we define derivation.

Definition 2.4. Let Υ_1 and Υ_2 be two sentential forms of a grammar, G. Υ_1 derives Υ_2, denoted by $\Upsilon_1 \Rightarrow^* \Upsilon_2$, if there is a sequence of zero or more sentential forms, $\alpha_1, ..., \alpha_n$:

$$\Upsilon_1 = \alpha_1 \Rightarrow ... \Rightarrow \alpha_n = \Upsilon_2 \text{ [67, 75].}$$

The sequence $\Upsilon_1 = \alpha_1 \Rightarrow \alpha_2 ... \Rightarrow \alpha_n = \Upsilon_2$ is called a derivation of Υ_2 from Υ_1.

Next, is the language of a grammar.

Definition 2.5. The language $\mathcal{L}(G)$ of a grammar G is the set of terminal strings $\omega \in V_T^*$ derivable from the start symbol S. Formally, represented thus,

$$\mathcal{L}(G) = \{w \mid w \in V_T^* \colon S \Rightarrow^* w\} \text{ [67, 73, 75].}$$

Example 2.2.1 is used to explain how context free grammars work.

Example 2.2.1. Let $G_1 = (\{S\}, \{\text{a, b}\}, R, S)$ where R is the set in Figure 2.1.

The grammar G_1 generates all possible strings over $\{a, b\}$ derivable from the symbol S, denoted by $\mathcal{L}(G_1) = \{w \mid w \in \{a, b\}^* \colon S \Rightarrow^* w\}$. One of the strings in $\mathcal{L}(G_1)$ is *aabbbaa*. A derivation of the string, *aabbbaa*, is shown Figure 2.2. In Figure 2.2, Rules 2.1–2.4 above the arrows show the particular rule that was applied.

From the derivation in Figure 2.2, S directly derives aSa written as $S \Rightarrow aSa$ while S derives *aabbbaa* written as $S \Rightarrow^* aabbbaa$.

Classically, context free grammars (CFGs) generate string or word languages. However, research has shown that the approach of context free grammars can be used in generating image languages. The next section discusses some picture grammars in this class.

$$R = \left\{ \right.$$

$$\begin{aligned} S &\longrightarrow aSa && \text{(Rule 2.1)} \\ S &\longrightarrow bSb && \text{(Rule 2.2)} \\ S &\longrightarrow a && \text{(Rule 2.3)} \\ S &\longrightarrow b && \text{(Rule 2.4)} \end{aligned}$$

$$\left. \right\}$$

FIGURE 2.1: Set of rules for G_1.

$$S \overset{2.1}{\Longrightarrow} a\underline{S}a \overset{2.1}{\Longrightarrow} aa\underline{S}aa \overset{2.2}{\Longrightarrow} aab\underline{S}baa \overset{2.4}{\Longrightarrow} aabbbaa.$$

FIGURE 2.2: The derivation of a string in Example 2.2.1.

2.3 Picture Grammars

2.3.1 Introduction

In the last section, CFGs were discussed. A CFG is one where no context is used. That is, at any step during derivation, any rule can be applied when the symbol on the left hand side of the rule appears in the sentential form of the derivation at that step. CFGs generate string or word languages. However, the approach of CFGs can be applied to image generating grammars by viewing the generated strings as expressions that generate images. That is, by associating with it an algebra whose domain is a set of images.

Hence, many image generating grammars have been developed based on the formal notations of grammars and the approach of CFGs and more are still being developed. In this section, we will discuss the following picture grammars: Random Context Picture Grammars [41], Tree Based Picture Grammars [33], Iterated Function Systems (IFS) [15], L-systems [119], Collage Grammars [32], Chain Code Picture Grammars [125], Array Grammars [171], Puzzle Grammars [156], Shape Grammars [143] and most recently, Bag Context Picture Grammars [43].

2.3.2 Random Context Picture Grammars

Random Context Picture Grammars (RCPGs) is a type of grammar that has the ability to combine most of the simplicity of context free grammars with most of the strength

of context sensitive grammars [41]. The essence of RCPGs is that they are more powerful than context free grammars and less strenuous to implement than context sensitive grammars [41]. RCPGs were developed from the knowledge of random context grammars. The random context grammar (RCGs) or random context string grammar was first introduced in 1972 by Van Der Walt [164].

In RCPGs, a derivation starts with an initial shape. At any step during the derivation, the developing picture consists of the initial shape divided into non-overlapping shapes, each containing a variable or terminal [41]. A variable is rewritten according to a context free production of the underlying grammar. This means either dividing the shape containing it into non-overlapping shapes or replacing it by a variable or terminal. A production may depend on context randomly distributed in the developing picture [41].

Context is classified as either permitting (enables the application of a rule) or forbidding (inhibits the application of a rule). There are three best known subclasses of RCPGs, namely Random Permitting Context Picture Grammars (RPCPG)—when a grammar uses permitting context only, Random Forbidding Context Picture Grammars(RFCPG)—when a grammar uses forbidding context only, and Context Free Picture Grammars (CFPG)—when a grammar does not use permitting or forbidding context [41, 42, 44, 45, 46].

Next, the definitions of random context picture grammars, pictorial form, picture, subpicture, gallery and examples are adapted from [41].

Definition 2.6. An RCPG is a 4–tuple (V_M, V_T, R, (S, σ)) where

- V represents a finite set of labels,
- V_M represents a set of variables, $V_M \subset V$,
- V_T represents a set of terminals, $V_T \subset V$,
- R represents a finite set of rules of the form $A \longrightarrow [x_{11}, x_{12}, \ldots, x_{mm}]$ ($\mathcal{P}$; $\mathcal{F}$) (see Figure 2.3), $m = 1, 2, 3, \ldots$, $A \in V_M$, $x_{11}, x_{12}, \ldots, x_{mm} \in V$ and $\mathcal{P}$, $\mathcal{F} \subseteq V_M$,
- A labelled square is a pair (A,σ), where A is a symbol (variable or terminal) from a set of labels, V, and σ is a coordinate in a plane.
- (S, σ) represents the initial labelled square with $S \in V_M$ and σ is a square with coordinates (0,0)...(1,1), where (0,0) and (1,1) are the lower left and upper right corners respectively.

If every rule in a grammar G has $\mathcal{P} = \mathcal{F} = \emptyset$, then we call G a context free picture grammar (CFPG), if only $\mathcal{F} = \emptyset$, then we call G a random permitting context picture

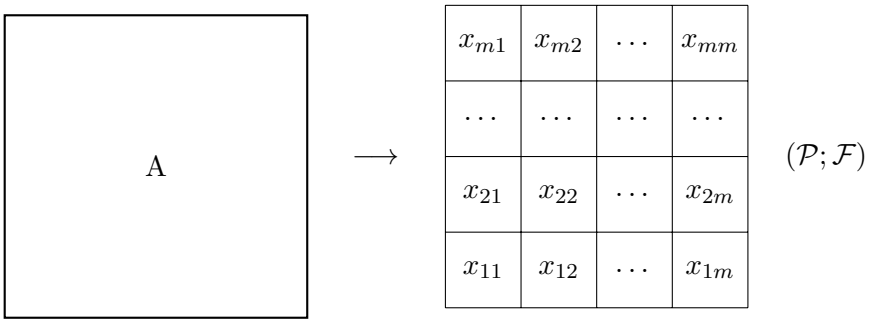


FIGURE 2.3: The rule of a random context picture grammar [41].

grammar (RPCPG), and if only $\mathcal{P} = \emptyset$, then we call G a random forbidding context picture grammar (RFCPG) [41].

Definition 2.7. A pictorial form is defined as any finite set of non overlapping labelled squares in the plane. If Π is a pictorial form, then we use $l(\Pi)$ to represent the set of labels used in Π. The size of a pictorial form Π is the number of squares contained in it, that is, $\mid \Pi \mid$ [41].

For an RCPG G and pictorial forms Π and Γ, we write $\Pi \Longrightarrow_G \Gamma$ if there is a rule $A \longrightarrow [x_{11}, x_{12}, \ldots, x_{mm}]$ $(\mathcal{P};\ \mathcal{F})$ in G, where Π contains a labelled square $(A,\ \alpha)$, $l(\Pi \setminus \{A, \alpha\}) \supseteq \mathcal{P}$ and $l(\Pi \setminus \{A, \alpha\}) \cap \mathcal{F} = \emptyset$, and $\Gamma = l(\Pi \setminus \{A, \alpha\}) \cup \{\ (x_1, x_{11}), (x_2, x_{12}), \ldots, (x_{mm}, \alpha_{mm})\}$. Hence, $\Longrightarrow_G^*$ denotes the reflexive transitive closure of $\Longrightarrow_G$ [41].

Next, we present picture, subpicture and gallery.

Definition 2.8. A picture is a pictorial form Π with $l(\Pi) \subseteq V_T$.

Definition 2.9. A subpicture Ω of a picture Π is any subset of Π that fills a square, that is, the union of all the squares in Ω is a square.

Definition 2.10. The gallery $\mathcal{G}(G)$ generated by a grammar $G = (V_N,\ V_T,\ R,\ (S,\ \sigma))$ is the set of pictures Π such that $\{(S,\ \sigma)\} \Longrightarrow_G^* \Pi$. We call the gallery generated by an RCPG, a Random Context Gallery $(RCPL)$. If $\mathcal{P} = \emptyset$ for every rule in G, then, the gallery generated by G is called a random permitting context gallery $(RPCPL)$ and if $\mathcal{F} = \emptyset$ for every rule in G, then, the gallery generated by G is called a random forbidding context gallery $(RFCPL)$.

Next, we give example of a gallery that can be generated by an $RCPG$ (see Example 2.3.1).

Example 2.3.1. Consider the iteration sequence of the Sierpiński carpet (see Figure 2.6). Let $G_{\text{carpet}} = (\{S,\ T,\ U,\ F\},\ \{w,\ b\},\ R,\ (S,\ \sigma))$, where R is the set in Figure 2.4.

$$R = \Bigg\{$$

$$\begin{aligned}
S &\longrightarrow [T,T,T,T,w,T,T,T,T,](\emptyset;\{U\}) && \text{(Rule 2.5)}\\
T &\longrightarrow U(\emptyset;\{S,F\}) \mid && \text{(Rule 2.6)}\\
&\quad\ F(\emptyset;\{S,U,F\}) \mid && \text{(Rule 2.7)}\\
&\quad\ b(\{F\};\emptyset) && \text{(Rule 2.8)}\\
U &\longrightarrow S(\emptyset;\{T\}) && \text{(Rule 2.9)}\\
F &\longrightarrow b(\emptyset;\{T\}) && \text{(Rule 2.10)}
\end{aligned}$$

$$\Bigg\}$$

FIGURE 2.4: The set of rules for the grammar G_{carpet} in Example 2.3.1.

The terminals *'w'* and *'b'* will be represented by *'light'* and *'dark'* colours respectively. The derivation of image in Figure 2.6(A) is shown in Figure 2.5.

The application of Rule 2.5 divides the initial square S into nine equal squares which are also recursively repeated after the application of Rules 2.6 and 2.9. This repetition can only be terminated when Rule 2.10 is applied after Rules 2.6 and 2.7 or Rule 2.8 is applied after Rule 2.5.

The grammar in Example 2.3.1 is used to produce the picture in Figure 2.6. The grammar can generate images with parts. Observe that the image parts in the fifth iteration of the Sierpiński carpet are no longer visible (see Figure 2.6 C), but the iteration can go on and on which makes it more difficult to see the parts. This is because the rules of the grammar are not designed to control the parts of the images. Also, it is a refinement type of grammar. That is, during derivation, it refines the initial shape, which is different from the additive type of grammar we are concerned with.

Example 2.3.2. Consider the gallery of images with submimages. Each subimage is refined differently. Let RCPG $G_{\text{sub}} = (V_M,\ V_T,\ R,\ (S,\ \sigma))$ where $V_M = \{\text{S, A, B, C, F}\}$, $V_T = \{w,\ b\}$ and R is the set of rules in Figure 2.7.

Some of the picture generated by the grammar G_{sub} are show in Figure 2.8. The terminals *'w'* and *'b'* are represented by *'light'* and *'dark'* colours respectively.

The random context controls the application of the rules. For example, Rule 2.11, can only be applied if there is no C present in the pictorial form, while Rule 2.15 can only

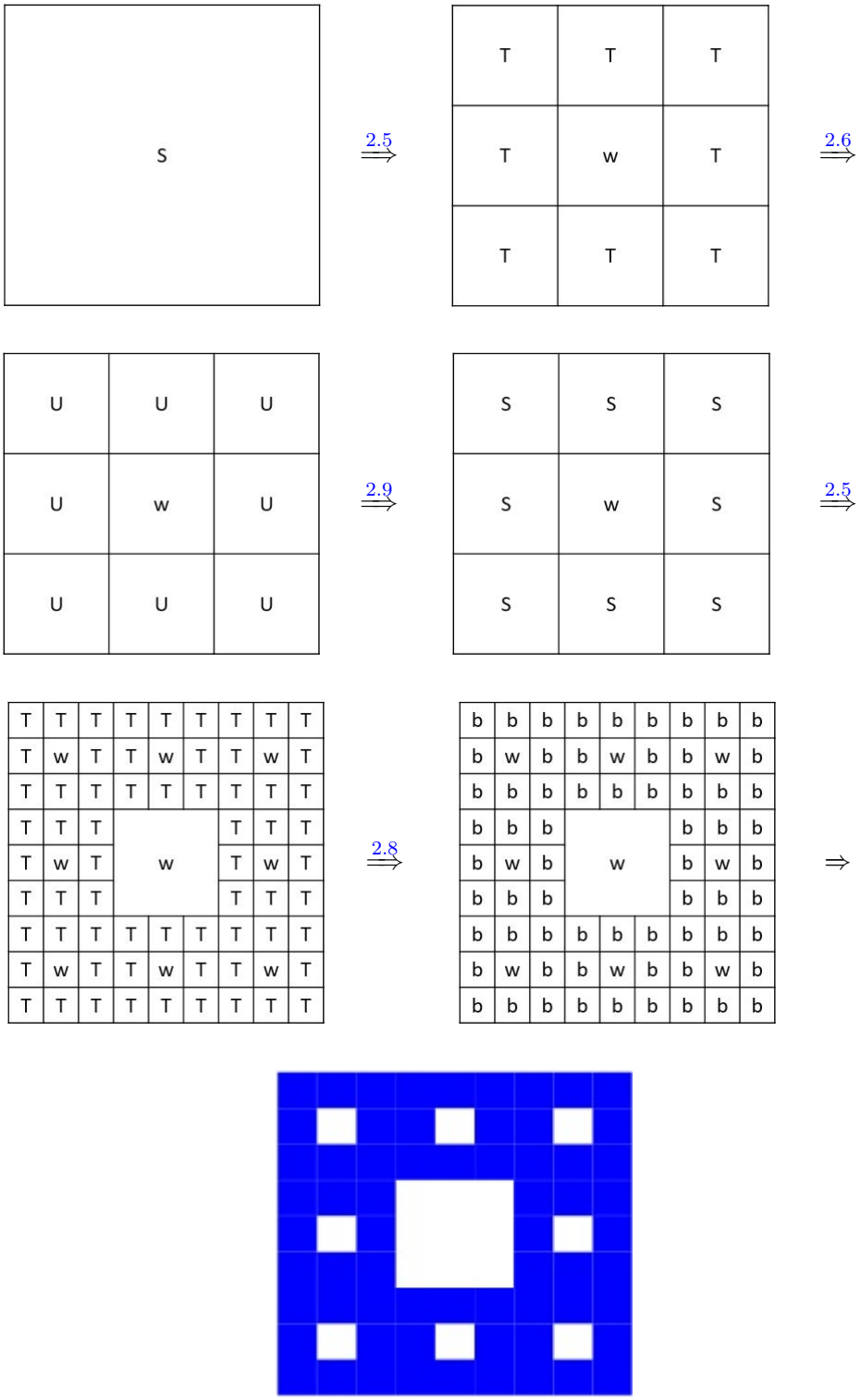
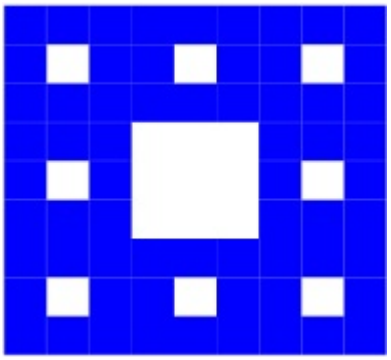
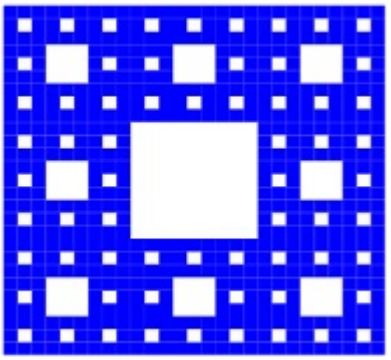


FIGURE 2.5: The derivation of image in Figure 2.6(A).

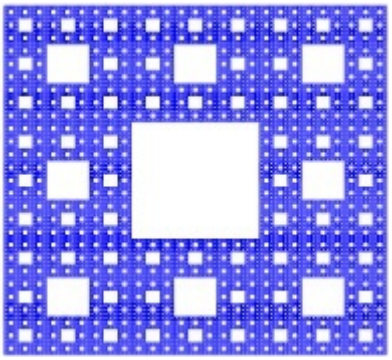
be applied if there is F and no S in the pictorial form. This control or regulation helps this the grammar to generate similar images with submimages in a refinement approach which is different from our type of bag context shape grammars that builds upon the axiom during derivation.



(A) Second refinement of the Sierpiński carpet.



(B) Third refinement of the Sierpiński carpet.

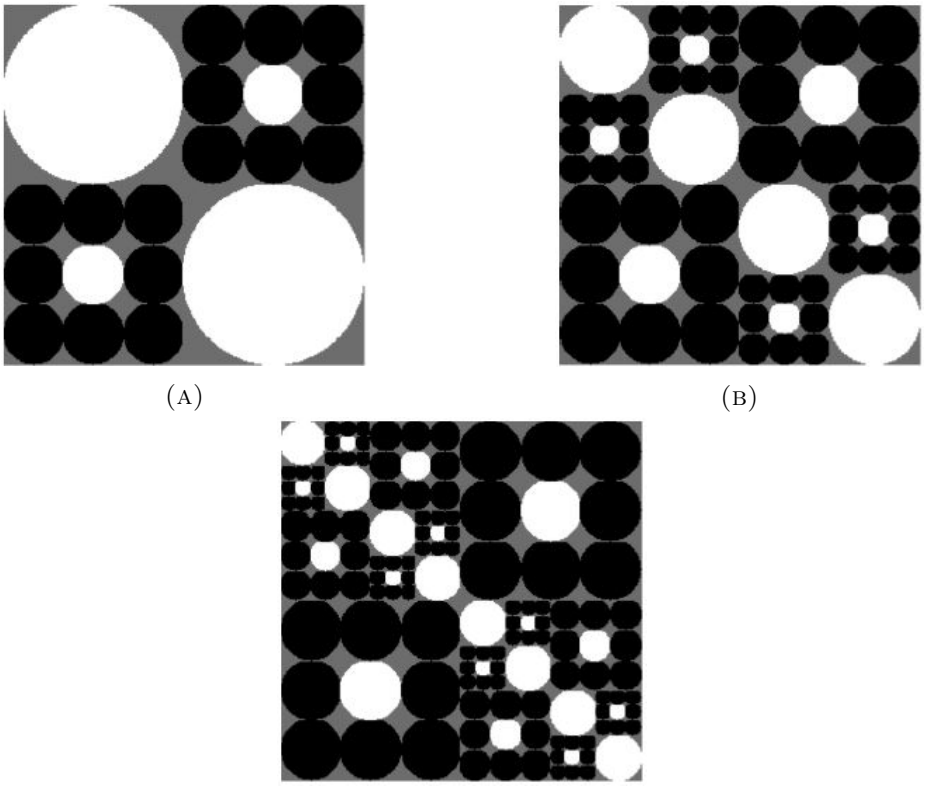


(C) Fifth refinement of the Sierpiński carpet.

FIGURE 2.6: The second, third and fifth iteration sequence of the Sierpiński carpet in Example 2.3.1.

$$
R = \left\{
\begin{array}{lcll}
S & \longrightarrow & [B, A, A, B](\emptyset; \{C\}) & \text{(Rule 2.11)} \\
B & \longrightarrow & [b, b, b, b, w, b, b, b, b](\emptyset; \{S, F\}) \mid & \text{(Rule 2.12)} \\
A & \longrightarrow & C(\emptyset; \{S, F\}) \mid & \text{(Rule 2.13)} \\
 & & F(\emptyset; \{S, C, F\}) \mid & \text{(Rule 2.14)} \\
 & & w(\{F\}; \{S\}) & \text{(Rule 2.15)} \\
C & \longrightarrow & S(\emptyset; \{A\}) & \text{(Rule 2.16)} \\
F & \longrightarrow & w(\emptyset; \{A, S\}) & \text{(Rule 2.17)}
\end{array}
\right\}
$$

FIGURE 2.7: The set of rules for the grammar G_{sub} in Example 2.3.2.



(A) (B)

(C)

FIGURE 2.8: Some pictures in $\mathcal{G}(G_{\text{sub}})$ of Example 2.3.2.

Next we discuss tree based picture grammars in Section 2.3.3.

2.3.3 Tree Based Picture Grammars

A tree grammar is can be regarded as a grammatical device, G that produces a tree language, $L(G)$ over an output signature Σ'. A ranked alphabet is a pair $(F,Arity)$ of an ordinary alphabet. Here F is a finite set and $Arity$ is a mapping from F to $\mathbb{N}$ of the form $F \rightarrow \mathbb{N}$. f^k represents a ranked alphabet symbol where f is the symbol and k is the rank. A signature is a set Σ of ranked symbols f^k (where $k \geq 0$ is the rank).

A tree transducer, τ computes a relation $\tau \subseteq T_\Sigma \times T_{\Sigma'}$, where T is a set of trees, T_Σ is a set of trees over Σ, $T_{\Sigma'}$ is a set of trees over Σ', Σ is the input signature, and Σ' is the output signature. The language generated by a tree transducer is denoted by $L(\tau) = \tau(T_\Sigma)$ [33]. A tree generator is also a tree grammar or a tree transducer.

There are different classes of tree generators such as, context free tree grammars, regular tree grammars, random context tree grammars, top down tree transducers (a tree transducer that generates an output in steps as it reads from the root down to the leaves of the tree) and bottom top tree transducers (a tree transducer that reads an

input tree by starting at the leaves and working upward toward the root of the tree) [14, 33, 34, 36, 161].

In this section, we focus on regular tree grammars. We present the definition of regular tree grammars in Definition 2.11.

Definition 2.11. A regular tree grammar is a 4–tuple $G = (N, \Sigma, R, S)$ where

- N represents a finite signature of nonterminals of rank 0
- Σ represents a finite signature of terminals, $N \cap \Sigma = \emptyset$
- $R \subseteq N \times T_{\Sigma \cup N}$ represents a finite set of rules of the form $A \to t\ [A_1, \ldots, A_k]$ for all $t^{(k)} \in \Sigma$ and $A_1, \ldots, A_k \in N$. For clarity, let R be represented as $A \to t$ where $A \in N$ and $t \in T_{\Sigma \cup N}$
- S represents an initial nonterminal, $S \in N$.

Let $t, t' \in T_{\Sigma \cup N}$ be derivation step from t to t' written as $t \Longrightarrow_G t'$ (or simply $t \Longrightarrow t'$). We say t directly produces t', if there is a rule $A \to s$ in R such that, $t = s[A]$ and $t' = s[r]$ [36, 126].

The language generated by G is $L(G) = \{t \in T_\Sigma \mid S \overset{*}{\Rightarrow}_G t\ \}$, where the relation $\overset{*}{\Rightarrow}_G$ is the transitive and reflexive closure of $\Longrightarrow_G$.

A tree generator can be turned into an image generating device by viewing the generated trees as expressions that generate images. That is, by associating with it an algebra whose domain is a set of images. Let $\mathcal{P}$ be such a Σ–algebra and let g be a tree generator whose output signature is a subset of Σ. Then the pair $(g, \mathcal{P})$ is called a tree generator based picture generator that can generate a picture language.

Example 2.3.3. Consider the iteration sequence of the Sierpiński triangle (see Figure 2.10).

Let $G_{\text{tree}} = (\{S,\ f^{(2)},\ q^{(2)},\ T,\ Q\},\ \{w, b\},\ R,\ S)$, where R is the set in Figure 2.9.

The terminals *'w'* and *'b'* represent *'light'* and *'dark'* colours as usual. The illustration of Example 2.3.3 is shown in Figure 2.10 [35, 41, 108].

Observe that Rule 2.19 will continue to repeat after rule 2.21. This could cause the parts of the picture to continue to refine. Hence tree grammars can generate similar images. The major difference between tree base picture grammars and bag context shape grammars is the derivation process. The former uses a refinement approach during derivation, while the latter uses an additive approach during derivation.

Next, we discuss the L–systems in Section 2.3.4.

$$
R = \left\{
\begin{array}{rcll}
S & \longrightarrow & f[f_1, f_2] & \text{(Rule 2.18)} \\
f[f_1, f_2] & \longrightarrow & [T, Q, T, T] \mid & \text{(Rule 2.19)} \\
 & & b & \text{(Rule 2.20)} \\
T & \longrightarrow & f[f_1, f_2] & \text{(Rule 2.21)} \\
Q & \longrightarrow & [q_1, q_2]q & \text{(Rule 2.22)} \\
.[q_1, q_2]q & \longrightarrow & w & \text{(Rule 2.23)}
\end{array}
\right\}
$$

FIGURE 2.9: The set of rules for G_{tree} in Example 2.3.3

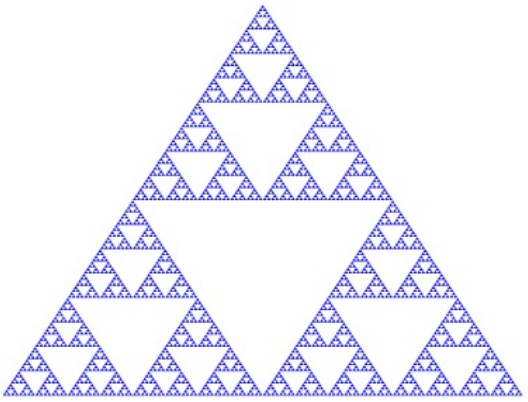


FIGURE 2.10: Seventh iteration of Sierpiński triangle.

2.3.4 L–Systems

Lindenmayer systems (L-systems) use a set of rewriting rules to successively replace complex objects with simple objects. In L-systems, rules are applied in parallel and simultaneously replace all letters in a given word. L-systems are widely used for the generation of fractals, realistic modelling of plants, etc [41, 98].

There are three major different classes of L-systems, such as DOL-systems, stochastic L-systems, and ETOL-systems. DOL-systems are a deterministic and context free class of L-systems [118], that is, they generate identical structures even with repetitive usage. This gave rise to the development of the stochastic L-systems. The stochastic L–systems generate structures with a variation using the same rules. To achieve this fit, it applies probability in every rule [118]. The Extended Table 0 (zero) L–systems (ETOL–system), generates structures using rules which do not depend on any context [34]. ETOL–systems use a table to describe structures. The use of tables in ETOL-systems gives them the ability to describe some natural and attractive structures otherwise they will be difficult to define [34].

The definition of L-systems was adapted from Prusinkiewicz [118]. Let V denote an alphabet, V^* is the set of all words over V and V^+ is the set of all non-empty words over V.

Definition 2.12. A 0L-system is an ordered triplet G=(V, w, R) consisting of:

- an alphabet V,
- an axiom $w \in V^+$, and
- a finite set of rules R.

A rule in R has the form $a \longrightarrow X$ for a pair (a,X). We call a the predecessor and X the successor of this rule. For any $a \in V$, there is at least one word $X \in V^*$ such that $a \longrightarrow X$. A 0L-system is deterministic if, and only if, for each $a \in V$ there is exactly one $X \in V^*$ such that $a \longrightarrow X$.

L-Systems also can generate images with variations and can generate similar images. The major drawback of L–systems is that all part of an image are refined at the same time during derivation.

In Example 2.3.4, the Sierpiński triangle is drawn using the L–systems.

Example 2.3.4. Let $G_{\text{cap}} = (V, \omega, R)$, where $V = \{F, G, +, -\}$, $\omega = F - G - G$ and R contains the set of rules in Figure 2.11.

$$R = \left\{ \begin{array}{lll} F & \longrightarrow & F - G + F + G - F \quad \text{(Rule 2.24)} \\ G & \longrightarrow & GG \quad \text{(Rule 2.25)} \end{array} \right\}$$

FIGURE 2.11: The set of rules for G_{cap} in Example 2.3.4.

The variables F and G represent "forward draw", the symbol "+" represents turn left angle 90°, the symbol "–" also represent turn right angle 90°. The derivation starts with the variable F, then, any of rules 2.24 or 2.25 can be recursively applied. One of the images generated with this approach is shown in Figure 2.12.

A major drawback of the L–system is that all parts of the image is refined at the same time during derivation. The difference between L-systems and bag context shape grammars is that L-systems generate images using a refinement method, while bag context shape grammars builds upon the axiom during derivation.

Next, we discuss the iterated function systems in Section 2.3.5.

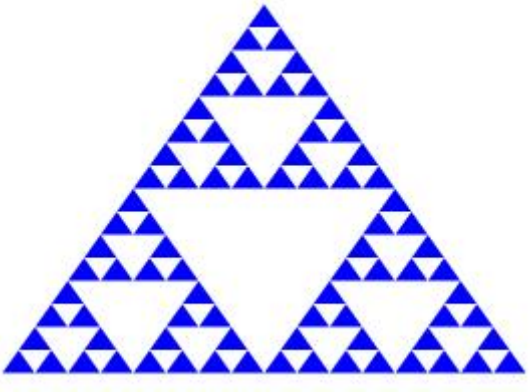


FIGURE 2.12: The fourth iteration of Sierpiński triangle.

2.3.5 Iterated Function Systems

Iterated Function Systems (IFSs) were developed by Barnsley and co-workers [15] for constructing fractals from a finite set of contractive maps. That is, they bring points closer together and make shapes smaller. A contractive map is a mapping that shrinks distances, they only exist on metric spaces. Barnsley and co-workers used this method to generate some real life images. One possible explanation for the success of this iterative approach is that the human eye discerns less information in such scenes than one might expect. IFSs generate images with a high degree of self-similarity, as a result, they generate a limited range of images.

Moreover, the Mutually Recursive Function Systems (MRFSs), were developed as a generalised type of IFSs. This development was to discover more ranges of fractal–like images that do not possess much level of self–similarity [33, 41, 93].

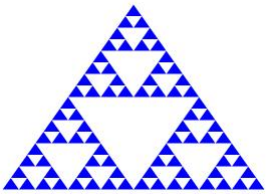
Next, we present the definition of IFS in Definition 2.13. The definition of IFS is adapted from Barnsley et al [15].

Definition 2.13. An iterated function system is a complete metric space X together with a finite set of contractive maps f_i, represented as, $\{f_i : X \longrightarrow X \mid i = 1, \ldots, N\}$, $N \in \mathbb{N}$.

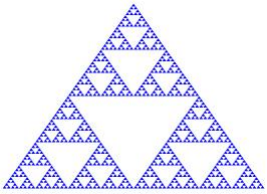
An example of some similar images generated using the approach of the IFS is the Sierpiński triangles shown in Figure 2.13.

The main drawback of an IFS is similar to that of an L-system, which is, all parts are refined at the application of the map, it is not possible to choose only a part of the maps available. This also shows the different between IFS and bag context shape grammars that builds upon the axiom during derivation. For example, it becomes difficult to

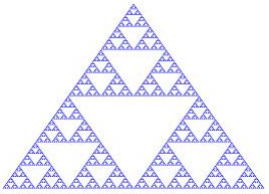
(A) The second iteration of Sierpiński Triangle.



(B) The fourth iteration of Sierpiński Triangle.



(C) The sixth iteration of Sierpiński Triangle.



(D) The seventh iteration of Sierpiński Triangle.

FIGURE 2.13: Some images generated using IFS approach.

tell the difference between some of the images at some point during a derivation, see Figures 2.13(C) and (D).

Next, we discuss collage grammars in Section 2.3.6.

2.3.6 Collage Grammars

Collage grammars are context free image generating grammars, which use the approach of hyperedge replacement for the generation of images [32, 41, 57, 76]. A collage is an image which is made up of different parts, where each part represents a set of geometric points in a Euclidean space [50, 56]. The generation of collages is based on the replacement of variables. The replacement of variables involves the building of functions that is applied to parts [41, 49].

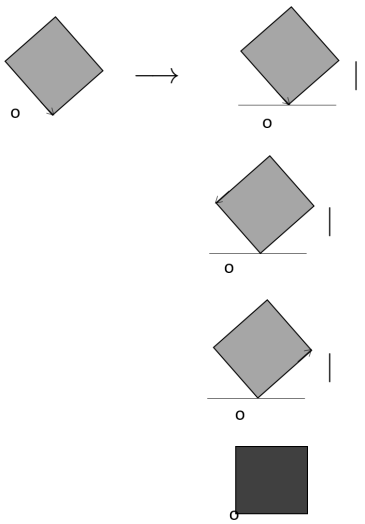
The formal definition of collage grammar is presented in Definition 2.14. The definition of collage grammar is adapted from Drewes et al [49].

Definition 2.14. A context free collage grammar is a regular tree grammar $G = (V_N, \Sigma, R, S)$, where V_N is a finite set of variables, Σ is a collage signature, R is a set of rules, and S is the starting symbol. The language generated by a context free collage grammar, G is a context free language.

Next, we illustrate this notion with Example 2.3.5, as adapted from Drewes [34].

Example 2.3.5. Consider the grammar G_{collage} which is linear and uses only a single variable. The set of rules are represented in Figure 2.14. Some images generated by G_{collage} are given in Figure 2.15.

$P = \Big\{$



(Rule 2.26)

(Rule 2.27)

(Rule 2.28)

(Rule 2.29)

$\Big\}$

FIGURE 2.14: The rules of G_{collage}

.

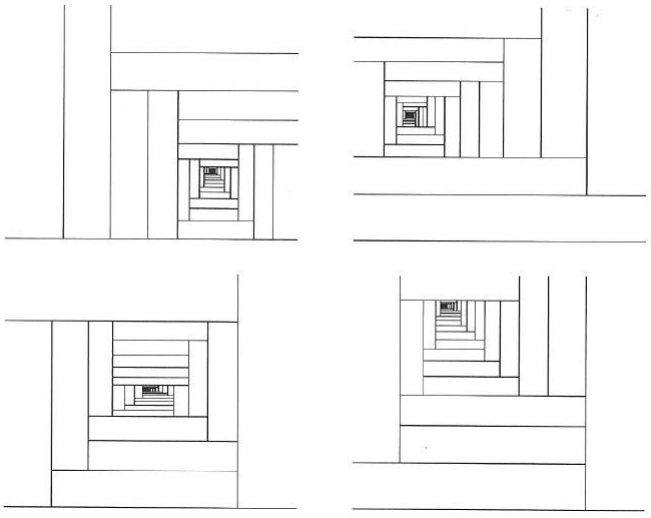


FIGURE 2.15: Some images generated by G_{collage}.

The drawback of Collage grammars is that they are context free, as such, they cannot generate images that consist of collages that are not straigh forward in growth as regards to the number of parts. Meanwhile, bag context shape grammars are also context free but the application of a rule is controlled. This becomes the major difference between the two models. Next, we discuss chain code picture grammars in Section 2.3.7.

2.3.7 Chain Code Picture Grammars

Chain code picture grammars concentrate on the generation of images that could be seen as line drawings. Here, images are formed with connected horizontal and vertical lines between points in the plane. Examples of such images include but are not limited to, maps, charts, characters, arbitrary contour images, etc [41, 125]. They operate with a set of unit lines from the integer Cartesian grid and is described by a word over the alphabet $\{u, r, l, d\}$, where u, r, l and d represent an upward, right, left and downward drawing movements of one unit length from the current point respectively [41, 125].

While chain code picture grammars are context free, bag context shape grammars are also context free but the application of a rule is controlled by a special vector of integer. Therefore, the two models generate images differently. A major drawback of chain code picture grammars is that it is difficult to generate closed curves like rectangles and squares, etc. because they are context free.

Next, we discuss the array grammars in Section 2.3.8.

2.3.8 Array Grammars

Array grammars form pictures with connected cells of two dimensional arrays in the plane. They are pattern generating models and can be extended to higher dimensions. They are also good for the development of two-dimensional cluster analysis. Array grammars can also be used for cellular automata, amongst other uses [41, 151, 171].

There are two major types of array grammars, namely, isometric and non–isometric array grammars. In isometric array grammers, the shape of the rewritten sub–array is preserved after rule application [171]. This is because the left and the right sides of the rules are geometrically identical. Further, the application of a rule can change the dimension of the rewritten sub–array in non–isometric array grammars [41, 151, 171].

Next, we present the formal definition of array grammar in Definition 2.15. All definitions and example in this part are adapted from [41, 151, 157].

Definition 2.15. An Array Grammar (AG) $G = (V_N, V_T, R, S, \#)$, has a finite alphabet V of labels or symbols, consisting of V_N and V_T of finite, nonempty sets of variables and terminals, respectively, R is a finite set of rules, $S \in V_N$ is the start symbol and $\# \notin V_N \cup V_T$ denotes the blank symbol. The rules are of the form $\alpha \longrightarrow \beta$, where α and β are arrays over $V \cup \{\#\}$ that satisfy the following conditions:

- the shapes of left and right hand sides (α and β) of the rule have geometrically identical shapes,

- the left hand side, α, of the rule contains at least one variable,
- terminals in α are not rewritten by the rule, and
- the application of the rule to a connected array gives another connected array.

A pictorial form over V is a two-dimensional finite connected array of symbols in V. The set of all nonempty pictorial forms over V is denoted by V^{2+}.

An Array Grammar is called a

- Monotonic Array Grammar (MAG) if no non-# in α is rewritten into a # by the application of a rule $\alpha \longrightarrow \beta$.
- Context-Free Array Grammar (CFAG) if the lefthand side of each rule consists of exactly one nonterminal and possibly one or more #'s.
- Regular Array Grammar (RAG) if each rule is of one of the forms shown in Figure 2.16, where $\{A, B\} \subseteq V_N$ and $\{a\} \in V_T$.

$R = \Bigg\{$

# A	$\longrightarrow$	B a		(Rule 2.30)
A #	$\longrightarrow$	a B		(Rule 2.31)
A	$\longrightarrow$	a		(Rule 2.32)
# / A	$\longrightarrow$	B / a		(Rule 2.33)
A / #	$\longrightarrow$	a / B		(Rule 2.34)

$\Bigg\}$

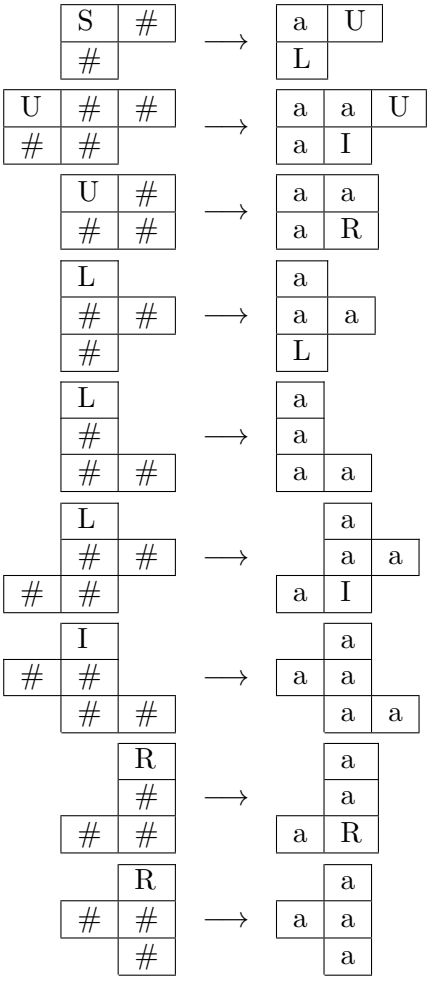
FIGURE 2.16: Rules of a regular array grammar.

Next, we illustrate array grammars in Example 2.3.6

Example 2.3.6. Let $h(x)$ and $w(x)$ denote the height and width of a rectangle x, respectively. Let $\mathcal{G}_\square = \{x \mid x \in \mathcal{G}_\square \text{ and } h(x) = 2i + 4,\ w(x) = 2j + 3,\ i,\ j = 0,\ 1,\ 2,\ \ldots\}$.

The gallery, $\mathcal{G}_\square$, is generated with the CFAG $\mathcal{G}_\square = (\{S, U, L, I, R\},\ \{a\},\ R,\ S\ \#)$, where R is the set of rules in Figure 2.17. One of the images generated with $\mathcal{G}_\square$ is shown in Figure 2.18.

$R = \Bigg\{$



$\Bigg\}$

FIGURE 2.17: Rules of $\mathcal{G}_{\square}$ [41].

To generate figures such as rectangles or squares using array grammars, there must be some kind of context, for example, they must be able to check whether each of the four edges is constructed in a straight line or have the same length in any two close edges. Achieving this may require the rules of RAGs or CFAGs to be rewritten with # as a form of context. For example, in the generation of solid rectangles in Figure 2.18, the CFAG uses local shapes of the host array as a guide as to which rule should be applied at a particular point. This shape–sensing or #–sensing ability comes from the restriction that the two sides of the rule must be geometrically identical [41].

The major drawback of this grammar is that rewriting rules of RAGs or CFAGs cannot sense the context of non-#'s in a pictorial form. Therefore, they cannot generate geometric figures that require non-#-context, example, the gallery of all hollow rectangles

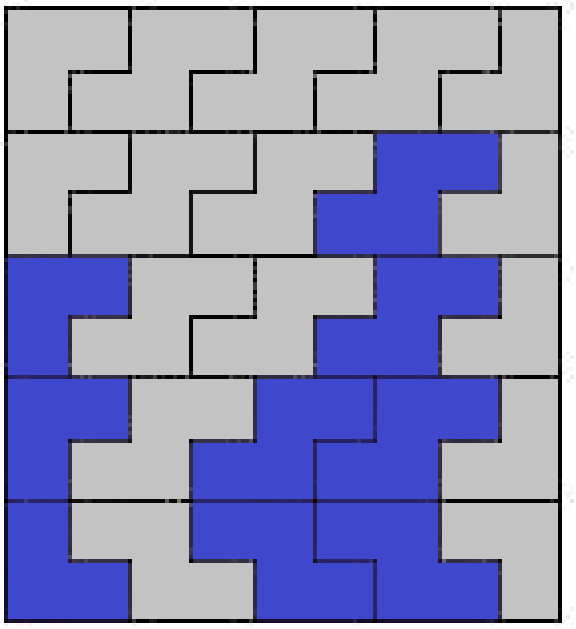


FIGURE 2.18: A member of $\mathcal{G}_{\square}$ [41].

of thickness one built with the letter a [170].

Although there are developments in array grammar systems with permitting features to address some of their drawbacks. These developments include but are not limited to context free array grammar systems with permitting features [151] which can generate arrays of L with base and height equal in length, permitting features in P systems generating picture arrays [152] which generates picture language with arrays, array P systems with permitting features [153] which can generate picture languages consisting of picture arrays in membrane computing. A more recent development in array grammar systems is the concept of cooperating context free array grammar systems [151], that can generate all hollow square frames in the so-called terminal mode, using only two array grammars. This research further demonstrated that the rules of some of these developments in array grammar systems can be rewritten as BCSGs rules in Chapter 7.

Next, we discuss Puzzle grammars in Section 2.3.9.

2.3.9 Puzzle Grammars

Puzzle grammars were derived from array grammars. In puzzle grammars, picture formation starts in a cell and gradually extends to other cells. The concepts, formal definition and examples of puzzle grammars are adapted from [41], [155] and [156].

Next, we give the definition of basic puzzle grammar in Definition 2.16.

Definition 2.16. A basic puzzle grammar (BPzG) $G = (V_N, V_T, R, S\ \#)$ has a finite alphabet V of labels, consisting of a disjoint, nonempty, finite sets V_N and V_T of variables and terminals, respectively, a finite set of rules R, a start symbol $S \in V_N$ and a special symbol $\# \notin V_N \cup V_T$ that denotes the blank symbol. A rule has one of the forms in Figure 2.19, where $\{A, B\} \in V_N$ and $\{a\} \in V_T$.

$R = \Big\{$

$A \longrightarrow$ [a] / B | (Rule 2.44)

[B] / a | (Rule 2.45)

[B] a | (Rule 2.46)

[a] B | (Rule 2.47)

[a] | (Rule 2.48)

a / [B] | (Rule 2.49)

B / [a] | (Rule 2.50)

B [a] | (Rule 2.51)

a [B] (Rule 2.52)

$\Big\}$

FIGURE 2.19: Rules of a basic puzzle grammar.

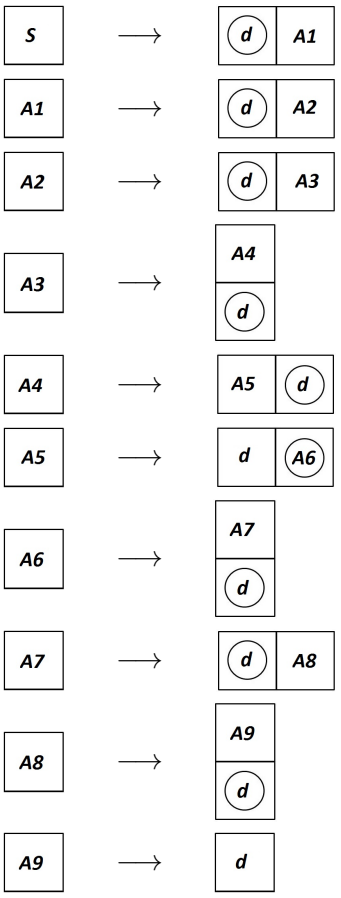
In puzzle grammars, a derivation begins with S written in a unit cell in the two–dimensional plane, with all other cells containing the blank symbol #. The derivation step, denoted by $\Longrightarrow_G^*$, begins with a nonterminal A in a cell, which is replaced by the right hand side of a rule with left hand side A. In this replacement, the square symbol of the right hand side of the rule used occupies the cell of the replaced symbol A, and the non–square symbol of the right hand side occupies the cell to the right, left, above or below the cell of the replaced symbol, depending on the type of rule that is applied. The replacement is only possible if the cell to be filled in by the non–square symbol contains a #.

A *picture* generated by G is a connected, finite array of elements of V_T derived in one or more steps from the start symbol; the set of such pictures–the *gallery*–is denoted by $\mathcal{G}(G)$. Other classes of puzzle grammars include *modified basic puzzle grammars* [155] and *context free puzzle grammars*.

Next, we illustrate puzzle grammars in Example 2.3.7, that was adapted from [155].

Example 2.3.7. Consider the basic puzzle grammar $G_1 = (\{A_i : 1 \leq i \leq 9\} \cup \{S\}, \{d\}, R, S)$ where R is the set in Figure 2.20.

$R = \Big\{$



$\Big\}$

FIGURE 2.20: The set of rules in G_1.

An image of $\mathcal{G}(G_1)$ is shown in Figure 2.21.

The major drawback of puzzle grammars is that, they do not generate galleries such as the set of all hollow rectangles. They need a large number of rules to generate relatively simple geometrical forms.

There are developments in the puzzle grammar system with permitting features to address some of its drawbacks. These developments include, but are not limited to, cooperating basic puzzle grammars systems [154], basic puzzle grammars with permitting features [71], etc.

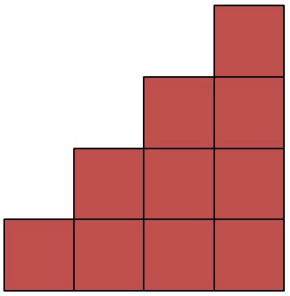


FIGURE 2.21: An image of $\mathcal{G}(G_1)$.

Next, we discuss shape grammars in Section 2.3.10.

2.3.10 Shape Grammars

Shape grammars were first introduced in 1971 by Stiny and Gips [147]. They presented a formalism (prescribed logical form) for the generation and arrangement of a class of geometric paintings. These paintings are the material representation of two dimensional shapes generated by shape grammars, having some algorithmic specifications in terms of recursive schemata as the basic formal component. Shapes are defined in the shape grammar formalism as labelled shapes and parametrised labelled shapes. Each such formalism defines the language of the grammar [143]. A shape grammar uses shapes and spatial rules to generate designs [147].

According to Stiny and Gips [147], a shape grammar consists of a set of nonterminal shapes, a set of terminal shapes, the starting shape and the set of rules. A rule consists of two shapes one on the left and another on the right side of the rule. During derivation, image designs are generated from the shape grammar by beginning with the starting shape and recursively applying the shape rules without restriction or control. The shape grammar rules are applied by identifying the part of the shape that is similar to the left side of the rule in terms of both nonterminal and terminal shapes and replaces it. The output of this application is the given shape with the right side of the rule substituted in the shape for an occurrence of the left side of the rule. When a terminal shape is added during the derivation it cannot be replaced, thus, the generation process is terminated when no rule in the grammar can be applied or when the shape is filled with the terminals only.

The language generated by a shape grammar is the set of shape designs generated by the grammar that do not contain any element of the nonterminal shapes [147, 172].

Next, the formal definition of shape grammar is presented in Definition 2.17. The definition of shape grammar is adapted from [147].

Definition 2.17. A Shape Grammar (SG) is a 4–tuple (V_M, V_T, R, S), where:

- V_M is a finite set of shapes called nonterminal shapes such that $V_T{}^* \cap V_M = \emptyset$; where $V_T{}^*$ represents all possible finite combinations of shapes taken from V_T,
- V_T is a finite set of symbols called terminal shapes,
- R is a finite set of ordered pairs (u,v) called spatial rules, such that, u is a shape on the left side of the rule and v is a shape on the right side of the rule,
- S is the start or initial shape.

The pair (u,v), represented as $u \rightarrow v$, provides the building block of the shape rule definition. A shape grammar rule consists of two labelled shapes one on the left and another on the right side of the rule. These shapes and the initial shape S (see Figure 2.22) are taken from the sets V_M and V_T. The initial shape S appears on the left side of at least one rule. The initial shape S and any shape that appears on the left side of a rule are not allowed to be empty, but shapes on the right side could be empty [147]. The language generated by this type of grammar is the set of shapes that do not contain any elements of V_M. The language of a shape grammar is a potentially infinite set of finite shapes [84, 147].

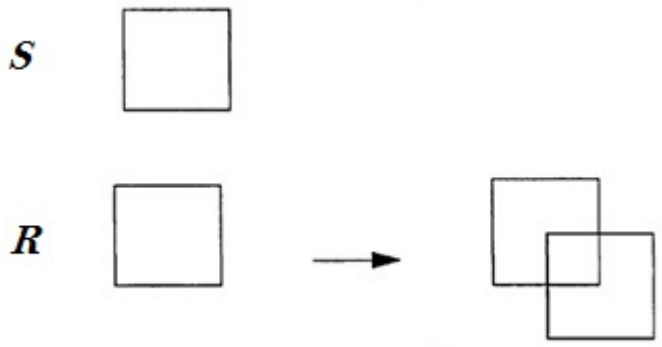


FIGURE 2.22: Shape grammar rule [159].

A shape grammar defined by filling open terms (specified points or coordinates) in a schema is called a parametric shape grammar (see Figure 2.24). A schema or rule $u \rightarrow v$ consists of parametrised labelled shapes u and v, where no labelled shapes in u are empty.

Next, we give some examples to describe shape grammars. Example 2.3.8 is adapted from [89].

Example 2.3.8. Consider a shape grammar $G_{\mathrm{SG}-1} = (V_M, V_T, R, S)$, where:

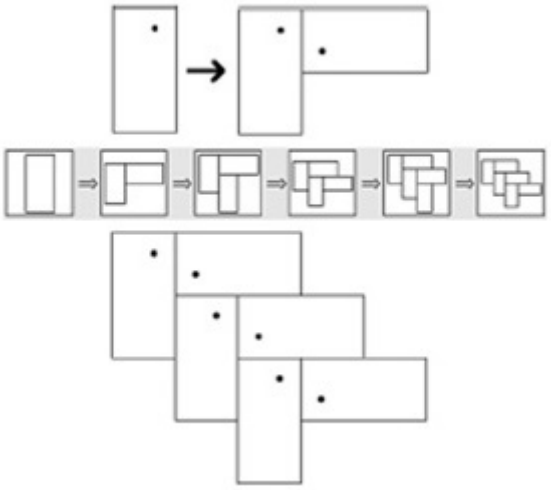


FIGURE 2.23: Labeled shape grammar rules and their derivation process [162].

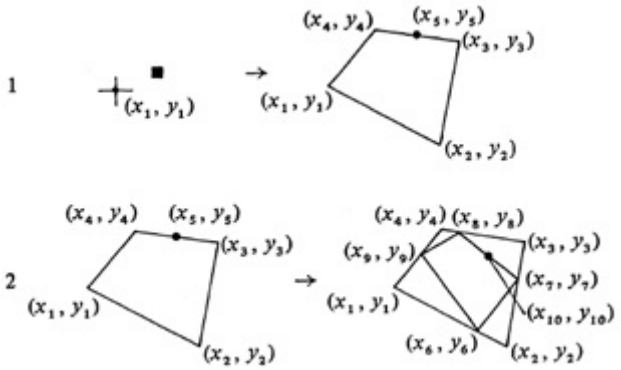


FIGURE 2.24: Parametric labeled shapes rules [143]

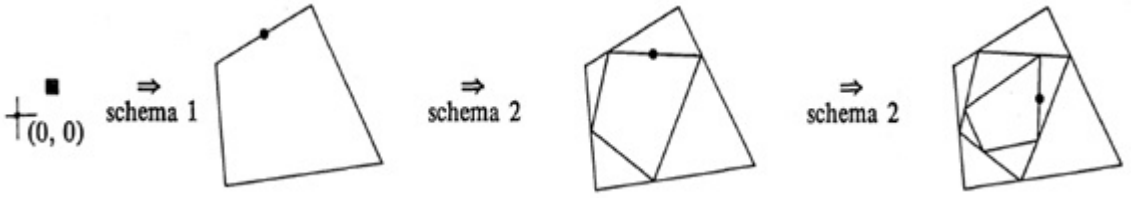


FIGURE 2.25: Parametric labeled shapes derivation [143]

- $V_M = \left\{ \square \right\}$,
- $V_T = \left\{ \square \right\}$,
- $R =$ the set in Figure 2.26,
- $S = \square$.

Some images generated by the set of rules in Figure 2.26 are shown in Figure 2.27.

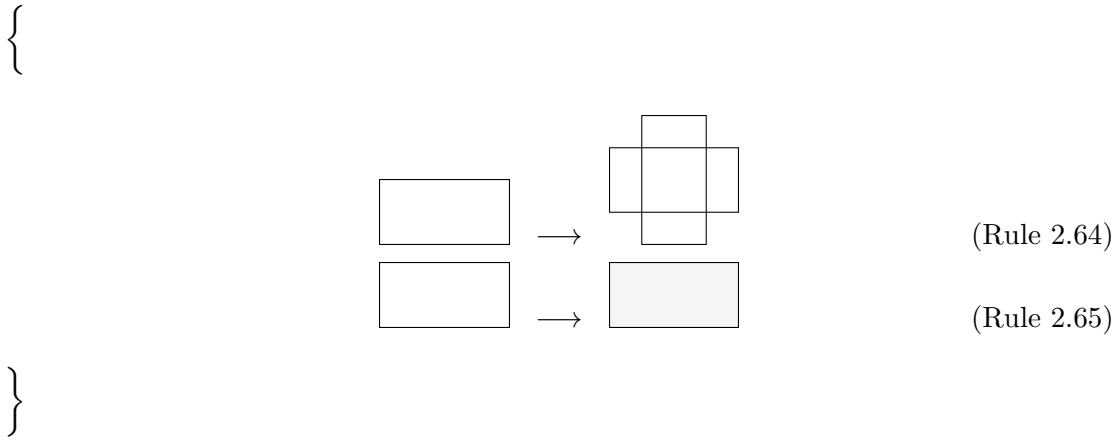


FIGURE 2.26: The set of rules for $G_{\mathrm{SG}-1}$ in Example 2.3.8.

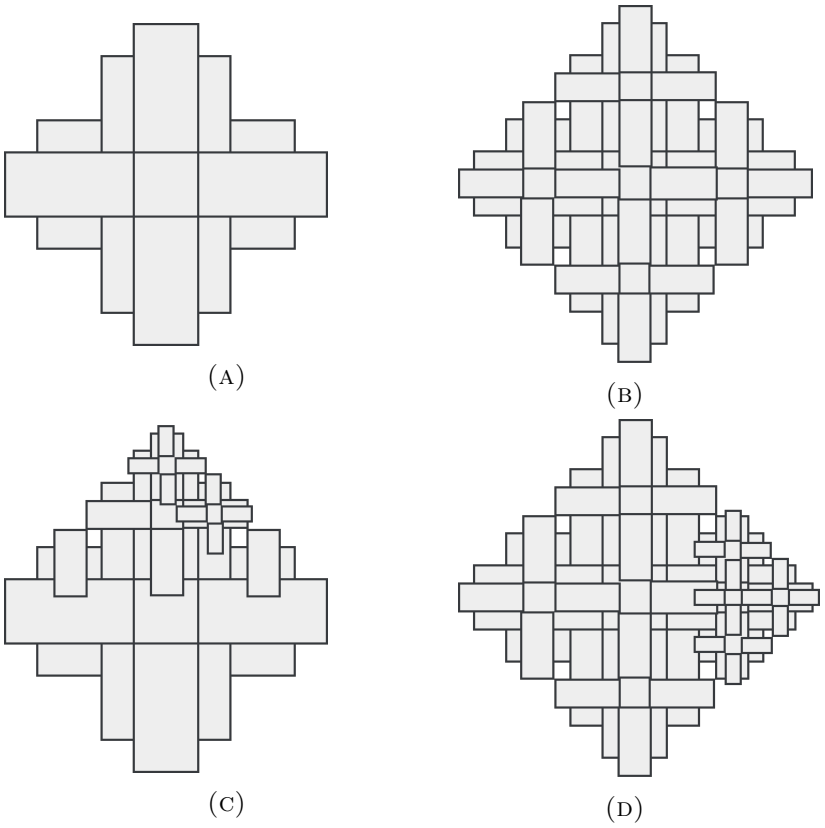


FIGURE 2.27: Some images generated by $\mathcal{G}(G_{\mathrm{SG}-1})$.

The demonstration of the derivation showing the generation of the image in Figure 2.27(A) is shown in Figure 2.28.

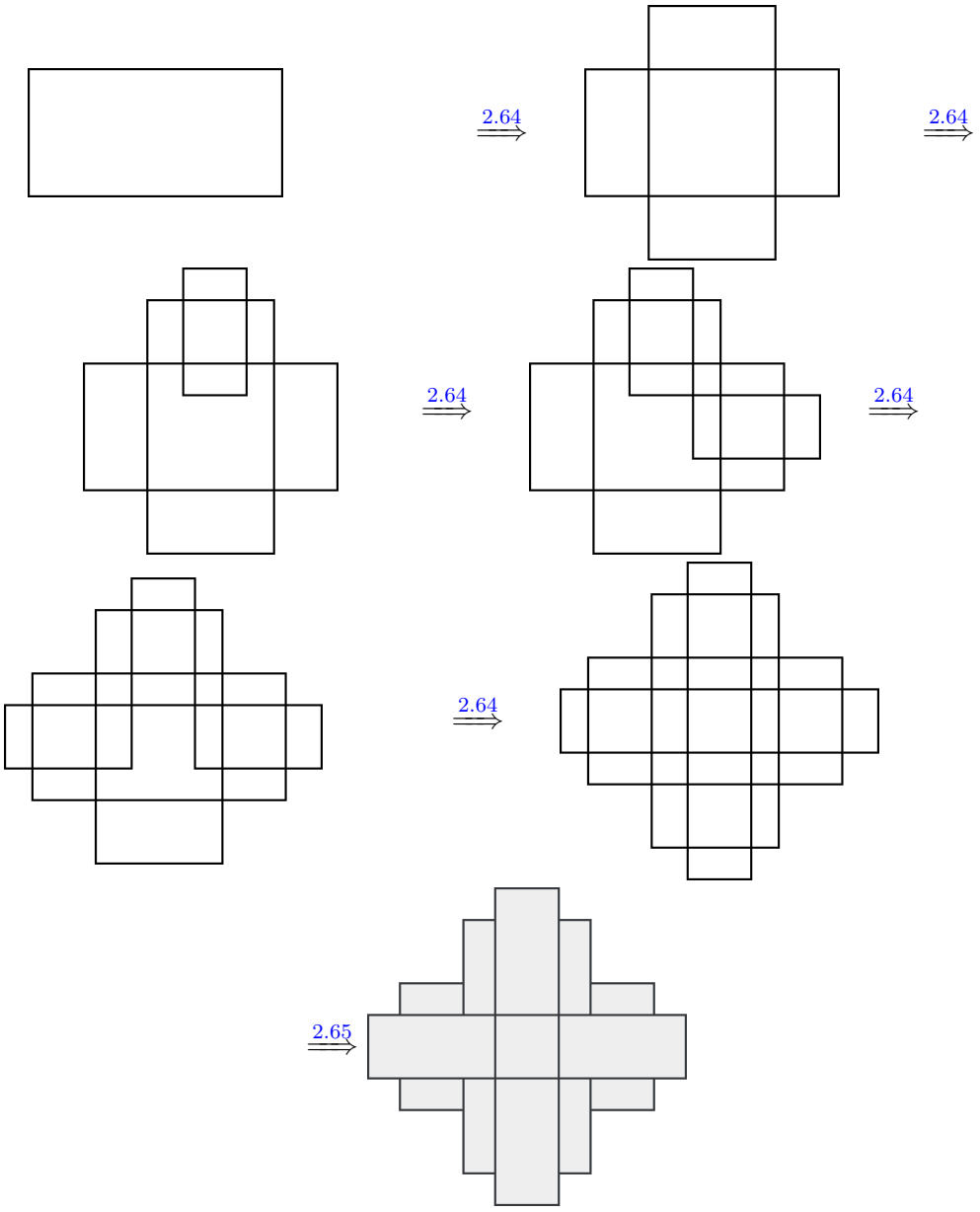


FIGURE 2.28: The derivation of the image in Figure 2.27(A).

Observe that Rule 2.64 can be recursively applied at any time without restriction or control until Rule 2.65 is applied to terminate the process. This can make the rules to generate an infinite number of images whose parts cannot be controlled during derivation. Figures 2.27(A)–2.27(D) also show that any part of the images can grow infinitely without control. There is no way to write the rules in Figure 2.28 to generate only images between Figures 2.27(A)–2.27(B) except the images are manually selected, that is, images whose similarities are close. Moreover, it is not possible to render the images after derivation using another shape without rewriting the rules with that particular

shape (see Figure 2.28). These challenges are due to shape grammars are context free. To address these challenges identified, the definition of an additive shape grammar system and BCSGs are given in Chapter 3. The additive shape grammars can allow any shape to be used to render images after derivation in a controlled manner with the help of bag context.

Next, we give another example to demonstrate how shape grammars work. Example 2.3.9 is adapted from [114].

Example 2.3.9. Consider a shape grammar $G_{\text{SG}-2} = (V_M, V_T, R, S)$, where:

- $V_M =$  ,
- $V_T =$  ,
- $R =$ the set in Figure 2.29,
- $S =$  .

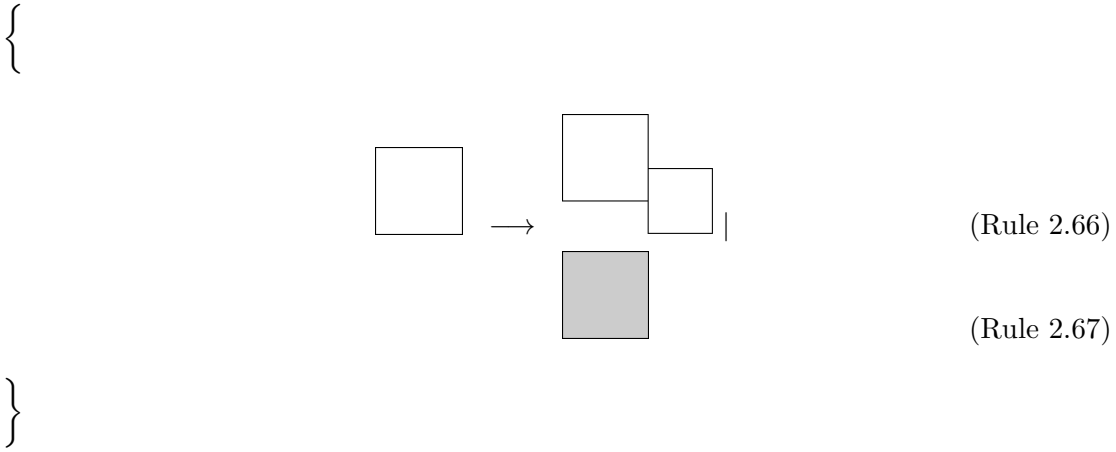


FIGURE 2.29: The set of rules for $G_{\text{SG}-2}$ in Example 2.3.9.

Some images generated by the set of rules in Figure 2.29 are shown in Figure 2.30.

Observe that there is no clear similarity between the images in Figures 2.30(A)–(D), given that they are generated by the same grammar. This is because Rule 2.66 can be recursively applied at any time without restriction or control until Rule 2.66 is applied to terminate the process. There is no way to control the generation process to ensure the images are generated with a common similarity. For instance, there is no way to control the rules and the derivation to generate images that are similar to the image in

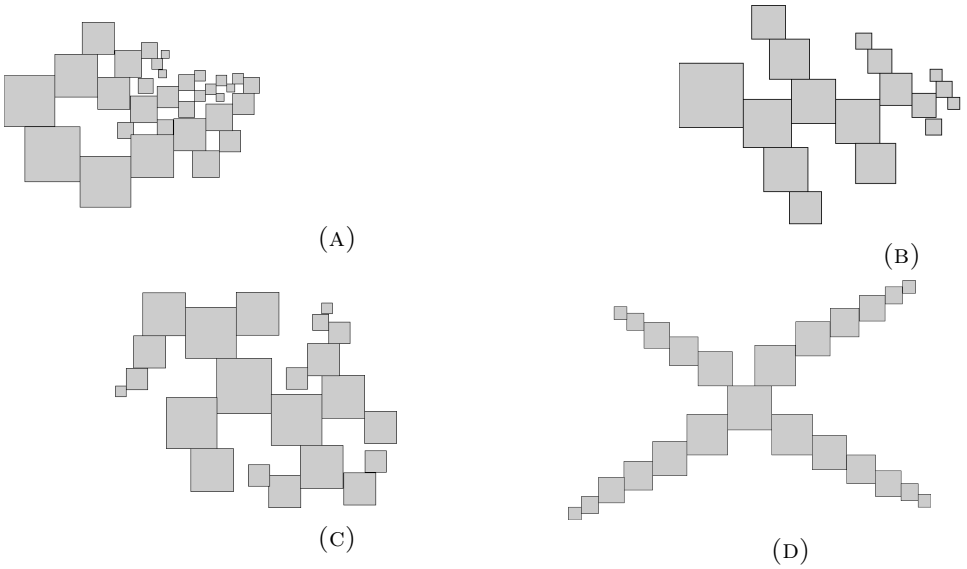


FIGURE 2.30: Some images generated by $\mathcal{G}(G_{\mathrm{SG}-2})$.

Figure 2.30(D) only. To generate similar images, they require users to generate different images and manually select the images that are similar. It is difficult to know if the next image to be generated will be similar to a selected image. Furthermore, there is no possibility to use any other shape apart from the ones used in the rules to render the images after derivation, see Figure 2.29.

Since the development of shape grammars, they have been widely used in architectural design, engineering design and product design [5, 21, 84, 85, 86, 143, 144, 145, 146, 148, 149]. We consider each shape grammar designs in Sections 2.3.10.1–2.3.10.3.

2.3.10.1 Shape Grammars in Architectural Design

In Bernao [16], shape grammars were used to develop an urban plan with increased flexibility. Here, rules were applied recursively to generate a design and/or plan. The design was used as a system solution because of the perceived quality in terms of efficiency in plan and flexibility rather than the specific one that does not apply rules recursively (see Figure 2.31).

Herbert et al [63], proposed a shape grammar based building plan for Ndebele homesteads. The shape grammar was used to generate designs within the social customs and structural properties of a traditional settlement removing the effect of external influence. Grammar rules were used to generate the floor plans of Ndebele homesteads. In this work, they considered the use of grammars to generate the courtyards, cattle kraal, meeting places, etc (see Figure 2.32).

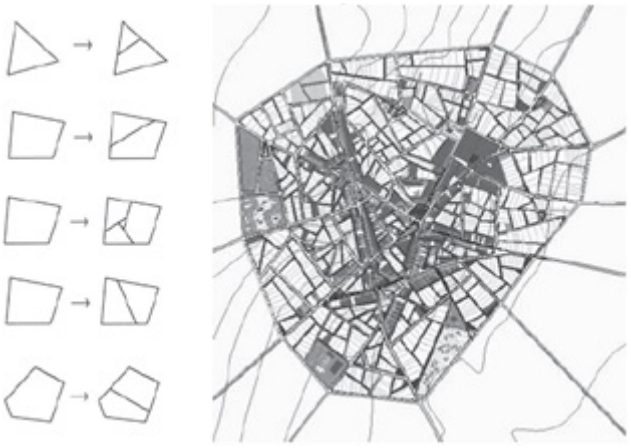


FIGURE 2.31: Urban generation rules and plan example of a possible solution generated by a recursive application of a rule [16]

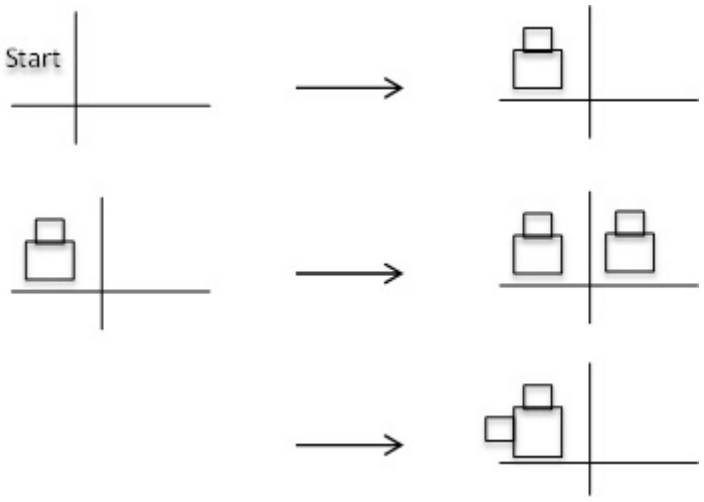


FIGURE 2.32: Grammar for design of Ndebele homesteads which represents the domain of first and second wives [63]

According to Sam et al [116], the approach of natural language processing (NLP) techniques were used to model the architecture of a building. Shape grammars were generated and used to analyze basic Palladian-style floor plans (see Figure 2.33) and preliminary results were presented using NLP tools. Other work on shape grammars in architectural design include but are not limited to Queen Anne houses [48], Prairie houses [90], Palladian villas [148], Alvaro Siza's patio houses [37], Traditional Chinese Architecture [94], etc.

2.3.10.2 Shape Grammars in Engineering Design

In Shea and Cagan [132], shape grammar rules were used to design a truss. For innovative and optimal performance, grammar rules were used to define the space of the design, make the topological and geometrical changes on the original truss. Shape adjustment,

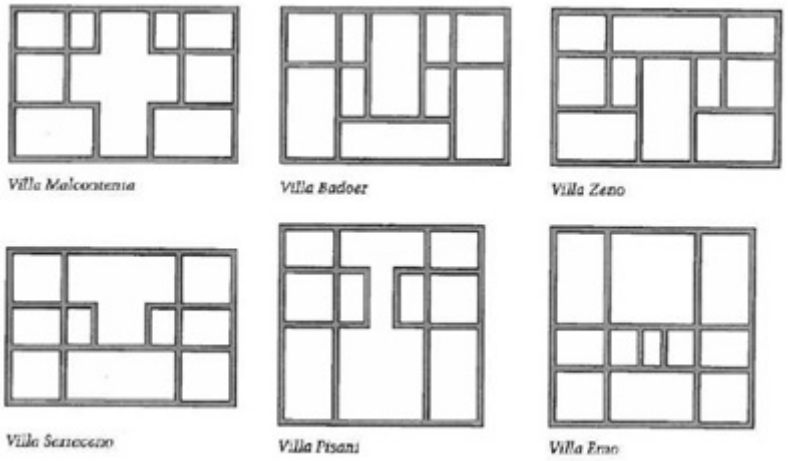


FIGURE 2.33: Floor plans for some Palladian Villas based on a 5 x 3 grid [148].

such as modification made on the position of a joint in the design and size modification, such as changes made on the cross section area of a single member is part of the geometrical change [5, 132]. According to Chase and Liew [20], Functional Behavioural and Structural (FBS) characteristics of designs were transformed using rule replacement. In this work, the FBS description was defined from the derivation of the original design using rules with associated graphs. Modification of FBS rule properties is done by selecting the rule from the rule library which replaces the original grammar rules. Li and Schmidt [96] worked on a process for customizing grammar rules of Epicyclic Gear Trains (EGT). In their work, new grammars were generated to change the functional structures explained in a chart symbol. The system adds a new grammar rule to the existing EGT rules and applies the rules in a dissimilar (different from the existing EGT rules) arrangement to produce a unique model. The aforementioned happens without removing the initial EGT grammar to retain its ability to produce all of its last designs.

2.3.10.3 Shape Grammars in Product Design

Orsborn et al [110] defined a parametric shape grammar for three classes of vehicles comprising; Pickups, Sports Utility Vehicles (SUVs), and Saloons. The grammars are made up of general modification rules and vehicle characteristic rules. General modification rules modify the shape at hand during the development process. The vehicle characteristic rules include universal rules and class specific rules. The former are rules that can be applied to any class while the latter are rules that can be applied to a specific class of vehicles. Combining rules with simple parametric changes of a vehicle produces hybrid and unique vehicles. Also, McCormack and Cagan [103] used parametric shape grammars to conceal the identity of Buick cars from 1947 to 2002. They used the grammar to rebuild known Buicks and introduced new Buick ideas. A sample Buick

was reformed with diverse properties as an essential component like fenders and hoods. Ahmad and Chase [2, 3] illustrated shape grammar rules for mobile phones and Greek temple facades. Their studies concentrated on using a narrative structure to improve the assessment of diverse construction features produced by the improved grammar rules. The transformation available relies on rule modification tools being addition, deletion, and replacement or substitution to transform either shapes or structural relations and their characterization. Other shape grammars in product design include but are not limited to Pencil Grammar [106], Coffeemaker Grammar [1], BMW grammar [38], Beetle grammar [10], Cellular automata rule patterns [135], etc.

2.3.10.4 Summary of Shape Grammars

The approach of shape grammars for the generation of an infinite number of images has been and is still being applied in different spheres of life, especially in architectural design, engineering design, product design, and decorative art. We use Table 2.2 to summarise the contributions made by different authors in the area of shape grammar systems.

TABLE 2.2: Summary of works on shape grammars discussed

SN	Author(s)	Contribution(s)
1.	Stiny and Gips [147]	Shape grammar in painting (geometry and sculpture)
2.	Stiny [143]	Shape grammar formalism
3.	Tapia [159]	Simple shape grammars
4.	You and Fu [172]	Syntactic generation of two dimensional shapes
5.	March and Stiny [100]	Shape grammar transformation
6.	Knight [88], Ahmad and Chase [4]	Shape grammar operations
7.	Chin [21], Ozkar and Stiny [112]	Shape grammar in architectural, engineering and product design
8.	Beirao [16], Herbert et al [63], Prentice et al [116], Flemming [48], Koning and Eizenberg [90], Stiny and Mitchell [148], Duarte [37]	Shape grammar in architectural design
9.	Shea and Cagan [132], Al-kazzaz and Bridges [5], Chase and Liew [20], Li et al [96]	Shape grammar in engineering design
10.	Orsborn et al [110], McCormack et al [103], Ahmad and Chase [3], Ahmad [2], Mitchell [106], Agarwal and Cagan [1], Ducla [38], Arida [10], and Speller et al [135]	Shape grammar in product design

The existing researches that use shape grammars for the generation of images lack the ability to regulate how similar images can be generated in a controlled manner. This is why these grammars generate images that are entirely different from each other, see Figure 2.30 or images that some part is too small for the human eye to see–Figure 2.27. Given the contributions made by other researchers in the area of shape grammar systems for the generation of images, no work to the best of our knowledge has considered bag context added to shape grammar rules for the generation of an infinite number of similar images in a regulated manner. This is the major essence of this research. Therefore, the next section discuses how bag context grammars work and how they can be used to control the application of a rule during derivation.

Next, we discuss bag context grammars in Section 2.4.

2.4 Bag Context Grammars

2.4.1 Introduction

In this section, the definition of Bag Context Grammars (BCGs) and some examples are adapted from Drewes et al [35]. First, we define some notations and symbols used in this part.

2.4.2 Definition of Symbols and Notations

- $\mathbb{N}$ represents the natural numbers.
- $\mathbb{Z}$ represents set of integers.
- A ranked alphabet is a pair $(F, Arity)$ of an ordinary alphabet. Here F is a finite set and $Arity$ is a mapping from F to $\mathbb{N}$ of the form $F \rightarrow \mathbb{N}$.
- f^k represents a ranked alphabet symbol where f is the symbol and k is the rank.
- A signature is a set Σ of ranked symbols f^k (where $k \geq 0$ is the rank).
- T_Σ represents the smallest set such that a $\in T_\Sigma$ for all $a^0 \in \Sigma$ and $f[t_1,\ldots,t_k] \in T_\Sigma$ for all $f^k \in \Sigma$ $(k \geq 1)$ and $t_1,\ldots,t_k \in \Sigma$.
- $T_{\Sigma \bigcup N}$ represents trees over $\Sigma \cup N$
- $\mathbb{Z}^I$ represents the vector of integers, where I is of the form $\{1,\ldots, \mathbb{Z}\}$.
- $\mathbb{Z}_\infty$ represents $\mathbb{Z} \cup \{-\infty, \infty\}$

- If $I = \{1, \ldots, k\}$, then elements of $\mathbb{Z}_{\infty}^{I}$ are written as $k-$tuples.
- $f^{(2)}$ represents a tree of two leaves or nodes
- $g^{(1)}$ represents a tree of one leaf or node
- $h^{(0)}$, $a^{(0)}$ represents a tree of zero leaves or node

Definition 2.18. A Bag Context (BC) tree grammar is a sextuple of the form $G = (N, \Sigma, R, S, I, \beta_0)$ where

- N is a set of nonterminal symbols of rank 0;
- Σ is a finite nonempty set called the terminal symbols. $N \cap \Sigma = \emptyset$;
- R is a finite set of rules of derivation;
- S is the initial nonterminal symbol, $S \in N$;
- I is the bag index; and
- β_0 is a vector called initial bag, $\beta_0 \in \mathbb{Z}^I$.

A rule is of the form

$$A \longrightarrow t\,(\lambda, \mu; \alpha),$$

where $A \in N$, $t \in T_{\Sigma \bigcup N}$, λ, $\mu \in \mathbb{Z}_{\infty}^{I}$ are the lower and the upper limits respectively, and $\alpha \in \mathbb{Z}^I$ is the bag adjustment [75].

Let (s, β), $(s', \beta') \in T_{\Sigma \bigcup N} \times {\mathbb{Z}_{\infty}}^{I}$. We have a derivation step from $(s, \beta) = (\upsilon\ [\mathbb{A}], \beta)$ to $((s', \beta') = (\gamma\ [\mathrm{t}], \beta')$, if there is a rule $A \longrightarrow t(\lambda, \mu; \alpha)$ in R, with $\lambda \leq \beta \leq \mu$ and $\beta' = \beta + \alpha$. We denote the derivation step by $(s, \beta) \Longrightarrow (s', \beta')$. The relation $\Longrightarrow^*$ is the transitive and reflexive closure of $\Longrightarrow$. Finally, the language generated by G is $L(G) = \{t \in T_{\Sigma} \mid (\mathrm{s}, \beta_0) \Longrightarrow^* (t, \beta), \beta \in \mathbb{Z}^I\}$. A rule of the form $A \longrightarrow t$ $(\text{-}\infty, \infty; 0)$ will be written as $A \longrightarrow t$ [75].

Next, we demonstrate how the BC tree grammars work in Examples 2.4.1 and 2.4.2.

Example 2.4.1. Let a BC tree grammar $G_{n1} = (N, \Sigma, R, S, \{1\}, (0))$, where $N = \{S, A, B\}$, $\Sigma = \{f^{(2)}, g^{(1)}, h^{(0)}\}$, and

$$
\begin{aligned}
R = \{S &\longrightarrow f[A,B] && (0,0;0)\\
A &\longrightarrow g[A] && (-\infty,\infty;1)\\
B &\longrightarrow g[B] && (-\infty,\infty;-1)\\
A &\longrightarrow h && (0,0;0)\\
B &\longrightarrow h && (0,0;0)\}.
\end{aligned}
$$

The language of the grammar $L(G_{n1}) = \{f[t,t] |\ t \in T_{\{g^{(1)},h^{(0)}\}}\}$. That is, the set of trees t that evolved from the parent tree (root node) $T \in \Sigma$ only.

Figure 2.34 shows the kind of tree the grammar in Example 2.4.1 will produce.

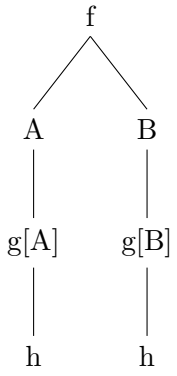


FIGURE 2.34: The tree representation of G_{n1} in Example 2.4.1.

Example 2.4.2. Let $G_{n2} = (N, \Sigma, R, S, \{1, 2, 3\}, (1,0,0))$, where $N = \{S, A, B\}$, $\Sigma = \{\mathrm{f}^{(2)}, \mathrm{a}^{(0)}\}$, and

$$
\begin{aligned}
R = \{S &\longrightarrow f[A,A] && ((1,0,0),(\infty,\infty,0);(-1,2,0))\\
A &\longrightarrow B && ((0,1,0),(0,\infty,\infty);(0,-1,1))\\
B &\longrightarrow S && ((0,0,1),(\infty,0,\infty);(1,0,-1))\\
S &\longrightarrow a && ((1,-\infty,-\infty),(\infty,0,0);(-1,-1,-1))\}.
\end{aligned}
$$

Figure 2.35 shows the kind of tree the grammar in Example 2.4.2 will produce.

In the beginning of every "round" (a round is the time it takes to apply the rules to add a level to the tree (see Example 2.4.2), the first bag position contains many illustrations of the nonterminal S in the present intermediate tree (the intermediate tree is the evolving

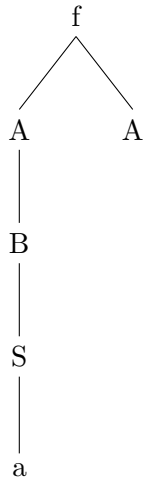


FIGURE 2.35: The tree representation of G_{n2} in Example 2.4.2.

tree). The first rule adds another level, noting the number of *A's* in the second bag position. The second and third rules replicate this number back to the first position and change all nonterminals back into S for a new round to begin. The last rule is applicable only at the start of a round (because of its upper limit). If it is applied, no other rule can be applied again (because the second and third positions are now negative), which means that the derivation terminates by continuous application of the last rule [75].

Next, we present bag context string grammars in Definition 2.19. The Definition of bag context string grammars is adapted from [33].

Definition 2.19. A *BC* string grammar is a sextuple of the form $G = (N, \Sigma, R, S, I, \beta_0)$ where

- N is a set of nonterminal strings of rank 0;
- Σ is a finite nonempty set called the terminal strings. $N \cap \Sigma = \emptyset$;
- R is a finite set of rules of derivation;
- S is the initial nonterminal string, $S \in N$;
- I is the bag index; and
- β_0 is a vector called initial bag, $\beta_0 \in \mathbb{Z}^I$.

A rule is of the form

$$A \longrightarrow s(\lambda, \mu; \alpha),$$

where $A \in N$, $s \in \Sigma \cup N$, λ, $\mu \in \mathbb{Z}_{\infty}^{I}$ are the lower and the upper limits respectively, and $\alpha \in \mathbb{Z}^{I}$ is the bag adjustment [75]. The derivation step in this type of grammar is the same as in a bc tree grammars (see Definition 2.18).

We now demonstrate this notion in Example 2.4.3. Let G_{n3} be a bc string grammar, for each $n \geq 2$, such that $L(G_{n3}) = \{{a_1}^m {a_2}^m \ldots {a_n}^m \mid m \geq 1\}$.

Example 2.4.3. Consider $G_{n3} = (N, \Sigma, R, S, \{1\}, (0))$, where $N = \{S, S_1, S_2, \ldots, S_n\}$, $\Sigma = \{a_1, a_2, \ldots, a_n\}$,

$$
\begin{aligned}
R = \{S &\longrightarrow S_1 S_2 \ldots S_n && (0,0;1)\\
S_1 &\longrightarrow a_1 S_1 && (1,1;1)\\
S_1 &\longrightarrow a_1 && (1,1;-1)\\
S_2 &\longrightarrow a_2 S_2 && (2,2;1)\\
S_2 &\longrightarrow a_2 && (0,0;0)\\
S_3 &\longrightarrow a_3 S_3 && (3,3;1)\\
S_3 &\longrightarrow a_3 && (0,0;0)\\
&\vdots\\
S_n &\longrightarrow a_n S_n && (n,n;1-n)\\
S_n &\longrightarrow a_n && (0,0;0)\}
\end{aligned}
$$

Figure 2.36 shows some of the set of strings the grammar G_{n3} in Example 2.4.3 will produce or generate.

$$\{S_1 S_1 \ldots S_n,\ a_1 S_1,\ a_1,\ a_2 S_2,\ a_2,\ a_3 S_3,\ a_3,\ \ldots,\ a_n S_n,\ a_n\}$$

FIGURE 2.36: The set of strings produced by G_{n3} in Example 2.4.3.

The bag context grammars where used generate tree and string languages. However, research has proven that bag context grammar can be added to picture grammar rule to generate picture languages in a controlled way. This approach is called bag context picture grammars (BCPGs) [43].

Next, we present BCPGs.

2.5 Bag Context Picture Grammars

2.5.1 Introduction

BCPGs is regarded as a special case of bag context tree grammars, where each tree is such that each node has n^2 children, where $n = 1, 2, \ldots$. BCPGs are also context free grammars whose rules are controlled by a k–tuple of integers, the bag. The bag changes during the derivation of a picture. A rule in the grammar can only be applied if the bag at that point is within the range defined by the lower and upper limits of the rule. When a rule is applied, it causes the bag's values to change by adding the bag adjustment, which is part of the rule [43]. In Ewert et al [43], BCPGs were used to generate structurally similar images using a refinement approach. Here, pictures are considered to be structurally similar when they share similar characteristics based on the way they were generated, such that, the pictures contain the same sub–pictures or they have the same number of refinement levels.

In BCPGs, a derivation starts with an initial square, then the square divides into smaller squares by the application of a rule. This process is what is called refinement approach in this part. The formal definition and example of BCPGs are presented in the next section.

2.5.2 Formal Definition of Bag Context Picture Grammars

We refer to the definition of notations in Section 2.4.2 of this chapter, now we present the definition of BCPGs in Definition 2.20. The definitions presented in this part are adapted from [43].

Definition 2.20. A bag context picture grammar, $G = (V_M, V_T, R, (S, \sigma), I, \beta_0)$, has a finite alphabet V of labels, consisting of disjoint subsets V_M of variables and V_T of terminals. R is a finite set of rules. There is an initial labelled square (S, σ), with $S \in V_M$. Finally, I denotes a finite bag index set and $\beta_0 \in \mathbb{Z}^I$ the initial bag.

A rule in R is of the form $A \rightarrow [x_{11}, x_{12}, \ldots, x_{mm}]$ $(L_\mathrm{b}, U_\mathrm{b} ; \delta)$, $m \in \mathbb{N}_+$, where $A \in V_M$, $[x_{11}, x_{12}, \ldots, x_{mm}] \subseteq V$, $L_\mathrm{b}, U_\mathrm{b} \in \mathbb{Z}^I_\infty$ and $\delta \in \mathbb{Z}^I$. The k-tuples L_b and U_b are the lower and upper limits, respectively, while δ is the bag adjustment. For simplicity, we write a rule of the form $A \rightarrow [x_{11}, x_{12}, \ldots, x_{mm}]$ $(-\infty, \infty; 0)$ as $A \rightarrow [x_{11}, x_{12}, \ldots, x_{mm}]$.

Definition 2.21. A pictorial form is any finite set of non-overlapping labelled squares in the plane. The size of a pictorial form, Π, is the number of squares contained in it, denoted by $\mid \Pi \mid$. If Π is a pictorial form, we denote by $l(\Pi)$ the set of labels used in Π.

Definition 2.22. Let $G = (V_M, V_T, R, (S, \sigma), I, \beta_0)$ be a BCPG, Π and Γ pictorial forms, and $\beta, \beta' \in \mathbb{Z}_\infty^I$. Then we write $(\Pi, \beta) \Longrightarrow (\Gamma, \beta')$ if there is a rule $A \longrightarrow [x_{11}, x_{12}, \ldots, x_{mm}]$ $(L_b, U_b ; \delta)$ in R, Π contains a labelled square (A, α), $L_b \leq \beta \leq U_b$, $\Gamma = (\Pi \setminus \{(A, \alpha)\}) \cup \{(x_{11}, \alpha_{11}), (x_{12}, \alpha_{12}), \ldots, (x_{mm}, \alpha_{mm})\}$ and $\beta' = \beta + \delta$, $\Longrightarrow^*$ denotes the reflexive transitive closure of $\Longrightarrow$.

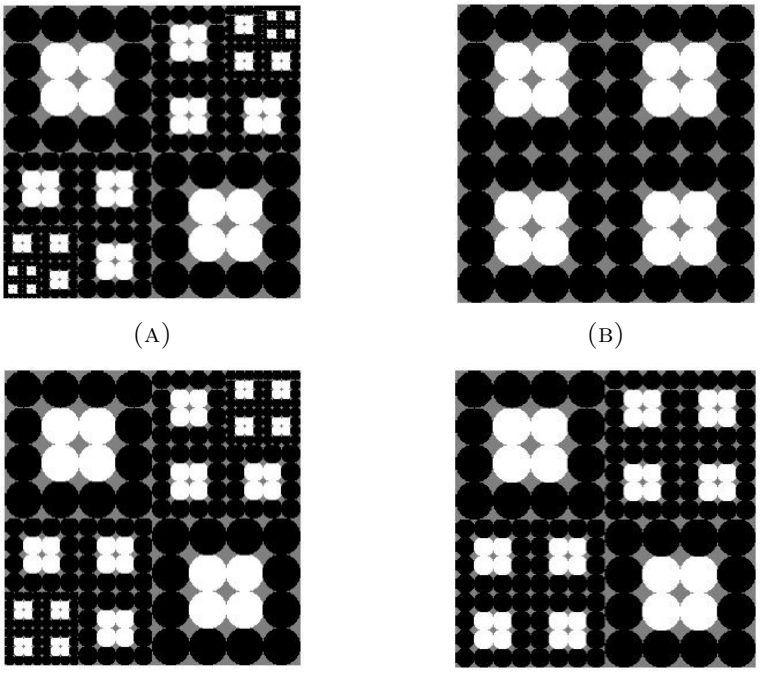
Definition 2.23. Let $G = (V_M, V_T, R, (S, \sigma), I, \beta_0)$ be a BCPG. The set $\mathcal{G}(G) = \{\Phi \mid (\{(S, \sigma)\}, \beta_0) \Longrightarrow^* (\Phi, \beta)$, for $\beta \in \mathbb{Z}_\infty^I$ and $l(\Phi) \subseteq V_T\}$ is called the (bag context) gallery generated by G. An element of $\mathcal{G}(G)$ is called a picture.

2.5.3 Illustration of BCPGs

Examples 2.5.1 and 2.5.2 are adapted from [76].

Example 2.5.1. Consider a gallery of images with four sub pictures, each sub picture can be recursively divided into four sub pictures. Let BCPGs, $G_{\text{BCPGs-Subimage}} = (V_M, V_T, R, (S, \sigma), \{1, 2, \ldots, 7\}, \{1, 0, 0, 0, 0, 0, 0\})$ where $V_M = \{S, A, B, A', B', F, C\}$, $V_T = \{w, b\}$, and R is shown in Figure 2.38.

Some pictures in $\mathcal{G}$ produced by $G_{\text{BCPGs-Subimage}}$ are shown in Figure 2.37.



(A) (B) (C) (D)

FIGURE 2.37: Some of the pictures in $\mathcal{G}$ of $G_{\text{BCPGs-Subimage}}$.

The strategy is that the first to seventh bag positions count the occurrences of the variables in V_M. Consider Rules 2.69 and 2.70 of Figure 2.38. Each can only be applied

if the second and third position in the lower limit is at least 1, that is if there is at least one occurrence of each of the variables A and B in the pictorial form. Next, we demonstrate that BCPGs can generate images within some limited bounds, see Example 2.5.2

$$
R = \left\{
\begin{array}{rcll}
S & \longrightarrow & [A,C,C,B]((1,0,0,0,0,0,0,0),\infty;(-1,1,1,0,0,0,2)) & \text{(Rule 2.68)} \\
A & \longrightarrow & [A',C,C,C]((0,1,1,0,0,0,0,0),\infty;(0,-1,0,1,0,0,3)) \mid & \text{(Rule 2.69)} \\
 & & F((0,1,1,0,0,0,0,0),\infty;(0,-1,0,0,0,1,0)) & \text{(Rule 2.70)} \\
B & \longrightarrow & [C,C,C,B']((0,0,1,1,0,0,0,0),\infty;(0,0,-1,0,1,0,3)) \mid & \text{(Rule 2.71)} \\
 & & F((0,0,1,0,0,1,0,0),\infty;(0,0,-1,0,0,1,0)) & \text{(Rule 2.72)} \\
A' & \longrightarrow & A((0,0,0,1,1,0,0,0),\infty;(0,1,0,-1,0,0,0)) & \text{(Rule 2.73)} \\
B' & \longrightarrow & B((0,1,0,0,1,0,0,0),\infty;(0,0,1,0,-1,0,0)) & \text{(Rule 2.74)} \\
F & \longrightarrow & C((0,0,0,0,0,1,0,0),\infty;(0,0,0,0,0,-1,1)) & \text{(Rule 2.75)} \\
C & \longrightarrow & [b,b,b,b,b,w,w,b,b,w,w,b,b,b,b,b] & \\
 & & ((0,0,0,0,0,0,0,1),\infty;(0,0,0,0,0,0,0,-1)) & \text{(Rule 2.76)}
\end{array}
\right\}
$$

FIGURE 2.38: The set of rules for $G_{\text{BCPGs-Subimage}}$.

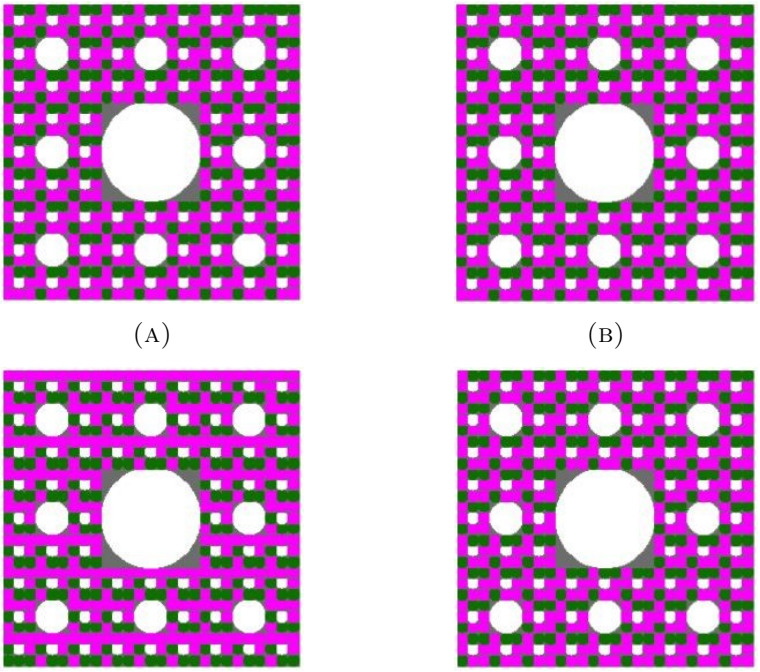
Next, Example 2.5.2 shows that BCPGs can regulate the refinement levels of images during a derivation.

Example 2.5.2. Consider a gallery of pictures whose refinement levels are limited during a derivation, such that, only the third refinement pictures are generated. Let BCPGs $G_{\text{BCPGs-LimitingRefinementLevel}} = (V_M, V_T, R, (S,\sigma), \{1,2,\ldots,8\}, \{1,0,0,0,0,0,0,0\})$ where $V_M = \{S, T, U, F, C\}$, $V_T = \{b, w, g\}$, and R is shown in Figure 2.40.

Some pictures in $\mathcal{G}$ produced by $G_{\text{BCPGs-LimitingRefinementLevel}}$ are shown in Figure 2.39.

To achieve the generation of third refinement pictures, the value of the last position of the lower limit in Rule 2.79 of Figure 2.40 is kept at 72. This ensures that the colouring of pictures only starts after the initial square has been subdivided thrice. Although BCPGs and BCSGs use bag context for the generation of similar images, BCPGs use a refinement approach of image generation while our type of BCSGs use an additive approach of image generation. To convert the theories of these grammars to real images, there is a need to development an image generating tool that can serve this purpose. This approach of converting grammar thoeries to images is called grammar interpreter. One such example is the shape grammar interpreter.

In the next section, we present shape grammar interpreters and show how they work.



(A) (B)

(C) (D)

FIGURE 2.39: Some of the pictures in $\mathcal{G}$ of $G_{\text{BCPGs-LimitingRefinementLevel}}$.

$$
R = \left\{
\begin{array}{lcll}
S & \longrightarrow & [T,T,T,T,w,T,T,T,T]((1,0,0,0,0,0,0,0), & \\
 & & (\infty,\infty,0,\infty,\infty,\infty,\infty,\infty);(-1,8,0,0,0,0,0,0)) & \text{(Rule 2.77)} \\
T & \longrightarrow & U\ ((0,1,0,0,0,0,0,0),(0,\infty,\infty,0,\infty,\infty,\infty,71); & \\
 & & \quad (0,-1,1,0,0,0,0,1)) \mid & \text{(Rule 2.78)} \\
 & & F\ ((0,1,0,0,0,0,0,72),(0,\infty,0,0,\infty,\infty,\infty,\infty); & \\
 & & \quad (0,-1,0,1,0,0,0,0)) \mid & \text{(Rule 2.79)} \\
 & & C\ ((0,1,0,1,0,0,0,0),\infty,(0,-1,0,0,1,0,0,0)) & \text{(Rule 2.80)} \\
U & \longrightarrow & S((0,0,1,0,0,0,0,0),(\infty,0,\infty,\infty,\infty,\infty,\infty,\infty); & \\
 & & (1,0,-1,0,0,0,0,0)) & \text{(Rule 2.81)} \\
F & \longrightarrow & C\ ((0,0,0,1,0,0,0,0),(\infty,0,\infty,\infty,\infty,\infty,\infty,\infty); & \\
 & & \quad (0,0,0,-1,1,0,0,0)) & \text{(Rule 2.82)} \\
C & \longrightarrow & b\ ((0,0,0,0,1,0,0,0),(\infty,\infty,\infty,\infty,\infty,4,\infty,\infty); & \\
 & & \quad (0,0,0,0,-1,1,0,0)) \mid & \text{(Rule 2.83)} \\
 & & g\ ((0,0,0,0,1,5,0,0),(\infty,\infty,\infty,\infty,\infty,\infty,2,\infty); & \\
 & & \quad (0,0,0,0,-1,0,1,0)) \mid & \text{(Rule 2.84)} \\
 & & C\ ((0,0,0,0,1,5,3,0),\infty;(0,0,0,0,0,-5,-3,0)) & \text{(Rule 2.85)}
\end{array}
\right\}
$$

FIGURE 2.40: The set of rules for $G_{\text{LimitingRefinementLevel}}$.

2.6 Shape Grammar Interpreters

A shape grammar interpreter is a software application that converts or models shape grammar rules. According to Mhin et al [105], an interactive design of probability density functions for shape grammars was introduced as a framework that enables a user to interactively design a probability density function for a shape grammar and to generate models according to the designed pdf. They extended existing exploratory modelling tools that are suitable to select a single model from a shape space to modelling a distribution of shapes. They also proposed a user interface to display, sort, and sample models to enable a user to quickly assign preference scores. In Crumley et al [28], shape grammars are specified and iterated to produce a required shape using a shape grammar interpreter algorithm (see Algorithm 1). The user provides a set of shape grammar rules and the initial shape (axiom). A new shape is produced when a rule set is run on the axiom by modifying the existing shape on each iteration. Application of the algorithm on shapes will generate a set of shapes of the form in Figure 2.41.

Algorithm 1 Shape Grammar Interpretation

```
1:  currShapeSet ← {Axiom}
2:  iteration ← 0
3:  while iteration < MaxIteration and currShapeSet.hasNonTerminals() do
4:      if parallelExecution = TRUE then
5:          for i ∈ currShapesSet do
6:              i ← doRuleDerivation(i)
7:      else
8:          a ← getFirstNonTerminal(currShapeSet)
9:          a ← doRuleDerivation(a)
10:     iteration ← iteration + 1
11: for i ∈ currShapeSet do
12:     if i.hasSymmetry() then
13:         s ← i.createSymmetricCopies()
14:         currShapeSet.insert(s, i − 1)
```

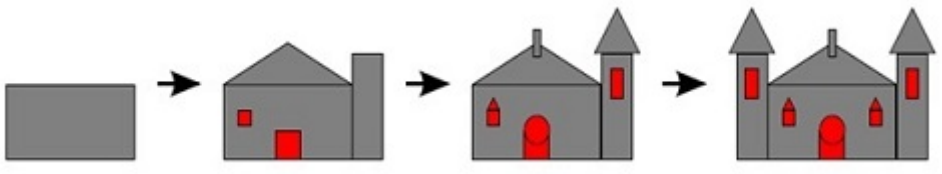


FIGURE 2.41: Simple shape grammar being interpreted to form a basic building [28].

Trescak et al [162] developed a shape grammar interpreter that allows users to interactively indicate the shape and rules also, have total control throughout the development process. Their system supports subshape detection (see Algorithm 2) and it is a platform

independent and open source application. The system supports addition, substitution, and modification rules.

Algorithm 2 Subshape detection algorithm

```
1: Input: inputShape, subShape
2: Output: Collection of subshapes
3: maxLines ← CreateMaxLines(inputShape)
4: subMaxLines ← CreateMaxLines(subShape)
5: inters ← CreateInters(MaxLines)
6: subInters ← CreateInters(subMaxLines)
7: transfs ← FindTransfs (subShape, inters, subInters)
8: for ∀ transfs do
9:     if ∀ subMaxLines ⊆ maxLines then
10:        subShapes ← TransShape (subShape)
   return subShapes
```

Trescak et al [162] used a modified version of Krishnamurti's algorithm [92] for subshape detection (see Algorithm 2). Rules are processed from left to right by transformation. Furthermore, Daniel et al [29], described an interactive system that enables both creating new buildings in the style of others and modifying existing buildings in a quick and intuitive manner. Santiago et al [129], focused on a specific editing application: copy and paste for procedural building models. They provided a simple to use, intuitive editing metaphor that allows non-experts to generate new content using pre-existing rulesets. In [134], Simon et al, proposed a real-time rendering approach for procedural cities. They implemented a new lightweight grammar representation that compactly encodes facade structures and allows fast per-pixel access.

Further, Jowers et al [69] provided a formal proof that adopting a finite fixed structure limits the capability of a shape grammar and can make it impossible to fully implement shape computations. Here, full implementation is in the sense that all possible rule applications are supported. The shape grammars investigated have previously been used to support arguments that in shape computations, shape structures should not be fixed. Other shape grammar interpreters include, but are not limited to, spatial grammar implementation [104], style grammars for interactive visualisation of architecture [7], prototype system for two and three dimensional shape grammars [95], parameterized three dimensional spatial grammars [66], shape grammar interpreter for curved shapes [78], shape grammar implementation for marching planning [40], shape detection with vision [79], Malag [27], etc.

However, the majority of these shape grammar interpreters do not allow for the use of bag context shape grammars to automatically generate similar images. In order to check the similarity of the images generated by these interpreters as discussed, because a number of them generate images that are entirely different or images that are identical,

there is a need to consider different models of image similarity. To ascertain which is best to use in terms of accuracy given the type of image. Next, we discuss different models of image similarity.

2.7 Models of Image Similarity

The similarity of images can be measured based on text or content [8, 39]. The similarity of images based on text involves the description of images as a set of free text phrases or keywords. Here, the comparison is based on the exact match of the query text. The similarity based on content involves the visual and semantic content of the image called image features, such as; colour, the image pattern, texture, location and shape [13]. In the context of this work, we focus on the models of similarity of images based on content. These types of models, also known as content-based image retrieval (CBIR) or query by image content (QBIC), involve techniques such as similarity by colour, shape, texture, statistical parameters, etc. [127].

2.7.1 Model of Image Similarity by Colour

Image similarity by colour is one of the most widely used similarity measures for images because of its robustness [127]. Image similarity by colour is mostly determined using colour descriptor models [131]. Many colour descriptor models are being used for image similarity, depending on the type of image. One such model is the Spatial Colour Distribution Descriptor (SpCD) [131]. The SpCD uses a machine learning approach to extract and evaluate the spatial colour information and matching of image features [130].

To evaluate the similarity of images using SpCD there are two major steps involved, extraction of image colour information and matching. The extraction of the spatial distribution of colour information of the image is performed in the following steps: the input image is partitioned into 64 blocks (8×8). Then the average of the pixel colour in each block is taken as a representative colour. The RGB colour space of the image is converted to YC_bC_r, which is, Y the luma component, C_b, and C_r the blue-difference and red-difference chroma components of the image. The YC_bC_r is transformed into three 8×8 matrices of 64 discrete cosine transform (DCT) coefficients each. Then a zigzag scan is performed with the three sets of 64 DCT coefficients. Finally, the matching process calculates the distance between two image colour distribution information. This process gives the value 0 if the two images are the same.

This model has proven to detect the similarity of syntactically generated images with a high degree of success and accuracy [77]. Therefore, this type of model will help measure

the similarity of syntactically generated images by grammars, which bag context shape grammars are one of them. The evaluation of BCSG generated image similarity using SpCD is presented in Chapter 6.

Image similarity by colour also involves colour models like colour signatures [127], colour moment [150] and colour histogram [158]. Colour histogram is the most commonly used colour model image similarity technique [128]. It uses three colour channels red, green and blue (RGB) because of the presence of images in the RGB format from image scanners. For a comparison to be successfully carried out, the normalized form of the colour histogram of the image is first derived with the formula:

$$I(i) = \frac{H(i)}{\Sigma_i H(i)} \qquad \text{(Equation 2.1)}$$

where $H(i)$ is the colour histogram, i is the histogram bin, and $I(i)$ is the normalized colour histogram of the image. Then, a histogram intersection (S_c^{HI}) is derived with

$$A = \Sigma_r \min(I_R(r), Q_R(r)) \qquad \text{(Equation 2.2)}$$

$$B = \Sigma_g \min(I_G(g), Q_G(g)) \qquad \text{(Equation 2.3)}$$

$$C = \Sigma_b \min(I_B(b), Q_B(b)) \qquad \text{(Equation 2.4)}$$

Then,

$$S_c^{HI}(I, Q) = \frac{A + B + C}{\min(|\ I\ |, |\ Q\ |)^3} \qquad \text{(Equation 2.5)}$$

where I_R, I_G and I_B are normalized colour histograms of the image to be compared while Q_R, Q_G, Q_B are normalized colour histograms of the target image. $S_c^{HI}(I, Q)$ is the similarity between two images and the value is between 0 and 1. If the value of $S_c^{HI}(I, Q)$ is 1 then it is either the image to be compared is contained in the target image or the target image is contained in the image to be compared. So we say, that the images are identical or that their similarity measure is high. This technique has proven to be faster than shape and texture techniques [74, 127, 128, 150, 158].

2.7.2 Model of Image Similarity by Shape

When two images appear to have the same colour, a shape model of similarity measure becomes an alternative. Here the histogram of the edge directions of the image is defined by the shape peculiarity. The edge properties of the image are derived using the Canny

edge operator [17], then, a histogram intersection is used to retrieve the shape. Two similar images will have different edge points (points in the image with coordinates at the location of an important local intensity change) and different histograms. The edge points of the image are normalized in order to get invariance of the histogram (points that are unique in the histogram) [74, 138]. Xu and Prince [169] proposed a gradient vector flow (GVF) technique to get the shape information of the image by getting a grayscale of the edge of the image. The algorithm for edge image processing is shown in Algorithm 3 [64].

Algorithm 3 Edge Image Detection

1: *Read the image and convert it to gray scale*
2: *Blur the image using a Gaussian filter [11, 72, 173]*
3: *Compute the gradient map of the blurred image*
4: *Compute GVF. (100 iterations and $\mu = 0.2$, where μ is a regulation parameter)*
5: *Filter out only strong edge responses using 2.5σ, where σ is the standard deviation of the GVF*
6: *Converge onto edge pixels satisfying the force balance condition yielding the edge image.*

2.7.3 Model of Image Similarity by Texture

The texture in an image involves the repetitive tone in the image. A co-occurrence matrix (spatial relationship of image pixels) [54] is used to compute the image texture extraction. Algorithm 4 shows the steps for image texture feature extraction as proposed by Gode and Ganar [54].

Algorithm 4 Texture Feature Extraction

1: *Image colour conversion*
2: *Grey scale quantification [51, 60]*
3: *Feature value calculation*
4: *Internal normalization*

2.7.4 Model of Image Similarity by Statistical Parameters

The similarity of images can be derived using some statistical parameters such as mean, median, standard deviation, and correlation coefficient. Here, the image is first converted to a grayscale (see Figure 2.42) to obtain a two dimensional array, say (A) (see Figure 2.43). The length of the array (A) is denoted by (N) and the positions of the array cells are denoted by A_i where $i > 0$. The array cells are filled with values from 0 to 9. If the value obtained after the computation of any of the statistical parameters for a

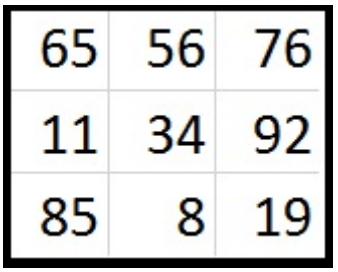


FIGURE 2.42: The gray scale representation of an image [102].

$A =$

i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_N
65	56	76	11	34	92	85	8	19

FIGURE 2.43: Array representation of a gray scale [102].

given image say (X) is the same as the value for another given image say (Y) then the similarity of the images (X and Y) are the same.

The mean μ of an image is given by

$$\mu = \frac{1}{N}\sum_{i=1}^{N} A_i \qquad \text{(Equation 2.6)}$$

The median is calculated by first sorting the array values into numerical or ascending order, then the middle value is picked to represent the median. If the length of the array is an odd number, Then,

$$Median = \left(\frac{N+1}{2}\right)^{th} value \qquad \text{(Equation 2.7)}$$

If the length of the array under consideration is an even number, then the average of the two values in the middle is picked. That is,

$$Median = \frac{\left\lfloor\frac{N+1}{2}\right)^{th} value + \left(\frac{N+1}{2}\right\rfloor^{th} value}{2} \qquad \text{(Equation 2.8)}$$

The Standard Deviation σ of an image is given by

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} \mid A^i - \mu \mid^2} \qquad \text{(Equation 2.9)}$$

The Correlation Coefficient R is the matching between two image matrices A and B (The matrices A and B are the array representation of two different images). See Equation 2.10.

$$R(A,B) = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{A_i - \mu_A}{\sigma_A}\right)\left(\frac{B_i - \mu_B}{\sigma_B}\right) \quad \text{(Equation 2.10)}$$

where A_i represents different values of array A, μ_A represents the mean value of array A, and σ_A represents the standard deviaton of array A. Also, the symbols B_i, μ_B, and σ_B represent diffrent values of aray B, the mean of array B, and the standard deviation of array B respectively [47, 83, 102, 117].

Many systems have been developed based on a CBIR approach. Such systems include, but are not limited to Blobworld [19], Content-Based Image Retrieval from Digital libraries (C-bird) [97], ImageMiner [91], Image RETrieval by Reduction and Overview (ImageRETRO) [165], ImageRover [160], Leiden 19th Century Portrait Database (LCPD) [68], Multimedia Analysis and Retrieval System (MARS) [111], Multimodal Information Retrieval System (MIR) [137], NETRA [99], Picasso [31], and PicToSeek [53].

Following the literature reviewed, we will use the SpCD for the evaluation of the similarity of the images generated by BCSGs. BCSGs generates syntactic images and SpCD has proven to detect the similarity of syntactically generated images with a high degree of success and accuracy [77]. Therefore, this type of model will be appropriate in measuring the similarity of BCSG generated images. To further demonstrate the relevance of the similarity of images that can be checked using these models, visual password schemes become paramount as they use similar images as distractors during authentication.

Next, we discuss visual password schemes in Section 2.8.

2.8 Visual Password Schemes

Authentication in computers is the process of verifying the genuineness of a digital credential. The most common methods of authentication are passwords and biometrics. The former involves the use of numbers and or characters called alphanumeric passwords. Alphanumeric passwords are sometimes forgotten, misplaced or attacked due to the human tendency of using easy to remember passwords. Biometrics involves the use of fingerprint, iris, voice pattern, etc., on an installed hardware device. It is a fact that when people grow old or have an accident their biometric features change. Criminals have taken advantage of these authentication problems to impersonate peoples' digital credentials to perpetrate crimes. Globally, more than 556 million people have become

victims of these crimes in the last 13 years [80, 124]. According to the United States federal trade commission [26], 40% of initial contacts of most fraud cases were by email, and 20% by Internet websites. In this context, it is estimated that 332,646 victims were affected by financial losses totalling $110 billion dollars [26]. In 2015, it is also estimated that R57.8 million was lost due to unauthorized access to computers in South Africa [139].

In the light of the above, visual passwords have been proposed as a replacement for alphanumeric passwords or biometric authentication systems [61, 109]. Visual password schemes (VPSs) are authentication systems that use images for enrolment and authentication [122, 123, 167]. They are classified into recognition or cued recall [167]. The VPSs have memory advantages over alphanumeric passwords because passwords are based on pure recall which can easily be forgotten due to human errors [107]. Studies have shown that pictures are recognized with 98 per cent accuracy more than words and sentences after a long time [139]. According to Shepard [133], seventeen percent recognition error was recorded after viewing 10,000 pictures. This could be part of the reasons why pictures are seen as more easily recalled than passwords [113]. However, use of visual passwords for authentication has been fraught with many challenges such as; biased pass image selection as people tend to select faces from their own race or background or attractive faces or the faces of models which makes it prone for guessing attacks, static image representation during enrolment and authentication which makes it easy for shoulder surfing attacks, consumption of large memory space and high bandwidth connection needed for passing images to and from during authentication [109].

To solve some of these challenges identified, the approach of picture grammars for the generation of abstract images for visual passwords was introduced [109]. The use of these grammars for image generation does not always produce images in a regulated manner. Some of these grammars generate images without considering the similarity between them. We used BCSGs to generate similar images in a regulated manner for use in a prototype VPS. See Chapter 7. Our type of VPS will contribute to improving some of the challenge identified in the existing VPSs. Next is the summary of related work.

2.9 Summary of Related Work

Some literature has stated that grammars are suitable for the generation of an infinite number of images but fail to generate only similar images in a controlled manner. We summarise the drawbacks of the picture grammars in Table 2.3.

TABLE 2.3: Summary of the drawbacks of the picture grammars discussed

S/N	Picture Grammar	Drawback(s)
1.	RCPG	Refines all the parts of the picture at a time during derivation. Although $RPCPGs$ and $RFCPGs$ dealt with that. $RCPGs$ start their derivation by refining the initial shape which is entirely different from the additive type of grammars we are concerned with.
2.	Tree based	No restriction in the application of a rule during derivation. That is, they refine all part of the image during derivation.
3.	L-systems	During a derivation of an images all parts are refined the same way.
4.	IFS	At every step during derivation all parts are refined at the application of the map, it is not possible to use only a selection of the maps available.
5.	Collage	They cannot generate galleries that consist of collages with a non-linear growth in the number of parts. For example, in the typical iteration sequence of the Sierpiński gasket, all parts of each gasket in the sequence must be equally refined.
6.	Chain code	They cannot generate closed curves like rectangles, squares, etc.
7.	Array	Rewriting rules of RAGs or CFAGs cannot sense the context of non-#'s in a pictorial form. Therefore, they cannot generate geometric figures that require non-#-context.
8.	Puzzle	They do not generate galleries such as the set of all hollow rectangles. Although some developments in Puzzle grammar with permitting feature can. They need a large number of rules to generate relatively simple geometrical forms.
9.	Shape	To generate similar images, they require users to generate different images and manually select the images that are similar. It is difficult to know if the next image to be generated will be similar to a selected image. Some lack the ability to regulate how similar images can be generated in controlled manner. They do not allow any shape to be used to render an image after derivation.

Considering the drawbacks of the picture grammars, particularly shape grammars as summarised in Table 2.3, we, therefore, propose an additive shape grammar class with bag context, called BCSGs for the generation of an infinite number of images in a controlled manner. BCSGs will contribute to improving on some of the drawbacks identified in shape grammars. We modify the Crumley et al [28] shape grammar interpretation algorithm by adding some scripts that allow the use of any shape for image rendering after derivation and the addition of bag context during derivation. The SpCD, a content based image retrieval model is used to mathematical check the similarity of the images generated by BCGSs. Then, the similar images generated using BCSGs are implemented in a prototype visual password.

Next, we present the bag context shape grammars in Chapter 3.

Chapter 3

Bag Context Shape Grammars

This chapter is an extended version of the paper published in the IAENG International Journal of Computer Science, 47(1), 2020, 75–86

3.1 Introduction

In this chapter, we first define a new additive shape grammar class that can allow the use of any shape to be used to implement images during rendering and after a derivation. By additive, we mean the application of a rule is done by building upon the axiom (starting symbol or shape) during the generative process. We give an example to demonstrate this notion. Then we formally define the bag context shape grammars and give some examples to illustrate the notion also. Further, we prove that every puzzle grammar with permitting features can be converted to a BCSG.

3.2 Additive Shape Grammars

The concept of additive shape grammar systems was motivated by the fact that most shape grammars do not allow any shape to be used after a derivation to implement the images during rendering (see Section 2.3.10 of Chapter 2). Hence, the need for an additive shape grammar that can allow any shape to be used after derivation to render images. Next, we present some notation used in this part.

3.2.1 Preliminaries

The symbol $\mathbb{N}$ represents the set of natural numbers $\{0, 1, 2, \dots\}$. $\mathbb{N}_+$ represents the set $\{1, 2, \dots\}$. For $k \in \mathbb{N}_+$, the set $\{1, 2, \dots, k\}$ is denoted by $[k]$. $\mathbb{Z}$ represents the set of integers $\{\dots, -2, -1, 0, 1, 2, \dots\}$, then $\mathbb{Z}^2 = \{(x, y) \mid x, y \in \mathbb{Z}\}$. $\mathbb{Z}_\infty$ represents $\mathbb{Z} \cup \{-\infty, \infty\}$. If $I = [k]$, then elements of $\mathbb{Z}_\infty^I$ are written as $k-$tuples. If we have a vector of the form $(k, k, \dots, k)$, then we denote it by $\mathbf{k}$. For example, the vector $(3, 3, \dots, 3)$ will be denoted by $\mathbf{3}$.

A point, (x, y) in $\mathbb{Z}^2$ is a position determined by a pair (x, y), where $x, y \in \mathbb{Z}$. We denote a point by p. A line, l, is a finite length [9]. In this part, a line segment, $\bar{l}$, is determined by any pair of two distinct points (p_1, p_2) on a line together with all the points of the line between p_1 and p_2, where p_1 and p_2 are called the endpoints. A shape, σ, is defined by the area enclosed by a finite set of line segments. A label can be denoted by any of the lowercase or uppercase letters, a–z and A–Z, respectively.

A labelled shape is a pair (A, σ), where A is the label of the shape taken from the uppercase or lowercase letters and the symbol σ is as defined above. A grid is a coordinate plane consisting of a space of small squares, with horizontal (x) axis and vertical (y) axis, see Figure 3.1. Every labelled shape is presented in a unit square on the grid. The initial shape is usually labelled S. The labelled shape (A, σ), is denoted by $(A, (x, y))$ in the rest of this work. Where $(x, y) \in \mathbb{Z}^2$ denotes the lower lefthand corner of the unit square the shape, σ, will be drawn.

An additive shape grammar rule is commonly expressed in the form in Rule 3.1,

$$(A, (x, y)) \longrightarrow \{(A_1, (x_1, y_1)), (A_2, (x_2, y_2)), \dots, (A_i, (x_i, y_i))\} \qquad \text{(Rule 3.1)}$$

where A is a variable (uppercase label) and $(x, y) \in \mathbb{Z}^2$ is as defined above. The arrow "$\longrightarrow$" is interpreted as "transformed to", $A_1, A_2 \dots A_i$ represent variable(s) and terminal(s) (lowercase labels), for $i \in \mathbb{N}_+$ and (x_1, y_1), (x_2, y_2), $\dots$, $(x_i, y_i) \in \mathbb{Z}^2$.

The interpretation is as follows: a rule as in Rule 3.1 can be applied to a developing image if the developing image contains the variable A. Then, we take the set difference of the developing image and the variable A and take the set union of the developing image and $\{(A_1, (x_1, y_1)), (A_2, (x_2, y_2)), \dots, (A_i, (x_i, y_i))\}$.

The number (n) of times a rule (r) is applied during the generative process is denoted by $r^{(n)}$. For instance, $2^{(5)}$ means that rule 2 is applied five times. The operations of shape union and difference treat shapes in the same basic way as the set theoretic operations of union and difference treat sets.

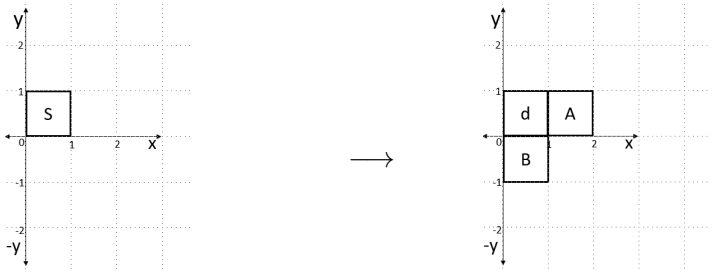
Next, we demonstrate how an additive shape grammar rule is rendered graphically. Let

$$(S,\,(x,\,y)) \longrightarrow \{(d,\,(x,\,y)),\,(A,\,(x+1,\,y)),(B,\,(x,\,y-1))\}$$

be a rule which is also the same as

$$(S,\,(0,\,0)) \longrightarrow \{(d,\,(0,\,0)),\,(A,\,(1,\,0)),\,(B,\,(0,\,-1))\},$$

when $x,\,y = 0$. The rule above will produce Figure 3.1 when rendered using square shapes.



(A) (B)

FIGURE 3.1: A simple shape grammar rule on the grid.

For visualisation purposes, every terminal is associated with a shape and the shape is filled with a chosen colour. The image is rendered to any size according to what is needed.

3.2.2 Formal Definition of Additive Shape Grammars

The definition of additive shape grammars is motivated by definition of shape grammars in [143].

Definition 3.1. An Additive Shape Grammar (ASG) $G = (V_{\mathrm{M}},\,V_{\mathrm{T}},\,R,\,(S,\,(x,\,y)))$ has a finite alphabet V of labels, consisting of disjoint subsets V_{M} of variables and V_{T} of terminals. R is the set of rules of the form $(A,\,(x,\,y)) \longrightarrow \{(A_1,\,(x_1,\,y_1)),\,(A_2,\,(x_2,\,y_2)),\ldots,\,(A_i,\,(x_i,\,y_i))\}$ where $A \in V_{\mathrm{M}}$, $\{A_1, A_2, \ldots, A_i\} \subseteq V$ and $(x,\,y),\,(x_1,\,y_1),\,\ldots,\,(x_i,\,y_i) \in \mathbb{Z}^2$ for $i \in \mathbb{N}_+$, $(S,(x,\,y))$ is the initial labelled shape, with $S \in V_{\mathrm{M}}$.

Definition 3.2. A pictorial form or evolving image is any set (composition) of labelled shapes in the plane denoted by Π. If Π is a pictorial form, we denote by $l(\Pi)$ the set of labels used in Π.

Definition 3.3. For an *ASG* G and pictorial forms Π and Γ, there is a derivation step from Π to Γ, if there is a rule $(s,\,(x,\,y)) \longrightarrow \{(s_1,\,(x_1,\,y_1)),(s_2,\,(x_2,\,y_2)),\ldots,(s_i,\,(xi,\,y_i))\}$ in R, where Π contains a labelled shape $(s,(x,\,y))$: $s \in V_M$ and Γ is $(\Pi \setminus \{(s,(x,\,y))\}) \cup \{(s_1,\,(x_1,\,y_1)),(s_2,\,(x_2,\,y_2)),\ldots,(s_i,\,(x_i,\,y_i))\}$: $\{s_1, s_2, \ldots, s_i\} \subseteq V$ for $i \in \mathbb{N}_+$ as usual.

We denote the derivation step by $\Pi \Longrightarrow \Gamma$. This simply means that Γ is directly derived from Π. If there is a sequence of zero or more derivation steps from Π to Γ, then we denote that by $\Pi \Longrightarrow^* \Gamma$. We now say that Γ is derived from Π.

Definition 3.4. An image is a pictorial form Π with $l(\Pi) \subseteq V_{\mathrm{T}}$.

Definition 3.5. The gallery $\mathcal{G}(G)$ generated by an additive shape grammar G is the set of images, Π, derivable from the initial shape $(S, (x, y))$, represented as: $\mathcal{G}(G) = \{\Pi : (S, (x, y)) \Longrightarrow^* \Pi\}$.

Next, we demonstrate how additive shape grammars work in Example 3.2.1.

Example 3.2.1. We want to generate the gallery of images that consist of the letter E (see Figure 3.2). Let $G_{\mathrm{E}} = (V_M, V_T, R, (S, (x, y)))$, where $V_M = \{S, A, B, C, D, E\}$, $V_T = \{ d \}$, $(S, (x, y)) = (S, (0, 0))$, and R is shown in Figure 3.3.

Some of the images in $\mathcal{G}(G_{\mathrm{E}})$ when rendered using square shapes with the label d associated with dark colour are shown in Figure 3.2. The images were scaled to the same size for presentation purposes as in Figure 3.2. The illustration of a derivation of G_{E} is given in Figure 3.4, having the lower lefthand corner of the initial shape S start at $x, y = 0$.

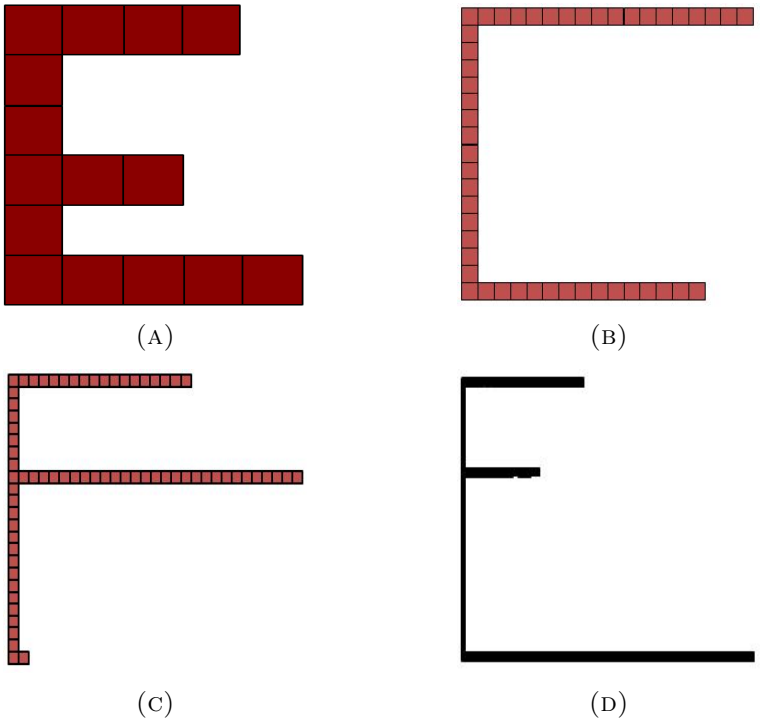


FIGURE 3.2: Some images in $\mathcal{G}(G_{\mathrm{E}})$.

The main purpose of additive shape grammars is to allow any shape to be used to render the images after derivation. The gallery in Figure 3.5 shows that different shapes can be used to render the same image when the derivation is complete. Some images obtained when the derivation in Figure 3.4 is rendered with different shapes are shown in Figure 3.5.

Observe that after Rule 3.2, any of the Rules 3.3–3.6 in Figure 3.3 are applicable and can be applied at any time. The same goes for Rules 3.7–3.10 after Rule 3.6, and Rules 3.11 and

$$
R = \Bigg\{
\begin{array}{lcll}
(S,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y)),(B,(x,y-1))\} & \text{(Rule 3.2)} \\
(A,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y))\} \mid & \text{(Rule 3.3)} \\
 & & \{(d,(x,y))\} & \text{(Rule 3.4)} \\
(B,(x,y)) & \longrightarrow & \{(d,(x,y)),(B,(x,y-1))\} \mid & \text{(Rule 3.5)} \\
 & & \{(d,(x,y)),(C,(x+1,y)),(D,(x,y-1))\} & \text{(Rule 3.6)} \\
(D,(x,y)) & \longrightarrow & \{(d,(x,y)),(D,(x,y-1))\} \mid & \text{(Rule 3.7)} \\
 & & \{(d,(x,y)),(E,(x+1,y))\} & \text{(Rule 3.8)} \\
(C,(x,y)) & \longrightarrow & \{(d,(x,y)),(C,(x+1,y))\} \mid & \text{(Rule 3.9)} \\
 & & \{(d,(x,y))\} & \text{(Rule 3.10)} \\
(E,(x,y)) & \longrightarrow & \{(d,(x,y)),(E,(x+1,y))\} \mid & \text{(Rule 3.11)} \\
 & & \{(d,(x,y))\} & \text{(Rule 3.12)}
\end{array}
\Bigg\}
$$

FIGURE 3.3: The set of rules for G_{E} in Example 3.2.1.

$$
\begin{aligned}
\{(S,(0,0))\} &\overset{3.2}{\Longrightarrow} \{(d,(0,0)),(A,(1,0)),(B,(0,-1))\} \\
&\overset{3.3^{(2)},3.4}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(B,(0,-1))\} \\
&\overset{3.5^{(2)},3.6}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(d,(0,-1)),(d,(0,-2)), \\
&\qquad (d,(0,-3)),(C,(1,-3)),(D,(0,-4))\} \\
&\overset{3.9,3.10}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(d,(0,-1)),(d,(0,-2)), \\
&\qquad (d,(0,-3)),(d,(1,-3)),(d,(2,-3)),(D,(0,-4))\} \\
&\overset{3.7,3.8}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(d,(0,-1)),(d,(0,-2)), \\
&\qquad (d,(0,-3)),(d,(1,-3)),(d,(2,-3)),(d,(0,-4)),(d,(0,-5)), \\
&\qquad (E,(1,-5))\} \\
&\overset{3.11^{(3)},3.12}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(d,(0,-1)),(d,(0,-2)), \\
&\qquad (d,(0,-3)),(d,(1,-3)),(d,(2,-3)),(d,(0,-4)),(d,(0,-5)), \\
&\qquad (d,(1,-5)),(d,(2,-5)),(d,(3,-5)),(d,(4,-5))\}
\end{aligned}
$$

FIGURE 3.4: A derivation of the picture in Figure 3.2(A) for G_{E} in Example 3.2.1.

3.12 after Rule 3.8. This can be seen as a limitation, as the images in the gallery may not only contain the letter E (see Figures 3.2(B) and (C)). This means that the additive shape grammars, in as much as they can allow for any shape to be used to render images after derivation (see Figure 3.5) which is one of the objectives of this study, may not always generate images in a regulated manner using the same grammar. As they do not consider the similarity of the images during a derivation. This simply means that additive shape grammars alone may not be able to generate images in a regulated manner at all times thus the need for the concept

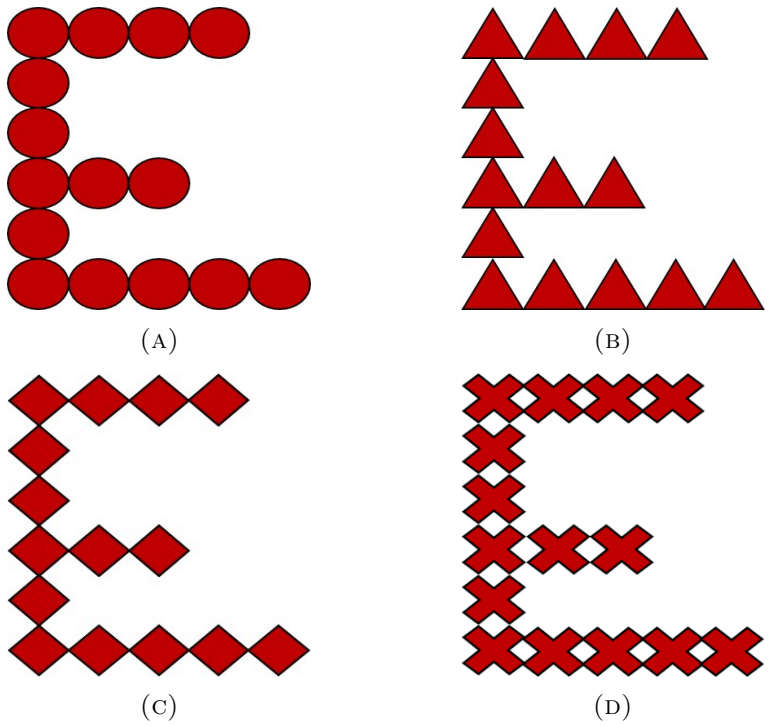


FIGURE 3.5: Image rendering of the derivation in Figure 3.4 with different shapes.

of bag context being added to the additive shape grammar rules to generate similar images of choice in a controlled manner. This control to the additive shape grammar rules and thus to the derivation process with the help of bag context will enable us to generate a gallery of images that will contain only images of $E's$ whose legs are equal length with the upper and lower spines the same length. This approach is what we called BCSGs in this work.

Next, we present BCSGs in Section 3.3.

3.3 Bag Context Shape Grammars

Informally, a bag context shape grammar may be defined as a shape grammar G in which the application of a rule is regulated by a special vector of integers called the bag β. For a rule r to be applied by G at a particular point in a derivation, the bag β must be within the range, determined by the lower limit and upper limit L_{b} and $U_{\mathrm{b}} \in \mathbb{Z}_{\infty}^{I}$, which are defined in r. The bag adjustment δ is made together with the application of the rule, that is, δ is also defined in r.

Next, we give the formal definition of bag context shape grammars and an example in Section 3.3.1.

3.3.1 Formal Definition of Bag Context Shape Grammars

This section is motivated by the definition of bag context tree grammars in [35, 43] and the definition of additive shape grammars in Section 3.2.2 of this thesis.

Definition 3.6. A BCSG is a grammar with the form $G = (V_{\mathrm{M}}, V_{\mathrm{T}}, R, (S, (x, y)), I, \beta_0)$, where V_{M}, V_{T}, and $(S, (x, y))$ are as in Definition 3.1, R is the set of shape grammar rules, where every rule is of the form $(A, (x, y)) \longrightarrow \{(A_1, (x_1, y_1)), (A_2, (x_2, y_2)), \ldots, (A_i, (x_i, y_i))\}$ $(L_{\mathrm{b}}, U_{\mathrm{b}} ; \delta)$, where $(A, (x, y)) \longrightarrow \{(A_1, (x_1, y_1)), (A_2, (x_2, y_2)), \ldots, (A_i, (x_i, y_i))\}$ is as in Definition 3.1, $L_{\mathrm{b}}, U_{\mathrm{b}} \in \mathbb{Z}^I_\infty$, and δ is the bag adjustment, $\delta \in \mathbb{Z}^I$. I is the finite bag index set of the form $[k]$ and β_0 is the initial bag, $\beta_0 \in \mathbb{Z}^I$.

Definition 3.7. Let a *configuration* be a pair (Π, β) where Π is pictorial form and β is the bag. For a BCSG G and *configurations* (Π, β) and (Γ, β'), there is a derivation step from (Π, β) to (Γ, β'), if there is a rule $(s, (x, y)) \longrightarrow \{(s_1, (x_1, y_1)), (s_2, (x_2, y_2)), \ldots, (s_i, (x_i, y_i))\}$ $(L_{\mathrm{b}}, U_{\mathrm{b}} ; \delta)$ in R, where Π contains a labelled shape $(s, (x, y))$: $s \in V_M$ with $L_{\mathrm{b}} \leq \beta \leq U_{\mathrm{b}}$. $\Gamma = (\Pi \setminus \{(s, (x, y))\}) \cup \{(s_1, (x_1, y_1)), (s_2, (x_2, y_2)), \ldots, (s_i, (x_i, y_i))\}$ and $\beta' = \beta + \delta$.

We denote the derivation step by $(\Pi, \beta) \Longrightarrow (\Gamma, \beta')$. This simply means that (Γ, β') is directly derived from (Π, β). If there is a sequence of zero or more derivation steps from (Π, β) to (Γ, β'), then we denote that by $(\Pi, \beta) \Longrightarrow^* (\Gamma, \beta')$. We now say that (Γ, β') is derived from (Π, β).

Definition 3.8. An image is a pictorial form Π with $l(\Pi) \subseteq V_T$ and the bag, β.

Definition 3.9. The gallery $\mathcal{G}(G)$ generated by a BCSG G is the set of images, Π, derivable from the initial labelled shape $(S, (x, y))$ and the initial bag β_0, represented as: $\mathcal{G}(G) = \{\Pi : ((S, (x, y)), \beta_0) \Longrightarrow^* (\Pi, \beta)$, for some $\beta \in \mathbb{Z}^I_\infty$ *and* $l(\Pi) \subseteq V_T\}$.

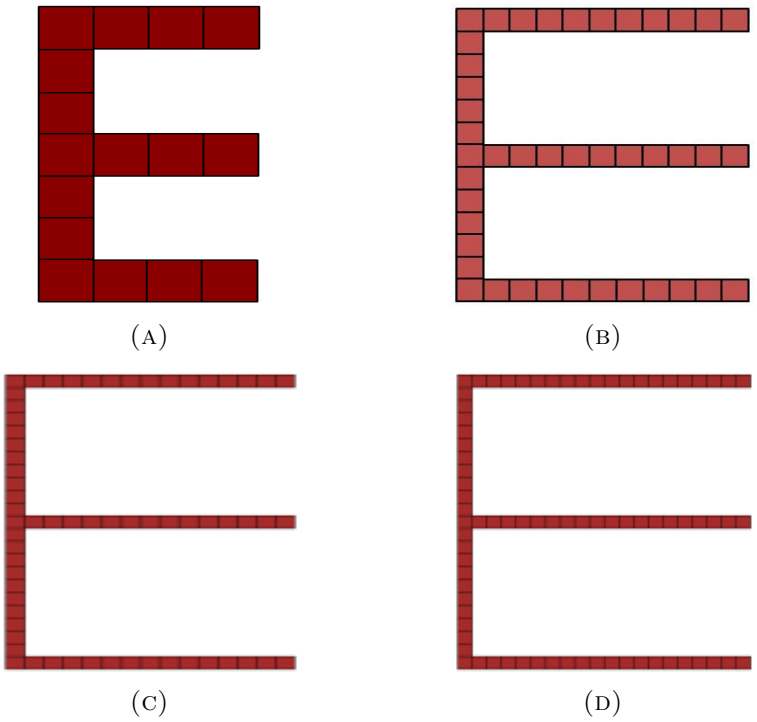
Next, we demonstrate BCSGs in Example 3.3.1.

3.3.2 Illustration of BCSGs

In Example 3.3.1, we generate images with the following: three legs of equal length and a spine. The second leg of the letter E to be generated divides the spine into an upper and a lower half which are the same length.

Example 3.3.1. We extend Example 3.2.1 by adding bag context to generate the images in Figure 3.6. Let $G_{\mathrm{E-bag}} = (V_M, V_T, R, (S, (x, y)), \{1, 2, 3, 4\}, 0)$, where $V_M = \{S, A, B, C, D, E\}$, $V_T = \{ d \}$, $(S, (x, y)) = (S, (0, 0))$, and R is shown in Figure 3.7.

Some of the images in the gallery produced by $G_{\mathrm{E-bag}}$ when rendered using square shapes with the label d associated with dark colour are shown in Figure 3.6. The images were scaled to the size as in Figure 3.6 for presentation purposes.



(A) (B)

(C) (D)

FIGURE 3.6: Some images in $\mathcal{G}(G_{\text{E-bag}})$

The strategy here is that the first, third, and fourth bag positions are used to control how the first, second and third legs grow. The second bag position is used to control the upper and lower spines of the letter *E*. The first (top) leg of the letter *E* is formed before any other part. The reason for this is that the number of times the rule that generates the top leg is applied is used to control the other legs. The upper spine is formed immediately after the top leg. The second leg is formed after the upper spine formation. Then the lower spine is formed followed by the third leg formation. The second leg must be exactly at half the spine which makes the upper and lower spines to be the same length. The three legs must also be the same length.

Next, we explain how the rules in Figure 3.7 are applied to form the image in Figure 3.6(A).

To achieve the strategy, the derivation starts with the initial labelled shape in the pictorial form, $\Pi = \{(S, (x, y))\}$. The application of Rule 3.13 of Figure 3.7 replaces the initial labelled shape $(S, (x, y))$ with the shapes labelled $(d, (x, y))$, $(A, (x + 1, y))$ and $(B, (x, y - 1))$ in the set. The bag adjustment, (1, 1, 0, 0) is added to the bag, (0, 0, 0, 0) which is (0 + 1, 0 + 1, 0 + 0, 0 + 0) then the bag becomes (1, 1, 0, 0).

We can apply Rule 3.14 twice because the bag at each application is within the defined range. That is the bag is greater than or equal to the lower limit, (1, 1, 0, 0) and less than or equal to the upper limit, ∞. The bag adjustment, (1, 0, 0, 0) is added to the bag at each application of Rule 3.14. This is followed by the application of Rule 3.15 because it is also within the defined range.

At this point, the bag is (2, 0, 0, 1). We apply Rule 3.16 twice to form the upper spine; the bag is (2, 2, 0, 1). Rule 3.17 is then applied to begin the formation of the second leg and the lower

$$
R = \Bigg\{
$$

$$
\begin{array}{lcll}
(S,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y)),(B,(x,y-1))\}(0,0;(1,1,0,0)) & \text{(Rule 3.13)} \\
(A,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y))\}((1,1,0,0),\infty;(1,0,0,0)) \mid & \text{(Rule 3.14)} \\
 & & \{(d,(x,y))\}((2,1,0,0),(\infty,\infty,\infty,1);(-1,-1,0,1)) & \text{(Rule 3.15)} \\
(B,(x,y)) & \longrightarrow & \{(d,(x,y)),(B,(x,y-1))\}((1,0,0,1),(\infty,\infty,\infty,1);(0,1,0,0)) \mid & \text{(Rule 3.16)} \\
 & & \{(d,(x,y)),(C,(x+1,y)),(D,(x,y-1))\} & \\
 & & \quad((1,2,0,1),(\infty,\infty,0,1);(0,0,0,1)) & \text{(Rule 3.17)} \\
(D,(x,y)) & \longrightarrow & \{(d,(x,y)),(D,(x,y-1))\}((0,1,1,3),(0,\infty,\infty,3);(0,-1,0,0)) \mid & \text{(Rule 3.18)} \\
 & & \{(d,(x,y)),(E,(x+1,y))\}((0,0,1,3),(\infty,0,\infty,3);(0,0,0,1)) & \text{(Rule 3.19)} \\
(C,(x,y)) & \longrightarrow & \{(d,(x,y)),(C,(x+1,y))\}((1,1,0,2),(\infty,\infty,\infty,2);(-1,0,1,0)) \mid & \text{(Rule 3.20)} \\
 & & \{(d,(x,y))\}((0,1,2,2),(0,\infty,\infty,2);(0,0,0,1)) & \text{(Rule 3.21)} \\
(E,(x,y)) & \longrightarrow & \{(d,(x,y)),(E,(x+1,y))\}((0,0,1,4),(\infty,\infty,\infty,4);(1,0,-1,0)) \mid & \text{(Rule 3.22)} \\
 & & \{(d,(x,y))\}((1,0,0,4),(\infty,\infty,0,4);0) & \text{(Rule 3.23)}
\end{array}
$$

$$
\Bigg\}
$$

FIGURE 3.7: The set of rules for $G_{\text{E}-\text{bag}}$ in Example 3.3.1.

spine; the bag adjustment, (0, 0, 0, 1) is added to the bag, (2, 2, 0, 1) then the bag becomes (2, 2, 0, 2).

We further apply Rule 3.20 twice and Rule 3.21 once to form the second leg. At this point, the bag adjustment, (0, 0, 0, 1) is added to the bag, (0, 2, 2, 2) the bag becomes (0, 2, 2, 3). We apply Rule 3.18 twice followed by Rule 3.19 because the bag at each application is within the defined range. At this point, the bag adjustment, (0, 0, 0, 1), is added to the bag, (0, 0, 2, 3) then the bag becomes (0, 0, 2, 4) and the lower spine is formed.

Finally, we apply Rule 3.22 twice and Rule 3.23 once to form the third leg. The bag adjustment, (0, 0, 0, 0) is added to the bag, (2, 0, 0, 4) the bag becomes (2, 0, 0, 4). At this point the picture formation is completed and the letter E is generated (see Figure 3.6(A)).

We can also generate more images by repeating the process and applying Rules 3.14 and 3.16 as long as they are within the defined range at each application. This is because the number of times Rule 3.14 is applied determines the number of times Rules 3.20 and 3.22 will be applied and the number of times Rule 3.16 is applied determines the number of times Rule 3.18 will be applied with the help of the bag. Then, the process can be terminated with Rules 3.15, 3.21 and 3.23 respectively in order to form a complete picture (see Figures 3.6(B)–(D)).

An illustration of one of the derivations of $G_{\text{E}-\text{bag}}$ using the picture in Figure 3.6(A) is given in Figure 3.8. The derivation has the lower lefthand corner of the initial shape S start at x, $y = 0$. The derivation is summarised in Table 3.1 showing the operations in the bag when an adjustment is made.

Some images obtained when the derivation in Figure 3.8 is rendered with different shapes are shown in Figure 3.9.

$$
\begin{aligned}
\{(S,(0,0))\} &\overset{3.13}{\Longrightarrow} \{(d,(0,0)),(A,(1,0)),(B,(0,-1))\}\\
&\overset{3.14^{(2)},3.15}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(B,(0,-1))\}\\
&\overset{3.16^{(2)},3.17}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(d,(0,-1)),(d,(0,-2)),\\
&\quad (d,(0,-3)),(C,(1,-3)),(D,(0,-4))\}\\
&\overset{3.20^{(2)},3.21}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(d,(0,-1)),(d,(0,-2)),\\
&\quad (d,(0,-3)),(d,(1,-3)),(d,(2,-3)),(d,(3,-3)),(D,(0,-4))\}\\
&\overset{3.18^{(2)},3.19}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(d,(0,-1)),(d,(0,-2)),\\
&\quad (d,(0,-3)),(d,(1,-3)),(d,(2,-3)),(d,(3,-3)),(d,(0,-4)),\\
&\quad (d,(0,-5)),(d,(0,-6)),(E,(1,-6))\}\\
&\overset{3.22^{(2)},3.23}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(d,(0,-1)),(d,(0,-2)),\\
&\quad (d,(0,-3)),(d,(1,-3)),(d,(2,-3)),(d,(3,-3)),(d,(0,-4)),\\
&\quad (d,(0,-5)),(d,(0,-6)),(d,(1,-6)),(d,(2,-6)),\\
&\quad (d,(3,-6))\}
\end{aligned}
$$

FIGURE 3.8: A derivation of the picture in Figure 3.6(A) for $G_{\mathrm{E-bag}}$.

TABLE 3.1: The bag during the derivation in Figure 3.8.

Rules	β	δ
3.13	0	(1, 1, 0, 0)
$3.14^{(2)}$	(1, 1, 0, 0)	(1, 0, 0, 0)
3.15	(3, 1, 0, 0)	(-1, -1, 0, 1)
$3.16^{(2)}$	(2, 0, 0, 1)	(0, 1, 0, 0)
3.17	(2, 2, 0, 1)	(0, 0, 0, 1)
$3.20^{(2)}$	(2, 2, 0, 2)	(-1, 0, 1, 0)
3.21	(0, 2, 2, 2)	(0, 0, 0, 1)
$3.18^{(2)}$	(0, 2, 2, 3)	(0, -1, 0, 0)
3.19	(0, 0, 2, 3)	(0, 0, 0, 1)
$3.22^{(2)}$	(0, 0, 2, 4)	(1, 0, -1, 0)
3.23	(2, 0, 0, 4)	0

Next, we compare bag context shape grammars and basic puzzle grammars with permitting features in Section 3.4.

3.4 The Power of BCSGs

In this section, we compare BCSGs to basic puzzle grammars with permitting features (also called permitting basic puzzle grammar). By permitting feature we mean that the grammar can only enable the application of a rule at any step during derivation if the permitting set of symbol(s) of that rule appear(s) in the pictorial form. This is also similar to our type of BCSGs

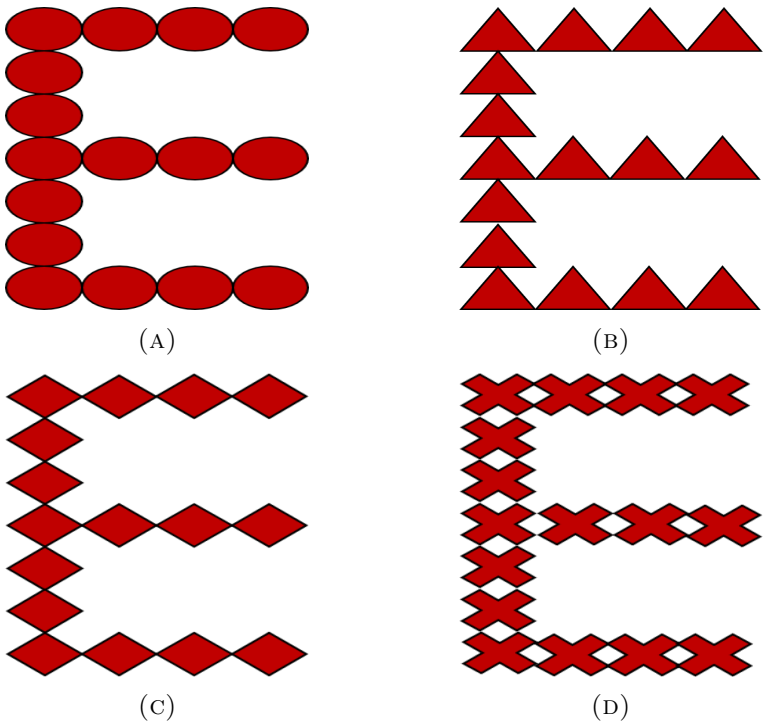


FIGURE 3.9: Image rendering of the derivation in Figure 3.8 with different shapes.

that uses the bag to control the application of a rule during derivation. The choice of permitting basic puzzle grammars is because they are one of the recent developments in puzzle grammar systems.

3.4.1 Permitting Basic Puzzle Grammars

We begin by understanding permitting basic puzzle grammar.

Definition 3.10. A permitting basic puzzle grammar (PBPzG) $G = (V_M, V_T, R, S)$ has disjoint, nonempty, finite sets of V_M and V_T of variables and terminals, respectively, a finite set of rules R, and a start symbol $S \in V_M$ as usual.

A rule is of the form $(A \to \alpha, per)$, where $A \to \alpha$ is the basic puzzle grammar rule as in Definition 2.16, $A \in V_M$, $\alpha \subseteq (V_M \cup V_T)$ and $per \subseteq V_M$ [71]. If $per = \emptyset$, then it is not mentioned in the rule. A derivation starts with S written in a unit cell in the two-dimensional plane, with all other cells containing the blank symbol $\# \notin V_M \cup V_T$.

For a $PBPzG$ G and a pictorial form Π, a rule, $A \to \alpha$, per in R can be applied if $A \in V_M$ and $per \subseteq l(\Pi) \setminus \{A\}$. Then, in a derivation step, denoted by $\Longrightarrow^*_{\Pi}$, a nonterminal A in a cell is replaced by the right hand side of a rule with lefthand side A. In this replacement, the square symbol of the right hand side of the rule used occupies the cell of the replaced symbol A and the non-square symbol of the right hand side occupies the cell to the right or left or above or below the cell of the replaced symbol, depending on the type of rule used. The replacement is possible only if the cell to be filled in by the non-square symbol contains a #.

An *image* generated by G is a connected, finite array of elements of V_T derived in one or more steps from the start symbol; the set of such images, the *gallery*, is denoted by $\mathcal{G}(G)$.

Example 3.4.1. Consider a PBPzG of the structure $G = (V_M, V_T, R, S)$ where $V_M = \{S, A, B, A', B', C, D\}$, $V_T = \{d\}$, R is the set of rules in Figure 3.10 and S is an array of the form

A
d B .

$R =$ {

$$S \longrightarrow \begin{array}{l}\text{A}\\ \boxed{\text{d}}\ \text{B}\end{array} \quad \text{(Rule 3.24)}$$

$$A \longrightarrow \begin{array}{l}A'\\ \boxed{\text{d}}\end{array}, \{\text{B}\} \quad \text{(Rule 3.25)}$$

$$B \longrightarrow \boxed{\text{d}}\ B', \quad \{A'\} \quad \text{(Rule 3.26)}$$

$$A' \longrightarrow A, \{B'\} \quad \text{(Rule 3.27)}$$

$$B' \longrightarrow B, \{A\} \quad \text{(Rule 3.28)}$$

$$A \longrightarrow C, \{B\} \quad \text{(Rule 3.29)}$$

$$B \longrightarrow D, \{C\} \quad \text{(Rule 3.30)}$$

$$C \longrightarrow d \quad \text{(Rule 3.31)}$$

$$D \longrightarrow d \quad \text{(Rule 3.32)}$$

}

FIGURE 3.10: The set of PBPzG rules in Example 3.4.1.

The derivation starts with S then Rule 3.25 is applicable because the permitting symbol B is present in the array. This grows the vertical arm by one square. At this point, Rule 3.26 can be applied because the permitting symbol, A' is present, hence this grows the horizontal arm by one square. Rule 3.27 can now be applied because the permitting symbol, B' is present followed by Rule 3.28 whose permitting symbol, A is present. The process can be repeated to grow both arms equal in length until Rules 3.29 and 3.30 are applied to change the symbol A to C and B to D. Then the process can be terminated by the application of Rules 3.31 and 3.32. This derivation generates a gallery in the shape of an L with base and height equal in length. The gallery generated by the grammar is presented in Figure 3.11. Next, we compare BCSGs and PBPzG.

3.4.2 Comparison of BCSGs and PBPzG

We consider a permitting basic puzzle grammar, PBPzG, $G = (V_M, V_T, R, S)$. Suppose $V_M = \{A_1, A_2, \ldots, A_m\}$, and the order of the elements in V_M is arbitrary but fixed. For rule $A \to \alpha$, *per*, we denote by *cnt*, the occurrences of the variable of V_M in the lefthand and righthand sides of the rule. In particular,

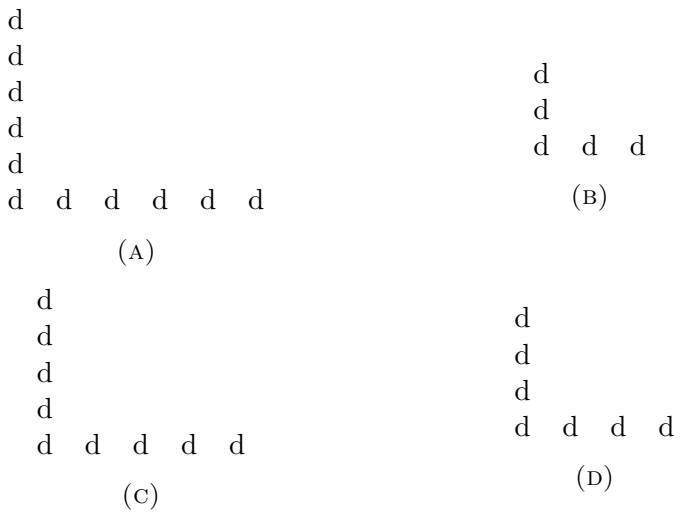


FIGURE 3.11: Some images in $\mathcal{G}(G)$.

- $cnt(A)$ denotes the m–tuple in which the component corresponding to A is set to 1, whereas all other components are set to 0,
- $cnt(\alpha)$ denotes the m–tuple $(n_1, n_2, \ldots, n_m)$ such that n_i is equal to the number of occurrences of A_i in $[x_1, \ldots, x_m]$, and
- $cnt(per)$, denotes the $\sum_{A \in per} cnt(A)$.

The following lemma is adapted from [43].

Lemma 3.11. *For every basic permitting puzzle grammar, $G = (V_M, V_T, R, S)$ there is an equivalent BCSG $G' = (V_{\mathrm{M}}, V_{\mathrm{T}}, R', (S, (x, y)), I, \beta_0)$. That is, $\mathcal{G}(G) \subseteq \mathcal{G}(G')$.*

Proof: Without loss of generality, assume that $V_M = \{A_1, A_2, \ldots, A_m\}$, $\forall\ m \in \mathbb{N}_+$. The bag index set $I = [m]$ records, at all times during a derivation, the number of occurrences of A_i in the ith position of the bag. Let the initial bag, $\beta_0 = cnt(S)$. For every permitting puzzle grammar rule, $A \to \alpha$, *per* in R, we write it to the equivalent bag context shape grammar rule $(A, (x, y)) \longrightarrow \{(A_1, (x_1, y_1)), (A_2, (x_2, y_2)), \ldots, (A_i, (x_i, y_i))\}$ $(L_{\mathrm{b}}, U_{\mathrm{b}}\ ;\ \delta)$ in R', where

- $L_{\mathrm{b}} = cnt(A) + cnt(per)$,
- $U_{\mathrm{b}} = cnt(A) + \infty.cnt(V_M) = cnt(A) + \infty.cnt(V_M) = cnt(A) + \infty = \infty$, and
- $\delta = cnt(\alpha)$ - $cnt(A)$.

Then, $\mathcal{G}(G) \subseteq \mathcal{G}(G')$.

Next, we use Lemma 3.11 to illustration that the rules of Example 3.4.1 of PBPzG can be converted to BCSG rules. We derive the values of the lower limit, L_{b}, the upper limit, U_{b}, and the bag adjustment, δ for each rule in Figure 3.10 see Table 3.2. Then the values obtained

in Table 3.2 are used to rewrite each rule in Figure 3.10 to a BCSG rule, see Figure 3.12 of Example 3.4.2.

TABLE 3.2: The conversion of the PBPzG rules in Figure 3.10 to bag context.

Rules	L_b	U_b	δ
3.24	(1, 0, 0, 0, 0, 0, 0)	∞	(-1, 1, 1, 0, 0, 0, 0)
3.25	(0, 1, 1, 0, 0, 0, 0)	∞	(0, -1, 0, 1, 0, 0, 0)
3.26	(0, 0, 1, 1, 0, 0, 0)	∞	(0, 0, -1, 0, 1, 0, 0)
3.27	(0, 0, 0, 1, 1, 0, 0)	∞	(0, 1, 0, -1, 0, 0, 0)
3.28	(0, 1, 0, 0, 1, 0, 0)	∞	(0, 0, 1, 0, -1, 0, 0)
3.29	(0, 1, 1, 0, 0, 0, 0)	∞	(0, -1, 0, 0, 0, 1, 0)
3.30	(0, 0, 1, 0, 0, 1, 0)	∞	(0, 0, -1, 0, 0, 0, 1)
3.31	(0, 0, 0, 0, 0, 1, 0)	∞	(0, 0, 0, 0, 0, -1, 0)
3.32	(0, 0, 0, 0, 0, 0, 1)	∞	(0, 0, 0, 0, 0, 0, -1)

Example 3.4.2. We want to rewrite the rules of Example 3.4.1 to BCSG rules using the values of L_b, U_b, and δ in Table 3.2 for each rule. Let a BCSG $G_\mathrm{bag} = (V_\mathrm{M}, V_\mathrm{T}, R', (S, (x, y)), \{1, 2, \ldots, 7\}, \{1, 0, 0, 0, 0, 0, 0\})$, where $V_\mathrm{M} = \{S, A, B, A', B', C, D\}$, $V_\mathrm{T} = \{d\}$, R' is as shown in Figure 3.12, and $(S, (x, y)) = ((d, (0, 0)), (A, (0, 1)), (B, (1, 0)))$.

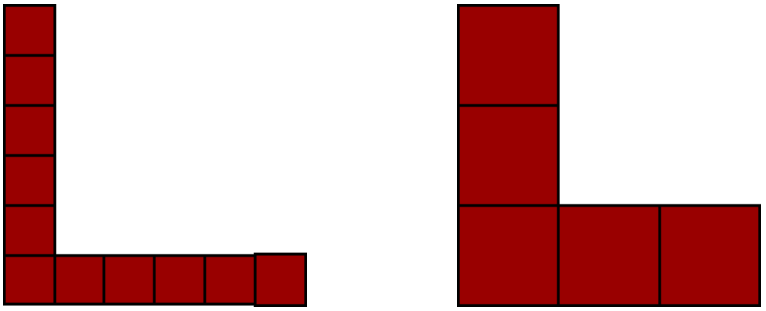
$$
R' = \left\{
\begin{aligned}
(S,(x,y)) &\longrightarrow \{(d,(x,y)),(A,(x,y+1)),(B,(x+1,y))\} \\
&\qquad ((1,0,0,0,0,0,0),\infty;(-1,1,1,0,0,0,0)) && \text{(Rule 3.33)}\\
(A,(x,y)) &\longrightarrow \{(d,(x,y)),(A',(x,y+1))\} \\
&\qquad ((0,1,1,0,0,0,0),\infty;(0,-1,0,1,0,0,0)) && \text{(Rule 3.34)}\\
(B,(x,y)) &\longrightarrow \{(d,(x,y)),(B',(x+1,y))\} \\
&\qquad ((0,0,1,1,0,0,0),\infty;(0,0,-1,0,1,0,0)) && \text{(Rule 3.35)}\\
(A',(x,y)) &\longrightarrow \{(A,(x,y))\}((0,0,0,1,1,0,0),\infty;(0,1,0,-1,0,0,0)) && \text{(Rule 3.36)}\\
(B',(x,y)) &\longrightarrow \{(B,(x,y))\}((0,1,0,0,1,0,0),\infty;(0,0,1,0,-1,0,0)) && \text{(Rule 3.37)}\\
(A,(x,y)) &\longrightarrow \{(C,(x,y))\}((0,1,1,0,0,0,0),\infty;(0,-1,0,0,0,1,0)) && \text{(Rule 3.38)}\\
(B,(x,y)) &\longrightarrow \{(D,(x,y))\}((0,0,1,0,0,1,0),\infty;(0,0,-1,0,0,0,1)) && \text{(Rule 3.39)}\\
(C,(x,y)) &\longrightarrow \{(d,(x,y))\}((0,0,0,0,0,1,0),\infty;(0,0,0,0,0,-1,0)) && \text{(Rule 3.40)}\\
(D,(x,y)) &\longrightarrow \{(d,(x,y))\}((0,0,0,0,0,0,1),\infty;(0,0,0,0,0,0,-1)) && \text{(Rule 3.41)}
\end{aligned}
\right\}
$$

FIGURE 3.12: The set of rules for G_bag in Example 3.4.2.

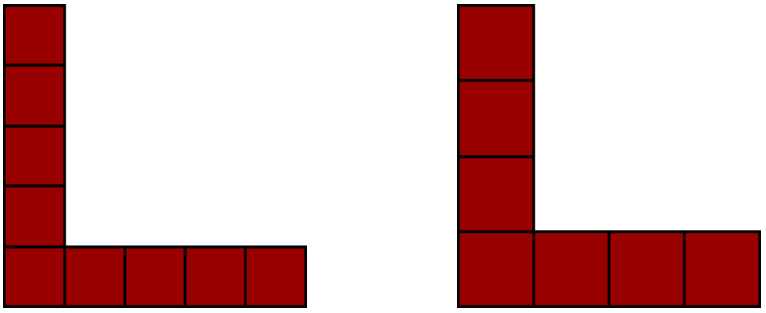
The derivation starts with S, then Rule 3.34 is applicable because the bag is within the defined range. That is the bag, (0, 1, 1, 0, 0, 0, 0) is greater than or equal to the lower limit, (0, 1, 1, 0, 0, 0, 0) and the bag is at the same time less than or equal to the upper limit, ∞, then the bag adjustment, (0, -1, 0, 1, 0, 0, 0) is added to the bag. Rule 3.35 is applicable followed by Rules 3.36 and 3.37 because they are within the range at each application. This process can be repeated to grow both arms equal in length until Rule 3.38 is applied followed by Rule 3.39.

Then the process can be terminated by applying Rules 3.40 and 3.41 because they are within the defined range.

This derivation generates images in the shape of an L with base and height equal in length. The images are presented in Figure 3.13 when the terminal d is replaced with a square shape which is filled with a dark colour. The images are also scaled to the same size for presentation purposes.



(A) (B)



(C) (D)

FIGURE 3.13: Some images in $\mathcal{G}(G_{\text{bag}})$.

Clearly the gallery $\mathcal{G}(G) \subseteq \mathcal{G}(G_{\text{bag}})$, see Figures 3.11 and 3.13. Next, we show that BCSGs can generate the same gallery with fewer rules and variables.

Example 3.4.3. Consider the gallery of an L shape whose base and height are equal in length. Let BCSG $G'_{\text{bag}} = (V_{\text{M}}, V_{\text{T}}, R'', (S, (x, y)), \{1, 2, 3\}, 0)$, where $V_{\text{M}} = \{S, A, B\}$, $V_{\text{T}} = \{d\}$, R''' is as shown in Figure 3.14, and $(S, (x, y)) = (S, (0, 0))$.

The strategy is that the first and the second bag positions are used to control how the base and the height of the L shape grow. The third bag position ensures that the base and height of the L are the same lengths.

To achieve the strategy, Rule 3.43 is applied immediately after Rule 3.42, then the bag adjustment, (0, 0, 1) is added to the bag. Rule 3.43 is no longer applicable because the third bag position is now 1 until Rule 3.45 is applied, which automatically adds the bag adjustment (0, 0, -1) to the bag. The third bag position is 0 making Rule 3.45 not to be applicable. This process can continue while the base and height grow at equal length until Rule 3.44 is applied followed by Rule 3.46. Hence, $\mathcal{G}(G) \subseteq \mathcal{G}(G'_{\text{bag}})$.

Observe that there are seven variables in Examples 3.4.1 while Example 3.4.3 has only three variables. Also, the number of rules in Figure 3.10 of Examples 3.4.1 has nine number of rules

$$
R'' = \left\{
\begin{array}{lcll}
(S,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x,y+1)),(B,(x+1,y))\}(0,0;(1,1,0)) & \text{(Rule 3.42)} \\
(A,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x,y+1))\}((1,1,0),(1,1,0);(0,0,1)) \mid & \text{(Rule 3.43)} \\
 & & \{(d,(x,y))\}((1,1,0),(1,1,0);(-1,0,1)) & \text{(Rule 3.44)} \\
(B,(x,y)) & \longrightarrow & \{(d,(x,y)),(B,(x+1,y))\}(1,1;(0,0,-1)) \mid & \text{(Rule 3.45)} \\
 & & \{(d,(x,y))\}((0,1,1),(0,1,1);(0,-1,-1)) & \text{(Rule 3.46)}
\end{array}
\right\}
$$

FIGURE 3.14: The set of rules for G'_{bag} in Example 3.4.3.

while the number of rules in Figure 3.14 is only five rules. Next, we present some limitation of the BCSGs in Section 3.5.

3.5 The Limitations of BCSGs

When a PBPzG is converted to a BCSG using the above Lemma 3.11, the resultant BCSG has a bag position for each variable of the PBPzG. By inspecting the BCSG, one can often reduce the number of bag positions. The question is whether there is a general rules for reducing bag positions.

Next, we summarise this chapter in Section 3.6.

3.6 Summary

In this chapter, we presented the theories and the concept of additive shape grammar systems. The additive shape grammars allow for any shape to be used to render images after derivation. Like other shape grammar systems, the example used to demonstrate the additive shape grammars showed that they are not capable of controlling the application of rules during a derivation. Then, the notion and method of bag context being added to the additive shape grammar rules to control the application of rules during derivation was introduced. This approach is called BCSGs. A demonstration of the BCSGs with an example showed that BCSGs can generate images in a regulated manner by controlling the application of the rules and the derivation process. Furthermore, we proved that every PBPzG rule can be converted to a BCSG rule and demonstrated the conversion process with examples. We then considered a set of images and demonstrated that bag context shape grammars can generate a set of images with fewer variables, rules, and even fewer bag positions.

Other developments in additive grammar models with permitting features such as cooperating context free array grammar systems [151], permitting features in P systems generating picture arrays [152], array P systems with permitting features [153], cooperating basic puzzle grammars

systems [154], etc remain open for further research with regard to BCSGs. We are not be able to cover all these developments in additive grammar models with permitting features in one thesis given the time frame. In the next chapter, we demonstrated the power of BCSGs with regards to similar image generation.

Chapter 4

More Examples of BCSGs

4.1 Introduction

In this chapter, we show the strength of BCSGs by demonstrating different ways we can use BCSGs to generate similar images. The strength of BCSGs in this section is not to say that other shape grammars or the additive shape grammars cannot generate images that BCSGs can generate. BCSGs are also shape grammars but with controlled rules.

However, most shape grammar systems or the additive shape grammar fail to generate only similar images in a controlled manner, (see Figures 2.27, 2.30, and 3.2). For a shape grammar or the additive shape grammar to generate similar images, it requires users to generate different images and manually select the images that are similar. It is difficult to know if the next image to be generated will be similar to a selected image. The existing shape grammars, even the additive shape grammars lack the ability to regulate how similar images can be generated in controlled manner.

Therefore, the strength of BCSGs in this context is to demonstrate different methods these BCSGs can generate an infinite number of similar images only, using the same grammar. Images are considered similar when they are generated with the same grammar and they share some resemblance in appearance, such as, colour or shape without being identical. For instance, images that contain subimages or images that are generated with a limited bound (lower are upper limits).

Next, we start by generating carpets.

4.2 Generation of Carpet with Variations

The focus here is the generation of carpets with a variation. Variation in this part is the generation of images with a slight difference either in the structure or in the colour placement. These variations in the image generation are controlled using the bag context during derivation.

The labels d and w are replaced with circle shape during rendering and they are associated with dark and light colours respectively. Then the images are scaled to the same size for presentation purposes.

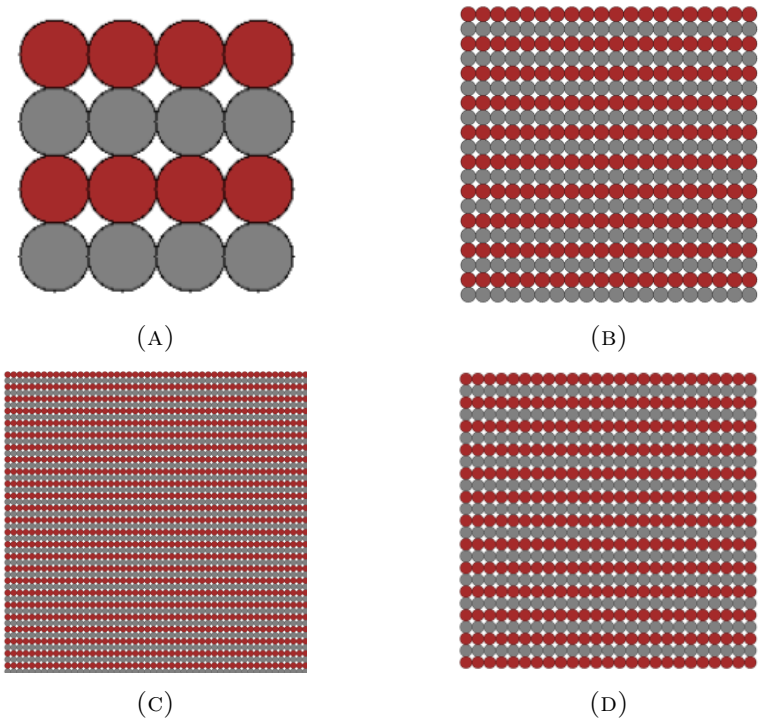
4.2.1 Generation of Carpet with Colour Variation

In Example 4.2.1 we consider a gallery of images that consists of carpets where each colour fills or dominates an entire row, see Figure 4.1.

Example 4.2.1. Let $G_{\text{Carpet}-\text{A}} = (V_M, V_T, R, S, \{1, 2, 3\}, 0)$, where $V_M = \{S, A, B, C\}$, $V_T = \{d, w\}$, $(S, (x, y)) = (S, (0, 0))$, and R is the set in Figure 4.2.

The first and second bag positions are used to control the variables $A - C$, as such, they ensure a colour dominates a row at a time. The third bag position is used to ensure the number of rows are exactly the same as the number of columns.

Some images in $\mathcal{G}$ produced by $G_{\text{Carpet}-\text{A}}$ are shown in Figure 4.1.



(A) (B) (C) (D)

FIGURE 4.1: Some of the images in $\mathcal{G}$ of $G_{\text{Carpet}-\text{A}}$.

The strategy is that the number of times Rule 4.2 of Figure 4.2 is applied determines the size of the image. Then only one colour is allowed to dominate a row at a time.

To achieve the strategy, the application of Rule 4.1 of Figure 4.2 starts the derivation. Then Rule 4.2 is applicable as long as the bag at each application is within the defined range. That is the bag is greater than or equal to the lower limit, (1, 0, 1) and less than or equal to the upper limit, (∞, 0, ∞). The bag adjustment, (1, 0, 1) is added to the bag which automatically

$$
R = \left\{
\begin{array}{lll}
(S,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y))\}(0,0;(1,0,1)) \quad \text{(Rule 4.1)} \\
(A,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y))\}((1,0,1),(\infty,0,\infty);(1,0,1)) \mid \quad \text{(Rule 4.2)} \\
 & & \{(d,(x,y)),(B,(x,y-1))\}((2,0,2),(\infty,0,\infty);(0,0,-1)) \quad \text{(Rule 4.3)} \\
(B,(x,y)) & \longrightarrow & \{(B,(x-1,y)),(w,(x,y))\}((1,0,0),(\infty,\infty,\infty);(-1,1,0)) \mid \text{(Rule 4.4)} \\
 & & \{(w,(x,y)),(C,(x,y-1))\}((0,1,1),(0,\infty,\infty);(0,0,-1)) \mid \quad \text{(Rule 4.5)} \\
 & & \{(w,(x,y))\}((0,(0,\infty,0);0) \quad \text{(Rule 4.6)} \\
(C,(x,y)) & \longrightarrow & \{(d,(x,y)),(C,(x+1,y))\}((0,1,1),(\infty,\infty,\infty);(1,-1,0)) \mid \text{(Rule 4.7)} \\
 & & \{(d,(x,y)),(B,(x,y-1))\}((1,0,1),(\infty,\infty,\infty);(0,0,-1)) \mid \text{(Rule 4.8)} \\
 & & \{(d,(x,y))\}((0,0,0),(\infty,0,0);0) \quad \text{(Rule 4.9)}
\end{array}
\right\}
$$

FIGURE 4.2: The set of rules for $G_{\text{Carpet}-\text{A}}$.

increments the first and third bag positions by 1 at each application. This is to control the variables, while the number of rows and columns are kept in check.

The application of Rule 4.3 adds the bag adjustment, (0, 0, -1) to the bag which automatically decrements the third bag position by 1. This also begins the formation of a new row. Then Rules 4.4 and 4.5 followed by Rules 4.7 and 4.8 are applicable as long as the bag at each application is within the range. This process continues until the third bag position is 0 then, any of Rule 4.6 or 4.9 of Figure 4.2 is applicable to complete the formation of the image. This strategy is sufficient to generate the images in Figure 4.1.

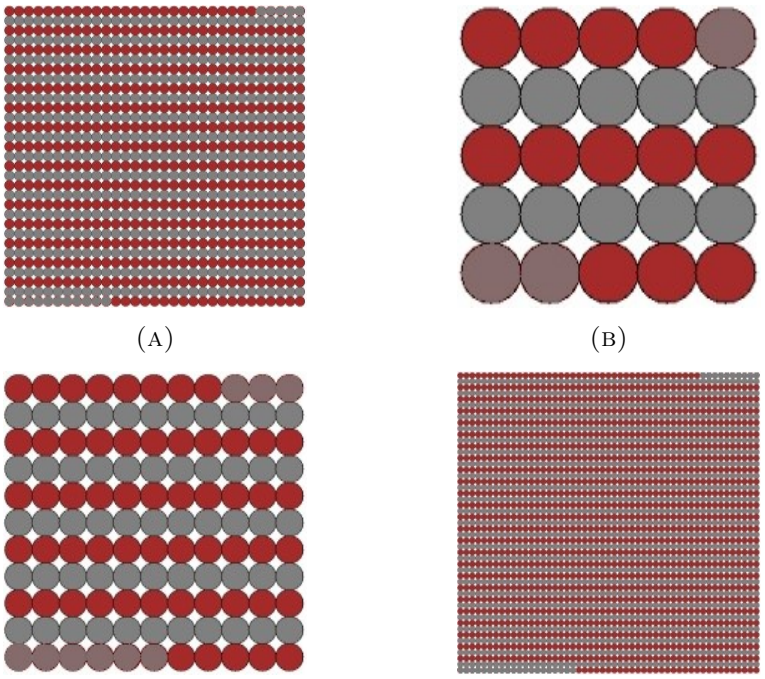
Next, we extend Example 4.2.1 to Example 4.2.2 to generate a gallery of images that consists of a light colour on the upper right corner of the first dark row. The number of the light colour on the lower left corner of the last dark row is twice the number of the light colour on the upper right corner of the first dark row. See Figure 4.3.

Example 4.2.2. Let $G_{\text{Carpet}-\text{B}} = (V_M, V_T, R, S, \{1, 2, 3, 4, 5\}, 0)$, where $V_M = \{S, A, B, C\}$, $V_T = \{d, w\}$, $(S,(x,y)) = (S,(0,0))$, and R is the set in Figure 4.4.

The strategy is that the first and second bag positions are used to control the variables A–C which help to keep colour on a row. The third bag position is used to ensure the number of rows is the same as the number of columns. The fourth and fifth bag positions are used to ensure the number of the light colour on the lower left corner of the last dark row of the image is twice the number of the light colour on the upper right corner of the first dark row of the image.

Some images in $\mathcal{G}$ produced by $G_{\text{Carpet}-\text{B}}$ are shown in Figure 4.3.

To achieve the strategy, the application of Rule 4.10 of Figure 4.4 starts the derivation. Then Rule 4.2 is applicable as long as the bag at each application is within the defined range, this automatically adds the bag adjustment (1, 0, 1, 0, 0) to the bag at each application. Hence the



(A) (B) (C) (D)

FIGURE 4.3: Some of the images in $\mathcal{G}$ of $G_{\text{Carpet}-\text{B}}$.

application of Rule 4.12 of Figure 4.4 enables Rules 4.13 and 4.15 to be applicable. This is to ensure that a light colour appears at the upper right corner of the first dark row.

After the application of Rule 4.17 of Figure 4.4 any of Rules 4.19–4.23 is applicable as long as the bag at each application is within the range. This process continues until the third bag position is 0, then Rule 4.25 is applicable followed by Rule 4.27. At this point the fourth and fifth bag positions are 0, then Rule 4.29 is applicable until the second bag position is 0. At this point, Rule 4.30 is applicable to complete the picture formation. This strategy is sufficient to generate the images in Figure 4.3. Next, we generate images with subimages using BCSGs.

4.2.2 Generation of Carpet with Different Parts

Here, we extend Example 4.2.2 to Example 4.2.3 to generate a gallery of images with subimages. The subimages are called corners in this section. The gallery generated here consists of light or dark colour on the upper right and lower left corners. The top left corner of each image in the gallery consists of dark and light colours, such that each colour dominates a row. This feature is replicated on the lower right corner of the image. See Figure 4.5.

Example 4.2.3. Let $G_{\text{Carpet}-\text{C}} = (V_M, V_T, R, S, \{1, 2, 3, 4, 5\}, 0)$, where $V_M = \{S, A, B, C, A', A''\}$, $V_T = \{d, w\}$, $(S, (x, y)) = (S, (0, 0))$, and R is the set in Figure 4.6.

Some images in $\mathcal{G}$ produced by $G_{\text{Carpet}-\text{C}}$ are shown in Figure 4.5.

The strategy here is that the variables A, A', and A'' in Example 4.2.3 are used for the upper left corner of the image. The variables B and C can be interchangeably used for the lower left

$$
R = \Bigg\{
$$

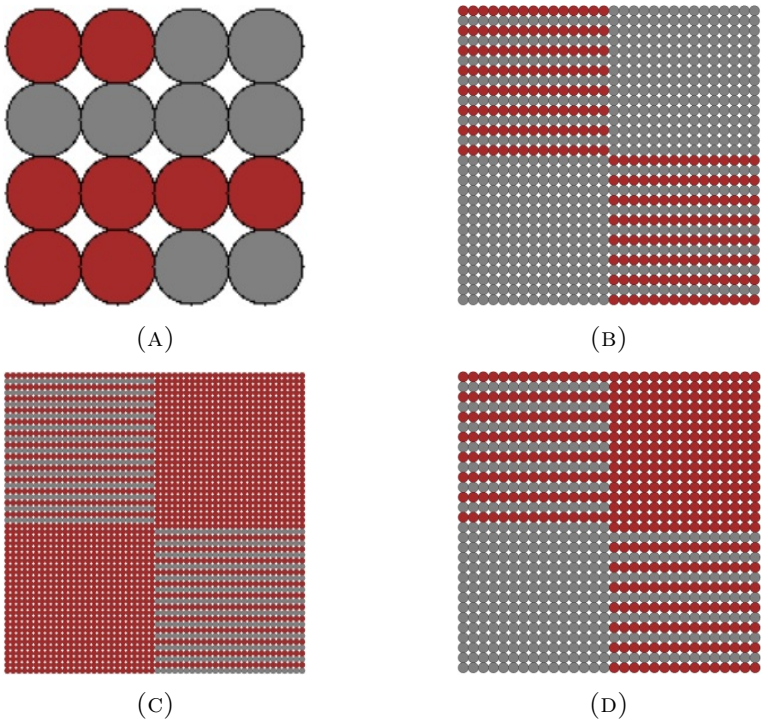
$$
\begin{array}{lcll}
(S,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y))\}(0,0;(1,0,1,0,0)) & \text{(Rule 4.10)}\\
(A,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y))\} & \text{(Rule 4.11)}\\
 & & \quad((1,0,1,0,0),(\infty,0,\infty,0,0);(1,0,1,0,0)) \mid & \text{(Rule 4.12)}\\
 & & \{(w,(x,y)),(B,(x,y-1))\} & \\
 & & \quad((2,0,2,0,0),(\infty,0,\infty,\infty,\infty);(0,0,-1,1,0)) \mid & \text{(Rule 4.13)}\\
 & & \{(w,(x,y)),(A,(x+1,y))\} & \text{(Rule 4.14)}\\
 & & \quad((2,0,2,0,0),(\infty,0,\infty,\infty,\infty);(1,0,1,1,0)) & \text{(Rule 4.15)}\\
(B,(x,y)) & \longrightarrow & \{(B,(x-1,y)),(w,(x,y))\} & \text{(Rule 4.16)}\\
 & & \quad((1,0,0,0,0),(\infty,\infty,\infty,\infty,0);(-1,1,0,0,0)) \mid & \text{(Rule 4.17)}\\
 & & \{(w,(x,y)),(C,(x,y-1))\} & \text{(Rule 4.18)}\\
 & & \quad((0,1,1,1,0),(0,\infty,\infty,\infty,0);(0,0,-1,0,0)) & \text{(Rule 4.19)}\\
(C,(x,y)) & \longrightarrow & \{(d,(x,y)),(C,(x+1,y))\} & \text{(Rule 4.20)}\\
 & & \quad((0,1,1,1,0),((\infty,\infty,\infty,\infty,0));(1,-1,0,0,0)) \mid & \text{(Rule 4.21)}\\
 & & \{(d,(x,y)),(B,(x,y-1))\} & \text{(Rule 4.22)}\\
 & & \quad((1,0,1,1,0),((\infty,0,\infty,\infty,0));(0,0,-1,0,0)) \mid & \text{(Rule 4.23)}\\
 & & \{(w,(x,y)),(C,(x+1,y))\} & \text{(Rule 4.24)}\\
 & & \quad((0,1,0,1,0),(\infty,\infty,0,\infty,\infty);(0,-1,0,-1,1)) \mid & \text{(Rule 4.25)}\\
 & & \{(w,(x,y)),(C,(x+1,y))\} & \text{(Rule 4.26)}\\
 & & \quad((0,1,0,0,1),(\infty,\infty,0,0,\infty);(0,-1,0,0,-1)) \mid & \text{(Rule 4.27)}\\
 & & \{(d,(x,y)),(C,(x+1,y))\} & \text{(Rule 4.28)}\\
 & & \quad((0,1,0,0,0),(\infty,\infty,0,0,0);(0,-1,0,0,0)) \mid & \text{(Rule 4.29)}\\
 & & \{(d,(x,y))\}(0,0;0) & \text{(Rule 4.30)}
\end{array}
$$

$$
\Bigg\}
$$

FIGURE 4.4: The set of rules for $G_{\text{Carpet}-\text{B}}$.

and upper right corners respectively or the upper right corner can also be mirrored on the lower left corner depending on which rule is applied during derivation. Meanwhile, the lower right corner of the image is a reflection of the upper left corner. The derivation starts from the upper left corner to the lower left corner then to the upper right and lower right corners of the image respectively. The size of the upper left corner of the image determines the size of the other three corners (parts) of the image. See Figure 4.5.

To achieve this strategy, the first and second bag positions are used to control the light and dark colours respectively. The third bag position is used to ensure the number of rows is equal to the number of columns. The fourth and fifth bag positions are used to control the lower left, upper right and lower right corners of the image respectively. The lower right corner mimics the upper left corner of the image. Such that the number of light and dark colours in the upper left corner is exactly the same in the lower right corner. The size of the upper left corner is duplicated on the lower left corner with a dark or light colour. Then the lower left corner of the image is mirrored on the upper right corner of the image with the same or an opposite colour depending on which rule is applied.



(A) (B) (C) (D)

FIGURE 4.5: Some of the images in $\mathcal{G}$ of $G_{\text{Carpet}-\text{C}}$.

The application of Rule 4.32 of Figure 4.6 after Rule 4.31 starts the formation of the upper left corner of the image. The bag adjustment, (1, 0, 1, 0, 0) is added to the bag as long as the bag at each application is within the range, this continues until Rule 4.33 or 4.34 is applied. The application of Rule 4.33 or 4.34 begins the formation of a new row and creates room for the formation of the upper right corner. Any of Rules 4.35–4.44 is applicable as long as the bag at each application is within the range. This process will form the upper left corner of the image, then begins the formation of the lower left corner with the application of Rule 4.42 or 4.43. If Rule 4.42 is applied then Rules 4.45–4.51 are applicable when the bag is within the range at each application. On the other hand, If Rule 4.43 is applied then Rules 4.52–4.58 are applicable. The lower left corner formation is completed by the application of Rule 4.51 or 4.56.

The formation of the upper right corner begins by the application of Rule 4.45 or 4.52 of Figure 4.6[1]. The bag adjustment, (1, -1, 0, 0, 0) is added to the bag until the second bag position is 0. Then any of Rules 4.46–4.48 or 4.53–4.55 is applicable so long as the bag at each application is within the range. This process continues until the third bag position is 0 then the upper right corner is complete. The formation of the lower right corner begins with the application of any of Rule 4.49, 4.51, 4.57 or 4.58, as long as the bag is within the range. At this point, any of Rules 4.35–4.44 is now applicable as long as the bag at each application is within the range as usual. This process continues until the lower right corner formation is complete. Then the entire picture formation is now complete. This strategy is sufficient to generate the images in Figure 4.5.

[1]See Figure 4.6 on page 84.

Next, we extend Example 4.2.3 to Example 4.2.4 to generate a gallery of images that consists of light or dark colour on the top right and lower left corners. The top left corner of each image in the gallery consists of dark and light colours, such that each colour dominates a row. The lower right corner of the image consists of light and dark colours, such that no colour dominates a row. Hence the upper left and lower right corners (parts) of the image are not the same, see Figure 4.7[2].

Example 4.2.4. Let $G_{\text{Carpet}-\text{D}} = (V_M, V_T, R, S, \{1, 2, 3, 4, 5\}, 0)$, where $V_M = \{S, A, B, C, D, A', A'', D'\}$, $V_T = \{d, w\}$, $(S, (x, y)) = (S, (0, 0))$, and R is the set in Figures 4.8(A)[3] and (B).

Some images in $\mathcal{G}$ produced by $G_{\text{Carpet}-\text{D}}$ are shown in Figure 4.7.

The strategy in Example 4.2.4 follows the same process of the formation of the upper left, lower left, and upper right corners of the images in Example 4.2.3. On the lower right corner of Example 4.2.4 generated images, no colour is allowed to dominate a row or column, see Figure 4.7.

To achieve the strategy, the fifth bag position is used to control when and when not to apply a dark or light colour. When the fifth bag position is 1 then a dark colour can be applied otherwise a light colour is applied. This ensures no colour is repeated immediately on a row or column after a rule that contains the colour has been applied. The lower right corner of the image is formed after all the other parts of the image have been completed. If the fifth bag position is 1 when the upper right corner of the image is formed then Rule 4.82 is applicable followed by Rules 4.91–4.94. If the fifth bag position is 0 when the upper right corner of the image is formed then Rule 4.84 is applicable followed by Rules 4.85–4.88.

This process continues until the third bag position is 0. The third bag position ensures the number of rows is equal to the number of columns. Then the process can be terminated by the application of any of the Rules 4.89, 4.90, 4.95, or 4.96 of Figure 4.8 as long as the bag is within the range at the time of application. This strategy is sufficient to generate the images in Figure 4.7.

Finally, we extend Example 4.2.4 to Example 4.2.5 to generate a gallery of images that consists of the following:

- a colour dominates a row on the upper left corner,
- a dark or light colour fills the lower left corner,
- a light or dark colour fills the upper right corner,
- the lower right corner consists of the following:
  - light and dark colours which fills the lower left and upper right corners respectively,
  - a colour is not allowed to dominate a row or column on the upper left and lower right corners respectively, see Figure 4.9[4].

[2]See Figure 4.7 on page 85.
[3]See Figures 4.8(A) and (B) on pages 86 and 87 respectively.
[4]See Figure 4.9 on page 88.

Example 4.2.5. Let $G_{\text{Carpet}-\text{E}} = (V_M, V_T, R, S, \{1, 2, 3, 4, 5\}, 0)$, where $V_M = \{S, A, B, C, D, A', A'', D', D''\}$, $V_T = \{d, w\}$, $(S, (x, y)) = (S, (0, 0))$, and R is the set in Figures 4.10 (A) and (B)[5].

Some images in $\mathcal{G}$ produced by $G_{\text{Carpet}-\text{E}}$ are shown in Figure 4.9.

The strategy is that the first and second bag positions are used to control the colours. The third bag position ensures the number of rows and columns are the same. The fourth and fifth bag positions are used to control the parts of the image.

To achieve the strategy, Rule 4.98 of Figure 4.10 (A) can be applied continuously after Rule 4.97, this adds the bag adjustment, (1, 0, 1, 0, 0) to the bag until any of Rule 4.99 or Rule 4.100 is applied. The application of Rule 4.99 or Rule 4.100 begins the formation of a new row and creates a room for the formation of the upper right corner. Then any of Rules 4.101–4.108 is applicable as long as the bag at each application is within the range. This process continues until the third bag position is 0 then any of Rule 4.103, 4.104, 4.107 or 4.108 is applicable if the bag is within the range at the time of application, this helps to complete the upper left corner of the image and move to the lower left part.

When the upper left corner is complete, any of Rules 4.109–4.112 or 4.115–4.120 of Figure 4.10 (A) is applicable until the fourth bag position is 0, then any of Rule 4.114 or Rule 4.121 can be applied to complete the lower left corner formation.

The application of Rule 4.115 or Rule 4.109 of Figure 4.10 (A) begins the formation of the upper right part of the image. The bag adjustment, (1, -1, 0, 0, 0) is added to the bag until the second bag position is 0. Then any of Rules 4.110–4.112 or 4.116–4.118 is applicable until Rule 4.113 or Rule 4.120 is applied. At this time the upper right corner is complete.

The application of Rule 4.122 or Rule 4.123 of Figure 4.10 (B) adds the bag adjustment, (1, -2, 1, -2, 1) to the bag, hence adding half the count of the content of the second and fourth bag positions to the first, third and fifth bag positions. This is to ensure the sizes of the parts of the lower right corner of the image are half the sizes of other parts of the image. Any of Rules 4.123–4.137 is applicable as long as the bag is within the range at each application until the lower right corner of the image is complete. This strategy is sufficient to generate the gallery in Figure 4.9.

4.3 Generation of Similar Images By Controlling The Building Levels

The building levels is the number of times shapes are added to generate images or subimages from the initial shape. We demonstrate how BCSGs can generate similar images within some defined bounds (lower and upper limits), thus within limited building levels. This is achieved

[5]See Figures 4.10 (A) and (B) on pages 89 and 90 respectively.

by generating similar images with the same building levels and different colour variations or generate similar images with different building levels and different colour variation.

4.3.1 Same Building Level with Colour Variation

In this part, we rewrite Example 4.2.5 in Examples 4.3.1 to generate only images with the same building level but with different colour placement using BCSGs see Figure 4.11[6]. This is achieved by limiting the number of some of the lower and upper limits of the rules in Figures 4.12(A) and (B)[7] to 24.

Example 4.3.1. Let a BCSG, $G_{\text{SameBuild}} = (V_M, V_T, R, S, \{1, 2, 3, 4, 5\}, 0)$, where $V_M = \{S, A, B, C, D, A', A'', D', D''\}$, $V_T = \{d, w\}$, $(S, (x, y)) = (S, (0, 0))$, and R is the set in Figures 4.12 (A) and (B).

Some images in $\mathcal{G}$ produced by $G_{\text{SameBuild}}$ are shown in Figure 4.11.

The strategy is that the first and second bag positions are used to control the colours. The third bag position ensures the number of rows and columns are the same. The fourth and fifth bag positions are used to control the parts of the image.

Observe that the highest value for some of the upper limits in Figures 4.12 (A) and (B) is limited to 24. the same as the highest value for some of the lower limits. Particularly, the first position of the lower limit in Rule 4.140 of Figure 4.12. This helps to keep the images in Figure 4.11 at the same building levels during derivation while the colour placement is different depending on the particular rule that is applied.

4.3.2 Different Building Level with Colour Variations

We used bag context to control the generation of images in a way that all the generated images have a little difference in their building levels. Then the variation of the colour is also controlled such that the number of the light colour on the upper part of the image is half the number the of light colour on the lower part of the images. The difference between the images is the variations in the building levels and colour placement. See Figure 4.13[8]. We demonstrate this notion in Example 4.3.2.

Example 4.3.2. We want to generate images with little difference in their building levels with different colour placement. Let $G_{\text{DiffBuild}} = (V_M, V_T, R, S, \{1, 2, 3, 4, 5\}, 0)$, where $V_M = \{S, A, B, C\}$, $V_T = \{d, w\}$, $(S, (x, y)) = (S, (0, 0))$, and R is the set in Figure 4.14[9].

The strategy is that the first and second bag positions are used to control the variables A–C which help to keep a colour on a row. The third bag position is used to ensure the number of

[6] see Figures 4.11 on page 91.
[7] See Figures 4.12 (A) and (B) on pages 92 and 93 respectively.
[8] See Figure 4.13 on page 94.
[9] See Figure 4.14 on page 95.

rows is the same as the number of columns. The fourth and fifth bag positions are used to ensure the number of the light colour on the lower left corner of the last dark row of the image is twice the number of the light colour on the upper right corner of the first dark row of the image. This is achieved by limiting some of the lower limits to 30 and some of the upper limits in the rules to 35. Particularly, Rules 4.181, 4.182 and 4.184 of Figure 4.14 respectively.

This simply means that Rules 4.182 and 4.184 of Figure 4.14 can only be applied when the values in the bag positions one and three are 30 respectively. This will help to generate only images that are similar to the images in Figure 4.13.

Some images in $\mathcal{G}$ produced by $G_{\text{DiffBuild}}$ are shown in Figure 4.13.

Next, we summarise this chapter in Section 4.4.

4.4 Summary

The power of BCGSs was presented in this chapter to demonstrate different methods BCSGs can generate similar images. However, the demonstration of the power of BCSGs was not to say that BCSGs can generate images other shape grammars or the additive shape grammars cannot generate. Rather we demonstrated different methods that BCSGs can generate only similar images in a controlled manner. With BCSGs, it is possible to know the next similar image to be generated to a selected image, avoiding the generation of images that some parts are too small for the human eye to see using the same grammar, which other shape grammars or the additive shape grammars lack the ability to do.

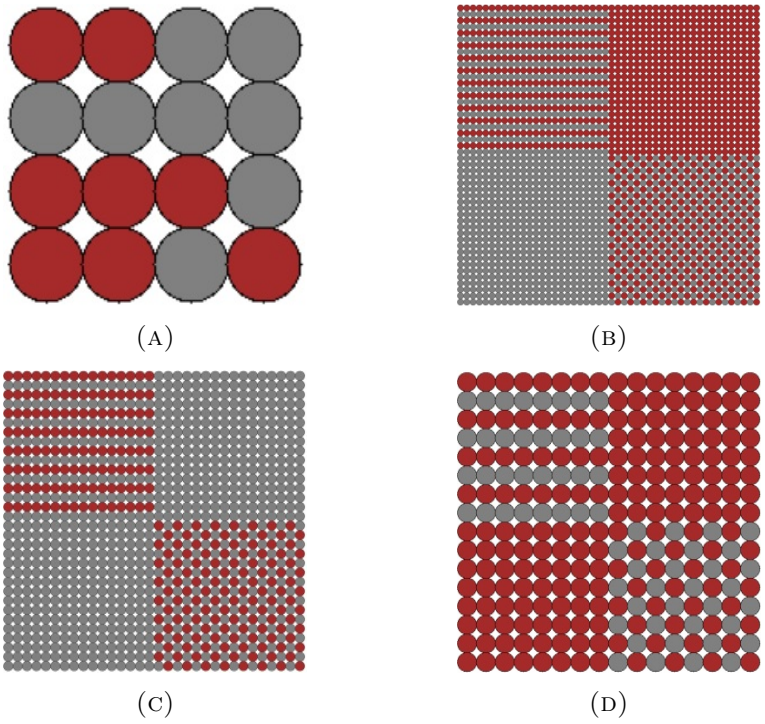
We also demonstrated the power of BCSGs by limiting the lower and upper bounds. This enabled us to generate similar images with the same building level but different colour placement using the same grammar. Furthermore, the BCSGs was used to generate similar images with different building levels and different colour placement using the same grammar. It is observed in the literature (see Sections 2.3.10 and 3.2) that other shape grammars or the additive shape grammars are not capable of controlling the generation of similar images in these manners. Next, we present the bag context shape grammar interpreter in the Chapter 5.

$$R = \Big\{$$

$$
\begin{array}{rcll}
(S,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y))\}(0,0;(1,0,1,0,0)) & \text{(Rule 4.31)}\\
(A,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y))\}((1,0,1,0,0),(\infty,0,\infty,0,0);(1,0,1,0,0)) \mid & \text{(Rule 4.32)}\\
 & & \{(b,(x,y)),(B,(x+1,y),(A',(x-1,y))\} & \\
 & & \quad((1,0,1,0,0),(\infty,0,\infty,\infty,0);(0,0,-1,1,0)) & \text{(Rule 4.33)}\\
 & & \{(b,(x,y)),(C,(x+1,y),(A',(x-1,y))\} & \\
 & & \quad((1,0,1,0,0),(\infty,0,\infty,\infty,0);(0,0,-1,1,0)) & \text{(Rule 4.34)}\\
(A',(x,y)) & \longrightarrow & \{(A',(x-1,y)),(w,(x,y))\} & \\
 & & \quad((1,0,0,0,0),(\infty,\infty,\infty,\infty,0);(-1,1,0,0,0)) \mid & \text{(Rule 4.35)}\\
 & & \{(w,(x,y)),(A'',(x,y-1))\} & \\
 & & \quad((0,1,1,0,0),(0,\infty,\infty,\infty,0);(0,0,-1,1,0)) \mid & \text{(Rule 4.36)}\\
 & & \{(d,(x,y)),(B,(x,y-1))\}((1,0,0,0,0),(\infty,0,0,\infty,0);0) \mid & \text{(Rule 4.37)}\\
 & & \{(w,(x,y)),(C,(x,y-1))\}((0,1,0,0,0),(0,\infty,0,\infty,0);0) \mid & \text{(Rule 4.38)}\\
 & & \{(d,(x,y))\}(0,(0,\infty,0,\infty,\infty);0) & \text{(Rule 4.39)}\\
(A'',(x,y)) & \longrightarrow & \{(d,(x,y)),(A'',(x+1,y))\}((0,1,0,0,0),\infty;(1,-1,0,0,0)) \mid & \text{(Rule 4.40)}\\
 & & \{(d,(x,y)),(A',(x,y-1))\}((1,0,1,0,0),\infty;(0,0,-1,1,0)) \mid & \text{(Rule 4.41)}\\
 & & \{(d,(x,y)),(C,(x,y-1))\}((1,0,0,0,0),(\infty,0,0,\infty,0);0) \mid & \text{(Rule 4.42)}\\
 & & \{(d,(x,y)),(B,(x,y-1))\}((1,0,0,0,0),(\infty,0,0,\infty,0);0) \mid & \text{(Rule 4.43)}\\
 & & \{(d,(x,y))\}(0,(\infty,0,0,\infty,\infty);0) & \text{(Rule 4.44)}\\
(C,(x,y)) & \longrightarrow & \{(d,(x,y)),(C,(x+1,y))\} & \\
 & & \quad((0,1,0,0,0),(\infty,\infty,\infty,\infty,0);(1,-1,0,0,0)) \mid & \text{(Rule 4.45)}\\
 & & \{(C,(x-1,y)),(d,(x,y))\} & \\
 & & \quad((1,0,1,0,1),(\infty,\infty,\infty,\infty,1);(-1,1,0,0,0)) \mid & \text{(Rule 4.46)}\\
 & & \{(d,(x,y)),(C,(x,y-1))\} & \\
 & & \quad((0,1,0,1,1),(0,\infty,\infty,\infty,1);(0,0,1,-1,-1)) \mid & \text{(Rule 4.47)}\\
 & & \{(d,(x,y)),(C,(x,y-1))\} & \\
 & & \quad((1,0,0,1,0),(\infty,0,\infty,\infty,0);(0,0,1,-1,1)) \mid & \text{(Rule 4.48)}\\
 & & \{(w,(x,y)),(A'',(x,y-1))\} & \\
 & & \quad((0,1,0,0,1),(0,\infty,0,\infty,1);(0,0,0,0,-1)) & \text{(Rule 4.49)}\\
 & & \{(w,(x,y)),(A',(x,y-1))\} & \\
 & & \quad((1,0,0,0,0),(\infty,0,0,\infty,0);(0,0,0,0,0)) & \text{(Rule 4.50)}\\
 & & \{(d,(x,y))\}(0,(0,\infty,\infty,0,\infty);(0,0,0,0,-1)) & \text{(Rule 4.51)}\\
(B,(x,y)) & \longrightarrow & \{(w,(x,y)),(B,(x+1,y))\} & \\
 & & \quad((0,1,0,0,0),(\infty,\infty,\infty,\infty,0);(1,-1,0,0,0)) \mid & \text{(Rule 4.52)}\\
 & & \{(B,(x+1,y),(w,(x,y)))\} & \\
 & & \quad((1,0,1,0,1),(\infty,\infty,\infty,\infty,1);(-1,1,0,0,0)) \mid & \text{(Rule 4.53)}\\
 & & \{(w,(x,y)),(B,(x,y-1))\} & \\
 & & \quad((1,0,0,0,0),(\infty,0,\infty,\infty,0);(0,0,-1,1,1)) \mid & \text{(Rule 4.54)}\\
 & & \{(w,(x,y)),(B,(x,y-1))\} & \\
 & & \quad((0,1,1,0,1),(0,\infty,\infty,\infty,1);(0,0,-1,1,-1)) \mid & \text{(Rule 4.55)}\\
 & & \{(d,(x,y))\}(0,(0,\infty,\infty,0,\infty);(0,0,0,0,-1)) & \text{(Rule 4.56)}\\
 & & \{(w,(x,y)),(A'',(x,y-1))\}((0,1,0,0,1),(0,\infty,0,\infty,1);(0,0,0,0,-1)) & \text{(Rule 4.57)}\\
 & & \{(w,(x,y)),(A',(x,y-1))\}((1,0,0,0,0),(\infty,0,0,\infty,0);(0,0,0,0,0)) & \text{(Rule 4.58)}
\end{array}
$$

$$\Big\}$$

FIGURE 4.6: The set of rules for $G_{\text{Carpet}-\text{C}}$.



(A) (B)

(C) (D)

FIGURE 4.7: Some of the images in $\mathcal{G}$ of $G_{\text{Carpet}-\text{D}}$.

$$R = \Bigg\{$$

$$
\begin{array}{rcll}
(S,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y))\}(0,0;(1,0,1,0,0)) & \text{(Rule 4.59)}\\
(A,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y))\}((1,0,1,0,0),(\infty,0,\infty,0,0);(1,0,1,0,0)) \mid & \text{(Rule 4.60)}\\
 & & \{(b,(x,y)),(B,(x+1,y),(A',(x-1,y))\} & \\
 & & \quad((1,0,1,0,0),(\infty,0,\infty,\infty,0);(0,0,-1,1,0)) & \text{(Rule 4.61)}\\
 & & \{(b,(x,y)),(C,(x+1,y),(A',(x-1,y))\} & \\
 & & \quad((1,0,1,0,0),(\infty,0,\infty,\infty,0);(0,0,-1,1,0)) & \text{(Rule 4.62)}\\
(A',(x,y)) & \longrightarrow & \{(A',(x-1,y)),(w,(x,y))\} & \\
 & & \quad((1,0,0,0,0),(\infty,\infty,\infty,\infty,0);(-1,1,0,0,0)) \mid & \text{(Rule 4.63)}\\
 & & \{(w,(x,y)),(A'',(x,y-1))\} & \\
 & & \quad((0,1,1,0,0),(0,\infty,\infty,\infty,0);(0,0,-1,1,0)) \mid & \text{(Rule 4.64)}\\
 & & \{(w,(x,y)),(B,(x,y-1))\}((0,1,0,0,0),(0,\infty,0,\infty,0);0) \mid & \text{(Rule 4.65)}\\
 & & \{(w,(x,y)),(C,(x,y-1))\}((0,1,0,0,0),(0,\infty,0,\infty,0);0) & \text{(Rule 4.66)}\\
(A'',(x,y)) & \longrightarrow & \{(d,(x,y)),(A'',(x+1,y))\}((0,1,0,0,0),\infty;(1,-1,0,0,0)) \mid & \text{(Rule 4.67)}\\
 & & \{(d,(x,y)),(A',(x,y-1))\}((1,0,1,0,0),\infty;(0,0,-1,1,0)) \mid & \text{(Rule 4.68)}\\
 & & \{(d,(x,y)),(B,(x,y-1))\}((1,0,0,0,0),(\infty,0,0,\infty,0);0) \mid & \text{(Rule 4.69)}\\
 & & \{(d,(x,y)),(C,(x,y-1))\}((1,0,0,0,0),(\infty,0,0,\infty,0);0) & \text{(Rule 4.70)}\\
(C,(x,y)) & \longrightarrow & \{(d,(x,y)),(C,(x+1,y))\} & \\
 & & \quad((0,1,0,0,0),(\infty,\infty,\infty,\infty,0);(1,-1,0,0,0)) \mid & \text{(Rule 4.71)}\\
 & & \{(C,(x-1,y)),(d,(x,y))\} & \\
 & & \quad((1,0,1,0,1),(\infty,\infty,\infty,\infty,1);(-1,1,0,0,0)) \mid & \text{(Rule 4.72)}\\
 & & \{(d,(x,y)),(C,(x,y-1))\} & \\
 & & \quad((0,1,0,1,1),(0,\infty,\infty,\infty,1);(0,0,1,-1,-1)) \mid & \text{(Rule 4.73)}\\
 & & \{(d,(x,y)),(C,(x,y-1))\} & \\
 & & \quad((1,0,0,1,0),(\infty,0,\infty,\infty,0);(0,0,1,-1,1)) \mid & \text{(Rule 4.74)}\\
 & & \{(d,(x,y))\}(0,(0,\infty,\infty,0,\infty);(0,0,0,0,-1)) \mid & \text{(Rule 4.75)}\\
 & & \{(w,(x,y)),(D',(x,y-1))\} & \\
 & & \quad((0,1,0,0,1),(0,\infty,0,\infty,1);(0,0,0,0,-1)) \mid & \text{(Rule 4.76)}\\
 & & \{(w,(x,y)),(D,(x,y-1))\}((1,0,0,0,0),(\infty,0,0,\infty,0);(0,0,0,0,1)) & \text{(Rule 4.77)}\\
(B,(x,y)) & \longrightarrow & \{(w,(x,y)),(B,(x+1,y))\} & \\
 & & \quad((0,1,0,0,0),(\infty,\infty,\infty,\infty,0);(1,-1,0,0,0)) \mid & \text{(Rule 4.78)}\\
 & & \{(B,(x+1,y),(w,(x,y)))\} & \\
 & & \quad((1,0,1,0,1),(\infty,\infty,\infty,\infty,1);(-1,1,0,0,0)) \mid & \text{(Rule 4.79)}\\
 & & \{(w,(x,y)),(B,(x,y-1))\} & \\
 & & \quad((1,0,0,0,0),(\infty,0,\infty,\infty,0);(0,0,-1,1,1)) \mid & \text{(Rule 4.80)}\\
 & & \{(w,(x,y)),(B,(x,y-1))\} & \\
 & & \quad((0,1,1,0,1),(0,\infty,\infty,\infty,1);(0,0,-1,1,-1)) \mid & \text{(Rule 4.81)}\\
 & & \{(w,(x,y)),(D',(x,y-1))\} & \\
 & & \quad((0,1,0,0,1),(0,\infty,0,\infty,1);(0,0,0,0,-1)) \mid & \text{(Rule 4.82)}\\
 & & \{(d,(x,y))\}(0,(0,\infty,\infty,0,\infty);(0,0,0,0,-1)) \mid & \text{(Rule 4.83)}\\
 & & \{(w,(x,y)),(D,(x,y-1))\}((1,0,0,0,0),(\infty,0,0,\infty,0);(0,0,0,0,1)) & \text{(Rule 4.84)}
\end{array}
$$

$$\Bigg\}$$

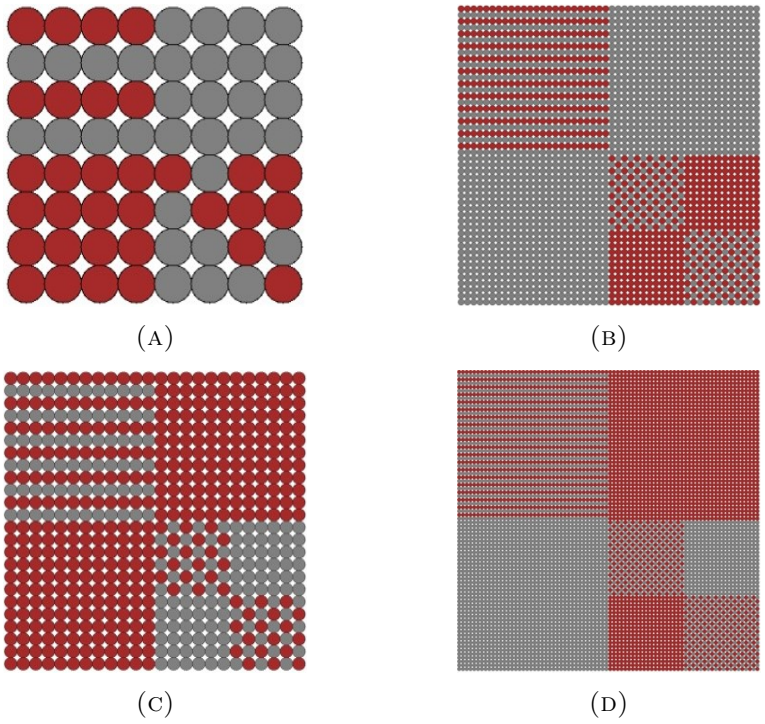
FIGURE 4.8: (A) The set of rules for $G_{\text{Carpet}-\text{D}}$.

$$
R = \Bigg\{
$$

$$
\begin{array}{rcll}
(D,(x,y)) & \longrightarrow & \{(D,(x-1,y)),(d,(x,y))\} & \\
 & & \quad ((1,0,0,0,1),(\infty,\infty,\infty,\infty,1);(-1,1,0,0,-1)) \mid & \text{(Rule 4.85)} \\
 & & \{(D,(x-1,y)),(w,(x,y))\} & \\
 & & \quad ((1,0,0,0,0),(\infty,\infty,\infty,\infty,0);(-1,1,0,0,1)) \mid & \text{(Rule 4.86)} \\
 & & \{(b,(x,y)),(D',(x,y-1))\} & \\
 & & \quad ((0,1,0,1,1),(0,\infty,\infty,1,1);(0,0,1,-1,-1)) \mid & \text{(Rule 4.87)} \\
 & & \{(w,(x,y)),(D',(x,y-1))\} & \\
 & & \quad ((0,1,0,0,0),(0,\infty,\infty,\infty,0);(0,0,1,-1,1)) \mid & \text{(Rule 4.88)} \\
 & & \{(w,(x,y))\}(0,(\infty,\infty,\infty,0,0);0) \mid & \text{(Rule 4.89)} \\
 & & \{(d,(x,y))\}((0,0,0,0,1),(\infty,\infty,\infty,0,1);0) & \text{(Rule 4.90)} \\
(D',(x,y)) & \longrightarrow & \{(b,(x,y)),(D',(x+1,y))\} & \\
 & & \quad ((0,1,0,1,0),(\infty,\infty,\infty,0);(1,-1,0,0,1)) \mid & \text{(Rule 4.91)} \\
 & & \{(w,(x,y)),(D',(x+1,y))\} & \\
 & & \quad ((0,1,1,0,1),(\infty,\infty,\infty,\infty,1);(1,-1,0,0,-1)) \mid & \text{(Rule 4.92)} \\
 & & \{(w,(x,y)),(D,(x,y-1)),\} & \\
 & & \quad ((1,0,0,0,0),(\infty,0,\infty,\infty,0);(0,0,1,-1,1)) \mid & \text{(Rule 4.93)} \\
 & & \{(d,(x,y)),(D,(x,y-1))\} & \\
 & & \quad ((1,0,1,1,1),(\infty,0,\infty,\infty,1);(0,0,1,-1,-1)) \mid & \text{(Rule 4.94)} \\
 & & \{(w,(x,y))\}(0,(\infty,\infty,0,0,0);0) \mid & \text{(Rule 4.95)} \\
 & & \{(d,(x,y))\}((0,0,0,0,1),(\infty,\infty,\infty,0,1);0) & \text{(Rule 4.96)}
\end{array}
$$

$$
\Bigg\}
$$

FIGURE 4.8: (B) The set of rules for $G_{\text{Carpet}-\text{D}}$.



(A) (B)

(C) (D)

FIGURE 4.9: Some of the images in $\mathcal{G}$ of $G_{\text{Carpet}-\text{E}}$.

$$
R = \Bigg\{
$$

$$
\begin{aligned}
(S,(x,y)) &\longrightarrow \{(d,(x,y)),(A,(x+1,y))\}(0,0;(1,0,1,0,0)) && \text{(Rule 4.97)}\\
(A,(x,y)) &\longrightarrow \{(d,(x,y)),(A,(x+1,y))\} && \\
&\quad ((1,0,1,0,0),(\infty,0,\infty,0,0);(1,0,1,0,0)) \mid && \text{(Rule 4.98)}\\
&\quad \{(b,(x,y)),(B,(x+1,y),(A',(x-1,y))\} && \\
&\quad ((1,0,1,0,0),(\infty,0,\infty,\infty,0);(0,0,-1,1,0)) \mid && \text{(Rule 4.99)}\\
&\quad \{(b,(x,y)),(C,(x+1,y),(A',(x-1,y))\} && \\
&\quad ((1,0,1,0,0),(\infty,0,\infty,\infty,0);(0,0,-1,1,0)) && \text{(Rule 4.100)}\\
(A',(x,y)) &\longrightarrow \{(A',(x-1,y)),(w,(x,y))\} && \\
&\quad ((1,0,0,0,0),(\infty,\infty,\infty,\infty,0);(-1,1,0,0,0)) \mid && \text{(Rule 4.101)}\\
&\quad \{(w,(x,y)),(A'',(x,y-1))\} && \\
&\quad ((0,1,1,0,0),(0,\infty,\infty,\infty,0);(0,0,-1,1,0)) \mid && \text{(Rule 4.102)}\\
&\quad \{(w,(x,y)),(B,(x,y-1))\} && \\
&\quad ((0,1,0,0,0),(0,\infty,0,\infty,0);0) \mid && \text{(Rule 4.103)}\\
&\quad \{(w,(x,y)),(C,(x,y-1))\} && \\
&\quad ((0,1,0,0,0),(0,\infty,0,\infty,0);0) && \text{(Rule 4.104)}\\
(A'',(x,y)) &\longrightarrow \{(d,(x,y)),(A'',(x+1,y))\} && \\
&\quad ((0,1,0,0,0),\infty;(1,-1,0,0,0)) \mid && \text{(Rule 4.105)}\\
&\quad \{(d,(x,y)),(A',(x,y-1))\} && \\
&\quad ((1,0,1,0,0),\infty;(0,0,-1,1,0)) \mid && \text{(Rule 4.106)}\\
&\quad \{(d,(x,y)),(B,(x,y-1))\} && \\
&\quad ((1,0,0,0,0),(\infty,0,0,\infty,0);0) && \text{(Rule 4.107)}\\
&\quad \{(d,(x,y)),(C,(x,y-1))\} && \\
&\quad ((1,0,0,0,0),(\infty,0,0,\infty,0);0) && \text{(Rule 4.108)}\\
(C,(x,y)) &\longrightarrow \{(d,(x,y)),(C,(x+1,y))\} && \\
&\quad ((0,1,0,0,0),(\infty,\infty,\infty,\infty,0);(1,-1,0,0,0)) \mid && \text{(Rule 4.109)}\\
&\quad \{(C,(x-1,y)),(d,(x,y))\} && \\
&\quad ((1,0,1,0,1),(\infty,\infty,\infty,\infty,1);(-1,1,0,0,0)) \mid && \text{(Rule 4.110)}\\
&\quad \{(d,(x,y)),(C,(x,y-1))\} && \\
&\quad ((0,1,0,1,1),(0,\infty,\infty,\infty,1);(0,0,1,-1,-1)) \mid && \text{(Rule 4.111)}\\
&\quad \{(d,(x,y)),(C,(x,y-1))\} && \\
&\quad ((1,0,0,1,0),(\infty,0,\infty,\infty,0);(0,0,1,-1,1)) \mid && \text{(Rule 4.112)}\\
&\quad \{(w,(x,y)),(D,(x,y-1))\} && \\
&\quad ((0,1,0,0,0),(0,\infty,0,\infty,1);(0,0,0,0,-1)) \mid && \text{(Rule 4.113)}\\
&\quad \{(d,(x,y))\}(0,(0,\infty,\infty,0,\infty);(0,0,0,0,-1)) && \text{(Rule 4.114)}\\
(B,(x,y)) &\longrightarrow \{(w,(x,y)),(B,(x+1,y))\} && \\
&\quad ((0,1,0,0,0),(\infty,\infty,\infty,\infty,0);(1,-1,0,0,0)) \mid && \text{(Rule 4.115)}\\
&\quad \{(B,(x+1,y),(w,(x,y)))\} && \\
&\quad ((1,0,1,0,1),(\infty,\infty,\infty,\infty,1);(-1,1,0,0,0)) \mid && \text{(Rule 4.116)}\\
&\quad \{(w,(x,y)),(B,(x,y-1))\} && \\
&\quad ((1,0,0,0,0),(\infty,0,\infty,\infty,0);(0,0,-1,1,1)) \mid && \text{(Rule 4.117)}\\
&\quad \{(w,(x,y)),(B,(x,y-1))\} && \\
&\quad ((0,1,1,0,1),(0,\infty,\infty,\infty,1);(0,0,-1,1,-1)) \mid && \text{(Rule 4.118)}\\
& && \text{(Rule 4.119)}
\end{aligned}
$$

$$
\Bigg\}
$$

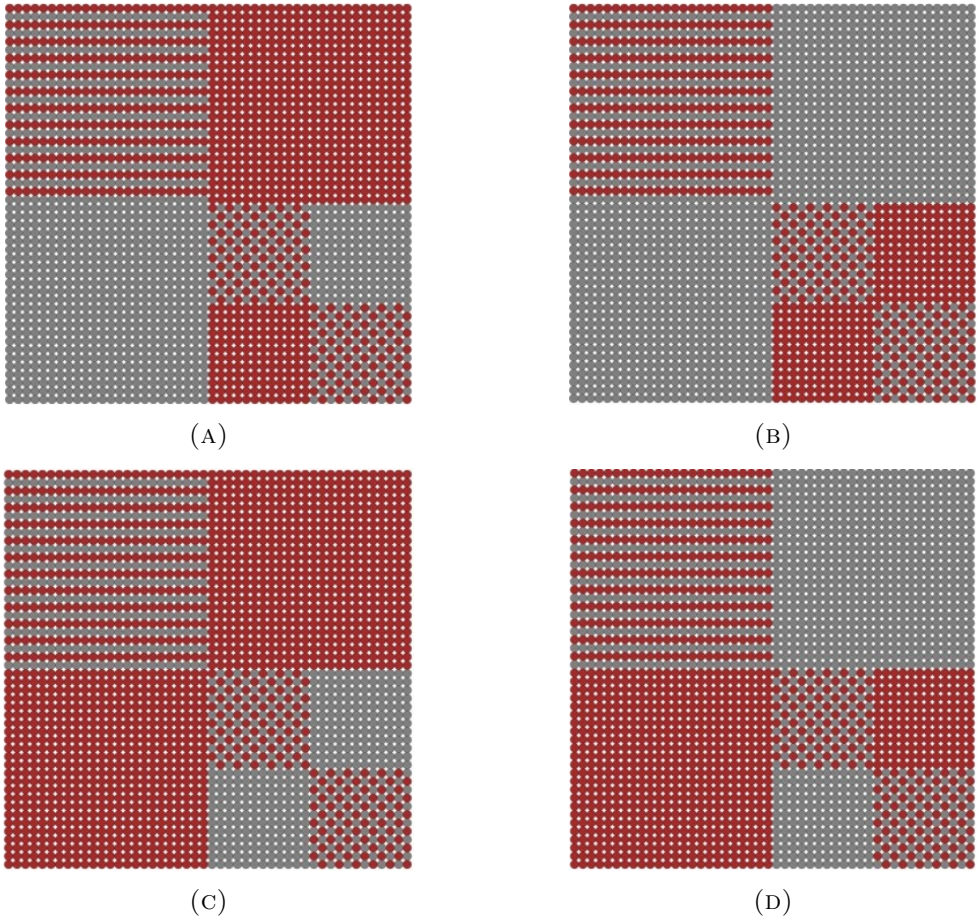
FIGURE 4.10: (A) The set of rules for $G_{\text{Carpet}-\text{E}}$.

$$
R = \Bigg\{
$$

$$
\begin{array}{rcll}
 & & \{(w,(x,y)),(D,(x,y-1))\} & \\
 & & \quad((0,1,0,0,0),(0,\infty,0,\infty,1);(0,0,0,0,-1)) \mid & \text{(Rule 4.120)} \\
 & & \{(d,(x,y))\}(0,(0,\infty,\infty,0,\infty);(0,0,0,0,-1)) & \text{(Rule 4.121)} \\
(D,(x,y)) & \longrightarrow & \{(d,(x,y)),(D,(x+1,y))\} & \\
 & & \quad((0,1,0,1,0),(\infty,\infty,\infty,\infty,0);(1,-2,1,-2,1)) \mid & \text{(Rule 4.122)} \\
 & & \{(w,(x,y)),(D,(x+1,y))\} & \\
 & & \quad((0,1,0,1,1),(\infty,\infty,\infty,\infty,1);(1,-2,1,-2,-1)) \mid & \text{(Rule 4.123)} \\
 & & \{(b,(x,y)),(D',(x,y-1)),(C,(x+1,y))\} & \\
 & & \quad((1,0,1,0,0),(\infty,0,\infty,0,1);(0,0,-1,1,1)) \mid & \text{(Rule 4.124)} \\
 & & \{(w,(x,y)),(D',(x,y-1)),(B,(x+1,y))\} & \\
 & & \quad((1,0,1,0,1),(\infty,0,\infty,0,1);(0,0,-1,1,-1)) & \text{(Rule 4.125)} \\
(D',(x,y)) & \longrightarrow & \{(D',(x-1,y)),(b,(x,y))\} & \\
 & & \quad((1,0,0,1,0),(\infty,\infty,\infty,\infty,0);(-1,1,0,0,1)) \mid & \text{(Rule 4.126)} \\
 & & \{(D',(x-1,y)),(w,(x,y))\} & \\
 & & \quad((1,0,0,1,1),(\infty,\infty,\infty,\infty,1);(-1,1,0,0,-1)) \mid & \text{(Rule 4.127)} \\
 & & \{(w,(x,y)),(D'',(x,y-1)),\} & \\
 & & \quad((0,1,1,0,1),(0,\infty,\infty,\infty,1);(0,0,-1,1,-1)) \mid & \text{(Rule 4.128)} \\
 & & \{(d,(x,y)),(D'',(x,y-1))\} & \\
 & & \quad((0,1,1,0,0),(0,\infty,\infty,\infty,0);(0,0,-1,1,1)) \mid & \text{(Rule 4.129)} \\
 & & \{(b,(x,y)),(C,(x,y-1))\} & \\
 & & \quad((0,1,0,0,0),(0,\infty,0,\infty,0);0) \mid & \text{(Rule 4.130)} \\
 & & \{(w,(x,y)),(B,(x,y-1))\} & \\
 & & \quad((0,1,0,0,1),(0,\infty,0,\infty,1);0) & \text{(Rule 4.131)} \\
D'',(x,y)) & \longrightarrow & \{(b,(x,y)),(D'',(x-1,y))\} & \\
 & & \quad((0,1,0,1,0),(\infty,\infty,\infty,\infty,0);(1,-1,0,0,1)) \mid & \text{(Rule 4.132)} \\
 & & \{(w,(x,y)),(D'',(x-1,y))\} & \\
 & & \quad((0,1,0,1,1),(\infty,\infty,\infty,\infty,1);(1,-1,0,0,-1)) \mid & \text{(Rule 4.133)} \\
 & & \{(b,(x,y)),(D',(x,y-1))\} & \\
 & & \quad((1,0,1,0,0),(\infty,0,\infty,0,1);(0,0,-1,1,1)) \mid & \text{(Rule 4.134)} \\
 & & \{(w,(x,y)),(D',(x,y-1))\} & \\
 & & \quad((1,0,1,0,1),(\infty,0,\infty,0,1);(0,0,-1,1,-1)) \mid & \text{(Rule 4.135)} \\
 & & \{(b,(x,y)),(B,(x,y-1))\} & \\
 & & \quad((0,1,0,0,0),(0,\infty,0,\infty,0);0) \mid & \text{(Rule 4.136)} \\
 & & \{(w,(x,y)),(C,(x,y-1))\} & \\
 & & \quad((0,1,0,0,1),(0,\infty,0,\infty,1);0) & \text{(Rule 4.137)}
\end{array}
$$

$$
\Bigg\}
$$

FIGURE 4.10: (B) The set of rules for $G_{\text{Carpet}-\text{E}}$.



(A) (B)

(C) (D)

FIGURE 4.11: Some of the images in $\mathcal{G}$ of $G_{\text{SameBuild}}$.

$$
R = \Bigg\{
$$

$$
\begin{aligned}
(S,(x,y)) &\longrightarrow \{(d,(x,y)),(A,(x+1,y))\}(0,0;(1,0,1,0,0)) && \text{(Rule 4.138)}\\
(A,(x,y)) &\longrightarrow \{(d,(x,y)),(A,(x+1,y))\} && \\
&\quad ((1,0,1,0,0),(24,0,24,0,0);(1,0,1,0,0)) \mid && \text{(Rule 4.139)}\\
&\quad \{(b,(x,y)),(B,(x+1,y),(A',(x-1,y))\} && \\
&\quad ((24,0,1,0,0),(24,0,24,24,0);(0,0,-1,1,0)) \mid && \text{(Rule 4.140)}\\
&\quad \{(b,(x,y)),(C,(x+1,y),(A',(x-1,y))\} && \\
&\quad ((24,0,1,0,0),(24,0,24,24,0);(0,0,-1,1,0)) && \text{(Rule 4.141)}\\
(A',(x,y)) &\longrightarrow \{(A',(x-1,y)),(w,(x,y))\} && \\
&\quad ((1,0,0,0,0),(24,24,24,24,0);(-1,1,0,0,0)) \mid && \text{(Rule 4.142)}\\
&\quad \{(w,(x,y)),(A'',(x,y-1))\} && \\
&\quad ((0,1,1,0,0),(0,24,24,24,0);(0,0,-1,1,0)) \mid && \text{(Rule 4.143)}\\
&\quad \{(w,(x,y)),(B,(x,y-1))\}((0,1,0,0,0),(0,24,0,24,0);0) \mid && \text{(Rule 4.144)}\\
&\quad \{(w,(x,y)),(C,(x,y-1))\}((0,1,0,0,0),(0,24,0,24,0);0) && \text{(Rule 4.145)}\\
(A'',(x,y)) &\longrightarrow \{(d,(x,y)),(A'',(x+1,y))\}((0,1,0,0,0),24;(1,-1,0,0,0)) && \text{(Rule 4.146)}\\
&\quad \{(d,(x,y)),(A',(x,y-1))\}((1,0,1,0,0),24;(0,0,-1,1,0)) && \text{(Rule 4.147)}\\
&\quad \{(d,(x,y)),(B,(x,y-1))\}((1,0,0,0,0),(24,0,0,24,0);0) && \text{(Rule 4.148)}\\
&\quad \{(d,(x,y)),(C,(x,y-1))\}((1,0,0,0,0),(24,0,0,24,0);0) && \text{(Rule 4.149)}\\
(C,(x,y)) &\longrightarrow \{(d,(x,y)),(C,(x+1,y))\} && \\
&\quad ((0,1,0,0,0),(24,24,24,24,0);(1,-1,0,0,0)) \mid && \text{(Rule 4.150)}\\
&\quad \{(C,(x-1,y)),(d,(x,y))\} && \\
&\quad ((1,0,1,0,1),(24,24,24,24,1);(-1,1,0,0,0)) \mid && \text{(Rule 4.151)}\\
&\quad \{(d,(x,y)),(C,(x,y-1))\} && \\
&\quad ((0,1,0,1,1),(0,24,24,24,1);(0,0,1,-1,-1)) \mid && \text{(Rule 4.152)}\\
&\quad \{(d,(x,y)),(C,(x,y-1))\} && \\
&\quad ((1,0,0,1,0),(24,0,24,24,0);(0,0,1,-1,1)) \mid && \text{(Rule 4.153)}\\
&\quad \{(w,(x,y)),(D,(x,y-1))\} && \\
&\quad ((0,1,0,0,0),(0,24,0,24,1);(0,0,0,0,-1)) \mid && \text{(Rule 4.154)}\\
&\quad \{(d,(x,y))\}(0,(0,24,24,0,24);(0,0,0,0,-1)) && \text{(Rule 4.155)}\\
(B,(x,y)) &\longrightarrow \{(w,(x,y)),(B,(x+1,y))\} && \\
&\quad ((0,1,0,0,0),(24,24,24,24,0);(1,-1,0,0,0)) \mid && \text{(Rule 4.156)}\\
&\quad \{(B,(x+1,y),(w,(x,y)))\} && \\
&\quad ((1,0,1,0,1),(24,24,24,24,1);(-1,1,0,0,0)) \mid && \text{(Rule 4.157)}\\
&\quad \{(w,(x,y)),(B,(x,y-1))\} && \\
&\quad ((1,0,0,0,0),(24,0,24,24,0);(0,0,-1,1,1)) \mid && \text{(Rule 4.158)}\\
&\quad \{(w,(x,y)),(B,(x,y-1))\} && \\
&\quad ((0,1,1,0,1),(0,24,24,24,1);(0,0,-1,1,-1)) \mid && \text{(Rule 4.159)}\\
& && \text{(Rule 4.160)}
\end{aligned}
$$

$$
\Bigg\}
$$

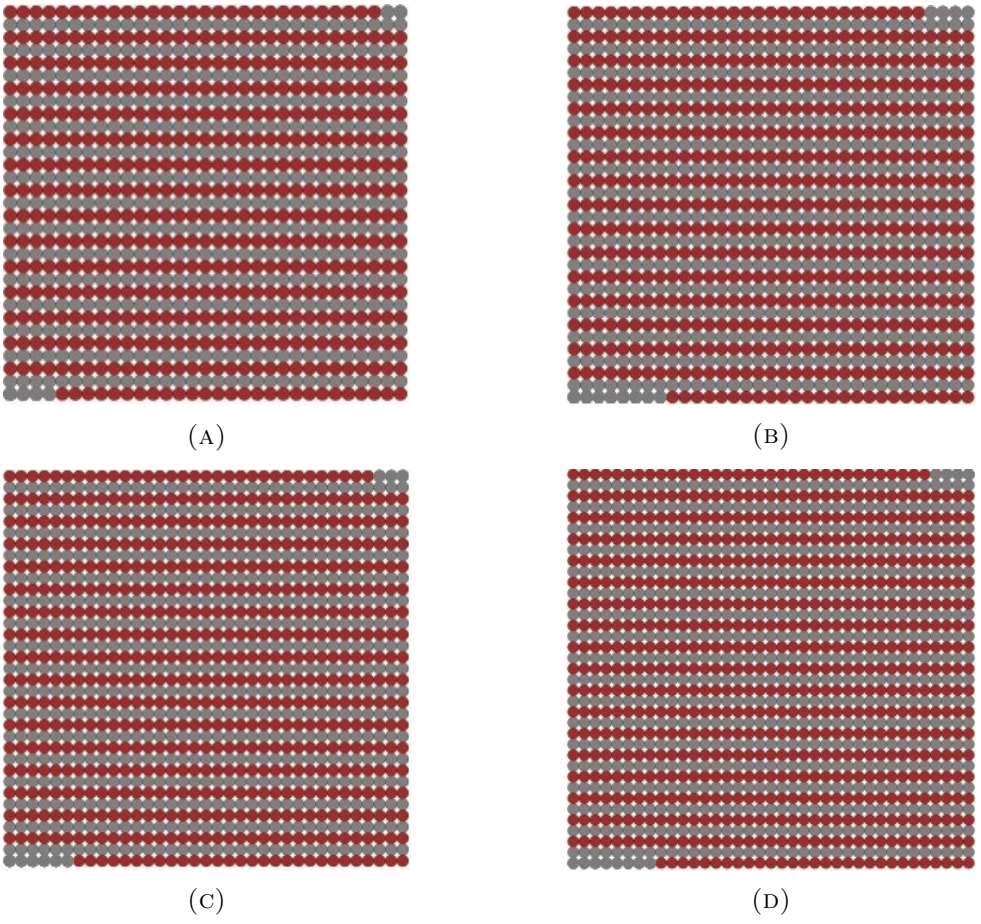
FIGURE 4.12: (A) The set of rules for $G_{\text{SameBuild}}$.

$$
R = \Bigg\{
$$

$$
\begin{array}{rcll}
 & & \{(w,(x,y)),(D,(x,y-1))\} & \\
 & & \quad((0,1,0,0,0),(0,24,0,24,1);(0,0,0,0,-1))\mid & \text{(Rule 4.161)}\\
 & & \{(d,(x,y))\}(0,(0,24,24,0,24);(0,0,0,0,-1)) & \text{(Rule 4.162)}\\
(D,(x,y)) & \longrightarrow & \{(d,(x,y)),(D,(x+1,y))\} & \\
 & & \quad((0,1,0,1,0),(24,24,24,24,0);(1,-2,1,-2,1))\mid & \text{(Rule 4.163)}\\
 & & \{(w,(x,y)),(D,(x+1,y))\} & \\
 & & \quad((0,1,0,1,1),(24,24,24,24,1);(1,-2,1,-2,-1))\mid & \text{(Rule 4.164)}\\
 & & \{(b,(x,y)),(D',(x,y-1)),(C,(x+1,y))\} & \\
 & & \quad((1,0,1,0,0),(24,0,24,0,1);(0,0,-1,1,1))\mid & \text{(Rule 4.165)}\\
 & & \{(w,(x,y)),(D',(x,y-1)),(B,(x+1,y))\} & \\
 & & \quad((1,0,1,0,1),(24,0,24,0,1);(0,0,-1,1,-1)) & \text{(Rule 4.166)}\\
(D',(x,y)) & \longrightarrow & \{(D',(x-1,y)),(b,(x,y))\} & \\
 & & \quad((1,0,0,1,0),(24,24,24,24,0);(-1,1,0,0,1))\mid & \text{(Rule 4.167)}\\
 & & \{(D',(x-1,y)),(w,(x,y))\} & \\
 & & \quad((1,0,0,1,1),(24,24,24,24,1);(-1,1,0,0,-1))\mid & \text{(Rule 4.168)}\\
 & & \{(w,(x,y)),(D'',(x,y-1)),\} & \\
 & & \quad((0,1,1,0,1),(0,24,24,24,1);(0,0,-1,1,-1))\mid & \text{(Rule 4.169)}\\
 & & \{(d,(x,y)),(D'',(x,y-1))\} & \\
 & & \quad((0,1,1,0,0),(0,24,24,24,0);(0,0,-1,1,1))\mid & \text{(Rule 4.170)}\\
 & & \{(b,(x,y)),(C,(x,y-1))\} & \\
 & & \quad((0,1,0,0,0),(0,24,0,24,0);0)\mid & \text{(Rule 4.171)}\\
 & & \{(w,(x,y)),(B,(x,y-1))\} & \\
 & & \quad((0,1,0,0,1),(0,24,0,24,1);0) & \text{(Rule 4.172)}\\
D'',(x,y)) & \longrightarrow & \{(b,(x,y)),(D'',(x-1,y))\} & \\
 & & \quad((0,1,0,1,0),(24,24,24,24,0);(1,-1,0,0,1))\mid & \text{(Rule 4.173)}\\
 & & \{(w,(x,y)),(D'',(x-1,y))\} & \\
 & & \quad((0,1,0,1,1),(24,24,24,24,1);(1,-1,0,0,-1))\mid & \text{(Rule 4.174)}\\
 & & \{(b,(x,y)),(D',(x,y-1))\} & \\
 & & \quad((1,0,1,0,0),(24,0,24,0,1);(0,0,-1,1,1))\mid & \text{(Rule 4.175)}\\
 & & \{(w,(x,y)),(D',(x,y-1))\} & \\
 & & \quad((1,0,1,0,1),(24,0,24,0,1);(0,0,-1,1,-1))\mid & \text{(Rule 4.176)}\\
 & & \{(b,(x,y)),(B,(x,y-1))\} & \\
 & & \quad((0,1,0,0,0),(0,24,0,24,0);0)\mid & \text{(Rule 4.177)}\\
 & & \{(w,(x,y)),(C,(x,y-1))\} & \\
 & & \quad((0,1,0,0,1),(0,24,0,24,1);0) & \text{(Rule 4.178)}
\end{array}
$$

$$
\Bigg\}
$$

FIGURE 4.12: (B) The set of rules for $G_{\text{SameBuild}}$.



(A) (B)

(C) (D)

FIGURE 4.13: Some of the images in $\mathcal{G}$ of $G_{\text{DiffBuild}}$.

$$
R = \Bigg\{
$$

$$
\begin{array}{lcll}
(S,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y))\}(0,0;(1,0,1,0,0)) & \text{(Rule 4.179)} \\
(A,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y))\} & \text{(Rule 4.180)} \\
 & & \quad ((1,0,1,0,0),(35,0,35,0,0);(1,0,1,0,0)) \mid & \text{(Rule 4.181)} \\
 & & \{(w,(x,y)),(B,(x,y-1))\} & \\
 & & \quad ((30,0,30,0,0),(35,0,35,35,35);(0,0,-1,1,0)) \mid & \text{(Rule 4.182)} \\
 & & \{(w,(x,y)),(A,(x+1,y))\} & \text{(Rule 4.183)} \\
 & & \quad ((30,0,30,0,0),(35,0,35,35,35);(1,0,1,1,0)) & \text{(Rule 4.184)} \\
(B,(x,y)) & \longrightarrow & \{(B,(x-1,y)),(w,(x,y))\} & \text{(Rule 4.185)} \\
 & & \quad ((1,0,0,0,0),(35,35,35,35,0);(-1,1,0,0,0)) \mid & \text{(Rule 4.186)} \\
 & & \{(w,(x,y)),(C,(x,y-1))\} & \text{(Rule 4.187)} \\
 & & \quad ((0,1,1,1,0),(0,35,35,35,0);(0,0,-1,0,0)) & \text{(Rule 4.188)} \\
(C,(x,y)) & \longrightarrow & \{(d,(x,y)),(C,(x+1,y))\} & \text{(Rule 4.189)} \\
 & & \quad ((0,1,1,1,0),((35,35,35,35,0));(1,-1,0,0,0)) \mid & \text{(Rule 4.190)} \\
 & & \{(d,(x,y)),(B,(x,y-1))\} & \text{(Rule 4.191)} \\
 & & \quad ((1,0,1,1,0),((35,0,35,35,0));(0,0,-1,0,0)) \mid & \text{(Rule 4.192)} \\
 & & \{(w,(x,y)),(C,(x+1,y))\} & \text{(Rule 4.193)} \\
 & & \quad ((0,1,0,1,0),(35,35,0,35,35);(0,-1,0,-1,1)) \mid & \text{(Rule 4.194)} \\
 & & \{(w,(x,y)),(C,(x+1,y))\} & \text{(Rule 4.195)} \\
 & & \quad ((0,1,0,0,1),(35,35,0,0,35);(0,-1,0,0,-1)) \mid & \text{(Rule 4.196)} \\
 & & \{(d,(x,y)),(C,(x+1,y))\} & \text{(Rule 4.197)} \\
 & & \quad ((0,1,0,0,0),(35,35,0,0,0);(0,-1,0,0,0)) \mid & \text{(Rule 4.198)} \\
 & & \{(d,(x,y))\}(0,0;0) & \text{(Rule 4.199)}
\end{array}
$$

$$
\Bigg\}
$$

FIGURE 4.14: The set of rules for $G_{\text{DiffBuild}}$.

Chapter 5

Bag Context Shape Grammar Interpreter

Part of this chapter is published in the Computer–Aided Design & Applications Journal, 17(3), 2020, 458–474.

5.1 Introduction

As proof of concept, the theories presented in bag context shape grammars are implemented in a prototype software called bag context shape grammar interpreter (BCSGI) which is used to generate images in a regulated manner. The main contribution of this tool is that the bag context is added to shape grammar rules to automatically control when these rules can be applied during derivation. This also helps in the generation of similar images. Another contribution is the improvement and implementation of the existing shape grammar interpretation algorithm [28].

Next, we explain how the BCSGI works in Section 5.2.

5.2 BCSGI Design

Here, we present the building blocks of the tool as follows:

5.2.1 Structure

The process of generating a gallery using the BCSGI consists of five stages. The five stages are:

1. **Initialization stage:** The rules are loaded into the rule set using the predefined format in Figure 5.1. The rule set is sorted into different lists according to the nonterminal (label) on the left hand side of each rule. The initial shape is assigned to the pictorial form and the initial bag is declared.
2. **Outer collection stage:** The nonterminals are picked from the pictorial form and a nonterminal is randomly selected.
3. **Inner collection stage:** The nonterminal selected in stage 2 is used to identify the sorted list that contains the rule(s) with the selected nonterminal on the left hand side of each rule. Then a rule is randomly picked from the identified sorted list.
4. **Derivation stage:** Checks if the bag of the selected rule in stage 3 is within the defined range and then performs rule derivation, automatically adjusting the bag. Execution is returned to stage 2 to randomly pick another nonterminal from the new pictorial form.
5. **Conversion stage:** Checks if the pictorial form consists completely of terminals. The pictorial form is then converted to a bitmap image when any shape is selected to render the image.

Next, we present the architectural flow of the stages for the generation of a finite number of similar images in the tool in Section 5.2.2.

5.2.2 Architecture

Figure 5.1 shows the architectural flow of the BCSGI showing how the stages are linked and communicate with each other.

5.2.3 Interpretation

Algorithm 1 shows the implementation of the BCSGI. The first stage of the algorithm requires that a user defines and loads the rules. A simple rule with shape whose lower left corner units are in centimetres is loaded using the following format in Listing 5.1. An example of the rule format in Listing 5.1 is shown in Listing 5.2.

```
1   RuleNo ([label], [coordinate]): ([label(s)], [coordinate(s)]) [bag]
```

Listing 5.1: A simple bag context shape grammar rule format.

```
1 ("S", (0,0),=): ("d", (0,0)), ("A", (0,1)) (0, 0; (1, 0))
2
3 ("A", (0,0)): ("d", (0,0)), ("A", (0,1)), ("B", (-1,0)) ((1, 0), (1, 2); (-1, 1))
```

Listing 5.2: A bag context shape grammar rule.

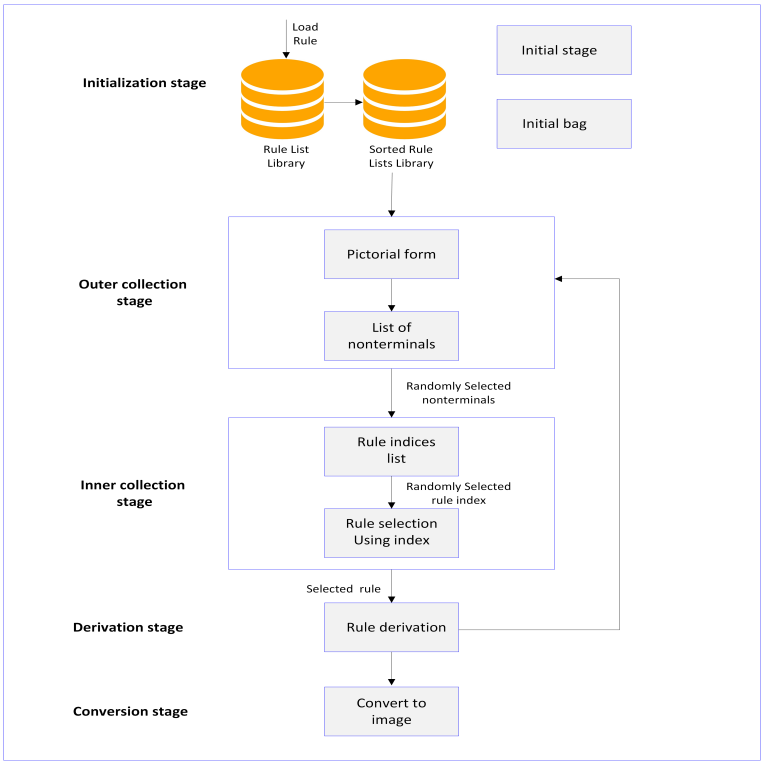


FIGURE 5.1: The architectural flow of BCSGI. The purpose of the randomization is to avoid giving any nonterminal or rule priority.

Next, we explain in detail how Algorithm 1 works.

The rules are loaded into the `ruleList` in Line 1. Line 2 sorts the `ruleList` into separate lists—`sortedruleLists`, according to the nonterminals in the left hand side of each rule in the `ruleList`. We initialise the `Bag` and the `pictorialForm` in Lines 3 and 4. Line 5 picks the `nonTerminal(s)` in the `pictorialForm` and stores the same in the `nonTerminalList()`. Line 6 randomly picks a number `i` from 1 to the length of the `nonTerminalList()`. The number is used as an index to pick a `nonTerminal` from the `nonterminalList(i)` in Line 7, then the `nonTerminal` with the index `i` is removed from the `nonTerminalList` in Line 8 in order not to pick it again because it has been picked already. Line 9 identifies and picks the list with the nonterminal which is picked in Line 8 from the `sortedruleLists()` and stores the same in the `sortedRule()` list.

Lines 10 and 11 pick the indices of all the rules in the `sortedRule` list and insert them in a list `ruleIndexes(ruleIndex)`.add, then Line 12 randomly picks a number `ruleNo` from 1 to the length of `ruleIndexes()`. Line 13 uses the `RuleNo` picked in Line 12 to select the rule with the index from the `sortedRule()` list, then, checks if the `Bag` is within the defined range. Line 14 does the derivation to the `pictorialForm`. The picked `nonTerminal` in the `pictorialForm` is replaced with its `Terminal` equivalent which also appeared on the right hand side (`RHS`) of the selected rule from the `sortedRule()` list; the remaining parts of the `RHS` of the selected rule maintain their positions. This automatically adjusts the bag in Line 15. The index is then removed from the `ruleIndexes()` list in Line 16 and Line 17 returns execution to Line 5 to pick a nonterminal from the new pictorial form.

Algorithm 5 Bag context shape grammar implementation

```
1: ruleList ← loadRules()
2: sortedruleLists ← Sort(ruleList)
3: Bag ← iniBag
4: pictorialForm ← iniShape
5: nonTerminalList ← pictorialForm().getNonterminal
6: i ← randomize(1, (nonTerminalList().max))
7: nonTerminal ← nonTerminalList(i)
8: nonTerminalList(i).remove
9: sortedRule() ← sortedruleLists.getList(nonTerminal)
10: while ruleIndex ∈ sortedRule do
11:     ruleIndexes(ruleIndex).add
12: RuleNo ← randomize(1,(ruleIndexes().max)
13: if Bag ≥ sortedRule[RuleNo].lowerLimit and Bag ≤
    sortedRule[RuleNo].upperLimit then
14:     pictorialForm ← (pictorialForm \ LHS(sortedRule[RuleNo])) ∪
    RHS(sortedRule[RuleNo])
15:     Bag.adjust
16:     ruleIndexes(RuleNo).remove
17:     pick another nonterminal from the new pictorial form at line 5
18: else if ruleIndexes().empty ≠ true then
19:     ruleIndexes(RuleNo).remove
20:     pick another rule number at line 12
21: else
22:     pick all the nonterminals from the new pictorial form at line 5
23: while pictorialForm ⊆ Terminals do
24:     pictorialForm(AnyShape).convertToBitmap
25:     return pictorialForm
```

If the `Bag` is not within the defined range as in Line 13 and the `ruleIndexes()` list is not empty as in Line 18, then, Line 19 removes the `RuleNo` from the `ruleIndexes` list because the rule with the index is not applicable at this time. Line 21 returns execution to Line 12 to pick another rule index from the `ruleIndexes` list. If the `Bag` is not within the defined range as in Line 13 and the `ruleIndexes()` list is empty, then, Line 22 returns execution to Line 5 to pick a `nonTerminal` from the `nonTerminalList()` which is extracted from the current `pictorialForm`.

A picture is formed by converting the `pictorialForm` to a bitmap in Line 24 when all the `nonTerminals` in the `pictorialForm` have been replaced with `Terminals` as in Line 23, then any shape can be used to render the image. The process is terminated at this point and Line 25 returns the image formed.

5.2.4 Implementation and Result

The BCSGI was implemented using the .Net framework Class Library (FCL). It can be run on any computer with `Windows 7` operating system or above, 2GB of *RAM* or above, 10GB of HDD or above. The framework can also run on any machine with CPU architecture of 32–bit or 64–bit.

It is designed to provide interactive and comfortable user experience, to produce images that are structurally similar but not identical in a regulated manner. It performs in a way that grammar can generate a finite number of similar but not identical images in a regulated manner. The BCSGI project file, the executable file and some example files (rules) are available on request to support designers for the implementation of shape grammars and bag context shape grammars in the formal language communities. Figure 5.4 shows some images generated using the tool.

GUI Allows for the automatic response by the click of a mouse when a rule set is loaded. The GUI is organized into two major sections, `Section One`–Rule Settings and `Section Two`–Generate, each section having its different parts.

The `Rule Settings` section as in Figure 5.2 contains the rule settings panel displays. `Parts 1` and `2` initialize the starting label and the coordinate, initial bag and the bag index. `Part 3` adds the rules to the rule list. Part 4 displays the list of rules added in `Part 3`. `Part 5` contains `Save rule`—keeps the rule ready for the derivation, `Load rule from drive`—retrieves rule as text file into the system, `save rule to drive`—extracts rule as text file to an external drive, and `clear`—wipes all the rules from the rule list.

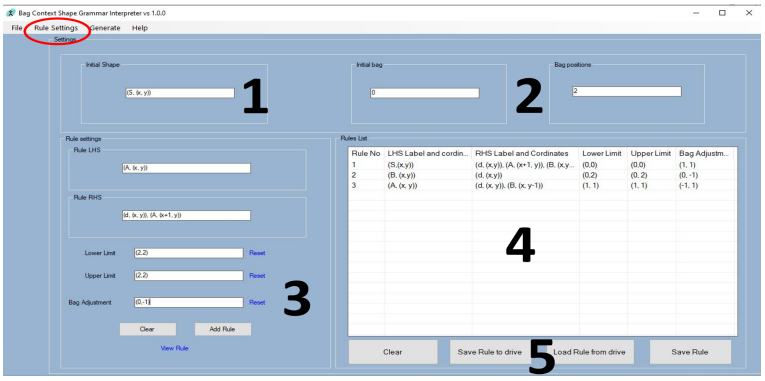


FIGURE 5.2: The BCSGI rule settings.

The `Generate` section as in Figure 5.3 contains the parts for the image generation. `Part 1` does the rule derivation which generates an image by requesting the user to select any shape of choice. `Part 2` is the plain canvas that displays the image generated. `Part 3` saves the image generated to drive while `Part 4` clears the content of `Part 2`. Then, `Parts 5` and `6` activate and show the stepwise process of the image derivation graphically.

Output Some images generated using the BCSGI are shown in Figure 5.4.

5.2.5 Performance

This tool is strictly a proof of concept to show that bag context can be added to shape grammar rules to generate images in a regulated manner. There are a few lines in the algorithm which we believe should potentially reduce the running time. For instance, during the derivation, the rules

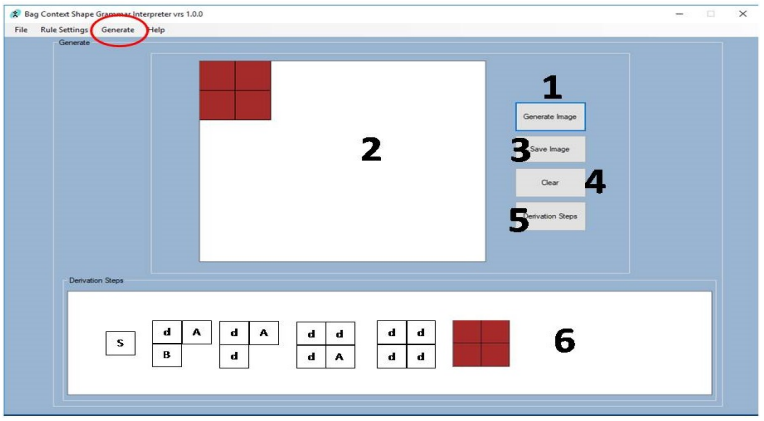
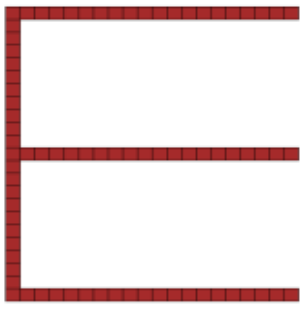
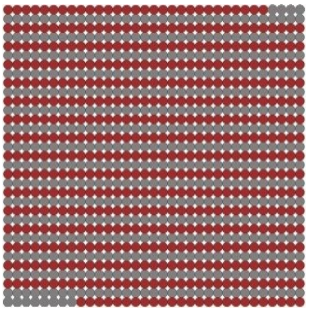


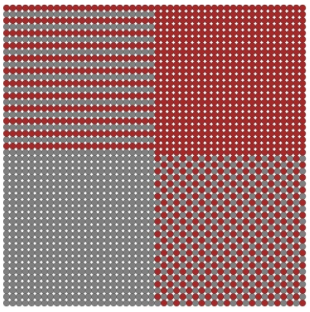
FIGURE 5.3: The BCSGI Generate Module Interface.



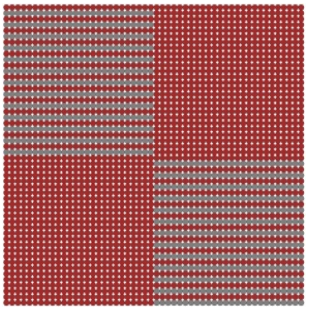
(A) An image in $\mathcal{G}(G_{\mathrm{E-bag}})$



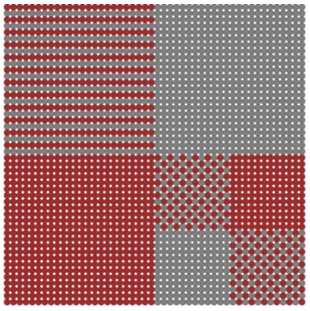
(B) An image in $\mathcal{G}(G_{\mathrm{Carpet-B}})$



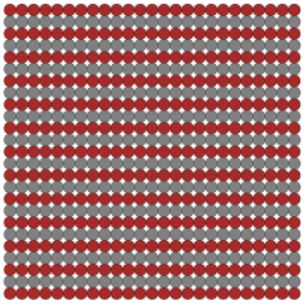
(C) An image in $\mathcal{G}(G_{\mathrm{Carpet-D}})$



(D) An image in $\mathcal{G}(G_{\mathrm{Carpet-C}})$



(E) An image in $\mathcal{G}(G_{\mathrm{Carpet-E}})$



(F) An image in $\mathcal{G}(G_{\mathrm{Carpet-A}})$

FIGURE 5.4: Some of the images generated by the BCSGI.

are selected from the `sortedRule` list using the picked nonterminal without needing to check the entire rule list from beginning to end. The selected rule number is removed from the rule indexes list in order not to pick the rule number again. The selected rule number is used to pick

a particular rule at a time without needing to check from the beginning of the sorted rule list to end. This reduces the number of a random selection of terminals and rules during the derivation (see Algorithm 1).

Therefore, the BCSGI is designed to provide interactive and comfortable user experience, to generate images that are structurally similar but not identical in a regulated and timely manner. It performs in a way that grammar can generate many similar but not identical images in a regulated manner using any shape.

Next, we summarise this part in Section 5.3.

5.3 Summary

In this part, we implemented the theories presented in BCSGs in a software called BCSGI. The tool generates images in a regulated manner using bag context added to additive shape grammar rules. The BCSGI accepts additive shape grammar rules and automatically allows when a rule should be applied during derivation. Then any shape can be used to render the image when the derivation is complete. We improved upon and implemented the existing shape grammar interpretation algorithm in [28]. We also tested the BCSGI with some bag context shape grammar rules and rendered them with different shapes, it generated the expected images. We recommend the BCSGI tool as a support to the teaching of shape grammar implementation in formal language classes or higher learning in general, a framework to support designers for the development and implementation of an improved BCSGI, etc. In the future, we will implement a more robust BCSGI tool that can run on different platforms with improved optimized performance. Next chapter checks the similarity of these images generated by BCSGI.

Chapter 6

The Similarity of Images Generated by BCSGs

This chapter is an extended version of the paper presented at the SAUPEC/RobMech/PRASA 2020 Conference

6.1 Introduction

This chapter follows from the definition of BCSGs in Chapter 3 and the generation of galleries using BCSGI in Chapter 5. The main objective of this chapter is to check if the images generated by BCSGs are similar but not identical. Given that other shape grammars including the additive shape grammars cannot avoid the generation of images that are entirely different from each other or images that some parts is too small for the human eye to see, see Figures 2.30 and 3.2. Hence, they do not consider the similarity of the images during derivation. We have demonstrated that BCSGs can generate images in a regulated manner considering the similarities between them. See Chapters 3 and 4 respectively. Then, the Spatial Colour Distribution Descriptor (SpCD) is used to check if these images generated by BCSGs are similar or not. SpCD uses a machine learning approach to extract colour information and matching of image features [65, 131].

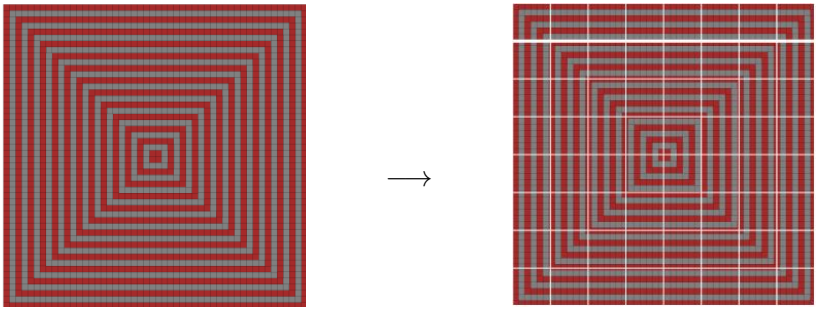
6.2 Image Similarity

Image similarity has been discussed extensively in Chapter 2. In this part, we focus on image similarity by colour [138]. Image similarity by colour is mostly determined using colour descriptor models [77, 140]. Many colour descriptor models are being used for image similarity,

depending on the type of image. One such model is the Spatial Colour Distribution Descriptor (SpCD) [130, 131]. SpCD involves two major steps, the extraction of the spatial distribution of colour information and matching. The extraction process in SpCD involves image partitioning, representative colour selection, colour space conversion, discrete cosine transform (DCT) transformation, and Zigzag scanning [65, 130]. The matching process evaluates the spatial colour information of two images and calculates the distance between them. The result of the calculated distance between two image colour information or features is the value 0 if the two images have the same colour descriptors (colour features) [65, 130]. SpCD has proven to detect the similarity of syntactically generated images with a high degree of success and accuracy [77, 109].

In this part, the image similarity measure is successfully carried out by first extracting the spatial distribution of colour information with the following steps:

1. Image partitioning: The input image is divided into 64 blocks (8×8). Figure 6.1(A) represents an input image and Figure 6.1(B) represents an output image in this stage.



(A) (B)

FIGURE 6.1: Image partitioning in SpCD.

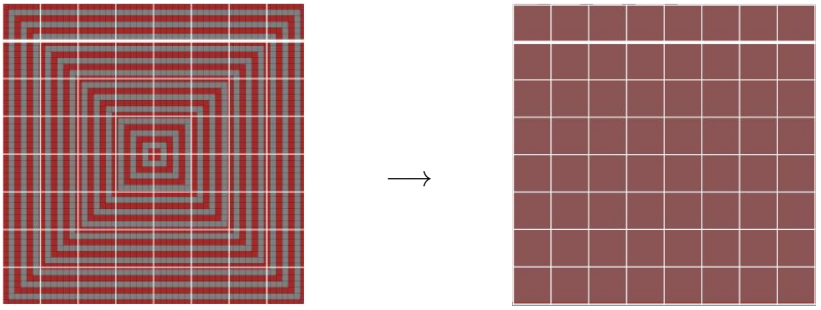
2. Representative colour selection: The average of the pixel colour in each block in Figure 6.1(B) is taken as a representative colour. Let P be the image divided into 64 blocks. Each block is denoted by $J_{m,n}$, $0 \leq m,\ n \leq 7$. J is the block and m, n represent the position of J in P. Each pixel that belongs to $J_{m,n}$ is denoted by $J(i)_{m,n}$, $0 \leq i \leq \ell$, where ℓ is the number of pixels in $J_{m,n}$. The average pixel value (μ) for each block in P is calculated thus:

$$\mu = \frac{1}{\ell} \sum_{i=0}^{\ell-1} J(i)_{m,n} \qquad \text{(Equation 6.1)}$$

Figure 6.2(A) represents the input image and Figure 6.2(B) represents the output image.

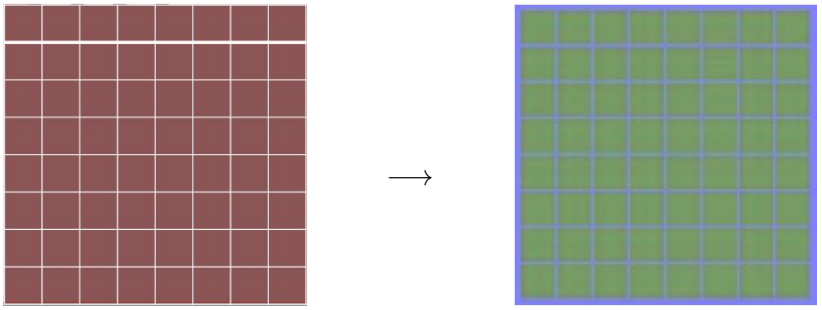
3. Colour space conversion: The RGB colour space of the image is converted to YC_bC_r. That is, the image is separated into Y the luma component, C_b, and C_r the blue-difference and red-difference chroma components. The input is the image in RGB colour space (see Figure 6.3(A)) and the output is the image in YC_bC_r colour space (see Figure 6.3(B)).

4. Discrete Cosine Transform (DCT): The YC_bC_r colour space conversion obtained in Figure 6.3(B) is transformed into three 8×8 matrices of 64 DCT coefficients each. Here, the luminance (Y), the blue chrominance (C_b), and the red chrominance (C_r) are transformed by the DCT. The input is the YC_bC_r and the output is three 8×8 matrices of 64 DCT



(A) (B)

FIGURE 6.2: Image representative colour selection in SpCD.



(A) (B)

FIGURE 6.3: Colour space conversion in SpCD.

coefficients each denoted by $DCTY$, $DCTC_b$, and $DCTC_r$. Given $P'_{\mathrm{M\times N}}$, P' is the image in YC_bC_r colour space with size 8×8 of 64 blocks, denoted by M×N. Each block is denoted by $J'_{m,n}$. J' is the block and m, n is the position of J' in $P'_{\mathrm{M\times N}}$. Then, the DCT in a 2D matrix is obtained using Equation 6.2.

$$P' = \alpha_p \alpha_q \sum_{m=0}^{7} \sum_{n=0}^{7} J'_{m,n} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N} \qquad \text{(Equation 6.2)}$$

Where p and q are defined in the range as follows: $0 \leq p \leq M-1$ where $M \in \mathbb{N}_+$ and $0 \leq q \leq N-1$ where $N \in \mathbb{N}_+$. α_p, α_q, and π are obtained as follows:

$$\alpha_p = \left\{ \begin{array}{cc} \frac{1}{\sqrt{M}}, & p = 0 \\ \sqrt{\frac{2}{M}}, & 1 \leq p \leq M-1 \end{array} \right\},$$

$$\alpha_q = \left\{ \begin{array}{cc} \frac{1}{\sqrt{N}}, & q = 0 \\ \sqrt{\frac{2}{N}}, & 1 \leq q \leq N-1 \end{array} \right\}, \text{and } \pi = \frac{22}{7} \quad [18].$$

5. Zigzag scanning: The low frequency coefficients in (4) are grouped by performing a zigzag scan with the three sets of 64 DCT coefficients, see Figure 6.4. The inputs here are the three 8×8 matrices of 64 DCT coefficients ($DCTY$, $DCTC_b$, and $DCTC_r$) obtained in (4) and the output is three zigzag scanned vectors (DY, DC_b, and DC_r).

The extraction process of SpCD is summarised in Figure 6.5.

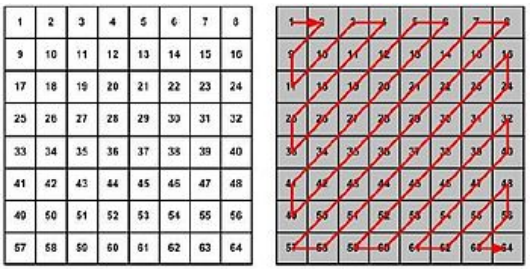


FIGURE 6.4: Representation of zigzag scan in SpCD [18].

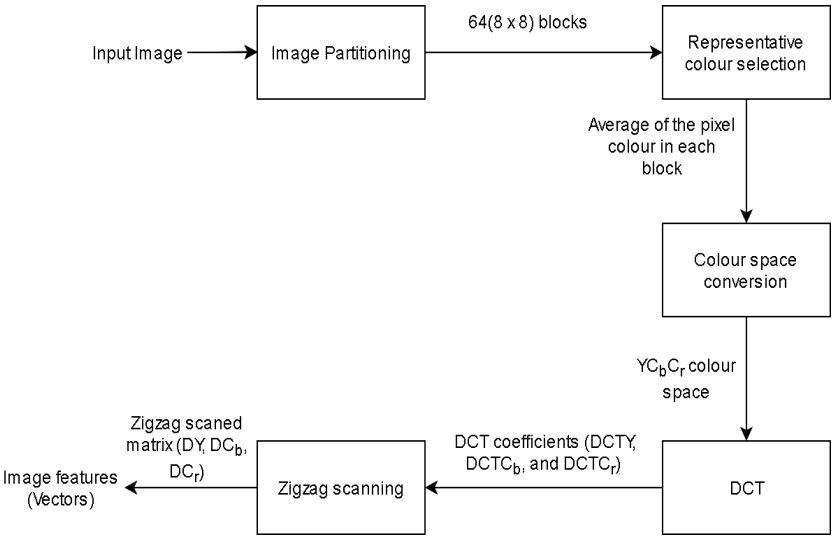


FIGURE 6.5: Image feature extraction using SpCD.

Then, the matching process evaluates the first image feature and the second image feature (target image vectors) extracted as in Figure 6.5 by calculating the distance between them. Given two sets of image descriptors (DY, DC_b, DC_r) and (DY', DC_b', DC_r'), the distance (d) between the two descriptors is calculated using (Equation 6.3).

$$d = \sqrt{\sum_{i=0}^{7} w_y(DY_i - DY_i')^2} + \sqrt{\sum_{i=0}^{7} w_b(DC_{b_i} - DC_{b_i}')^2} + \sqrt{\sum_{i=0}^{7} w_r(DC_{r_i} - DC_{r_i}')^2} \quad \text{(Equation 6.3)}$$

where i is the zigzag scanning order of the coefficients (DY, DC_b, DC_r). The variables w_y, w_b and w_r are the weighting values for the coefficients respectively. The distance d is 0 if two images have the same colour descriptors. The process is summarised in Figure 6.6.

6.2.1 Implementation

The SpCD was implemented using Img(Rummager) software. Img(Rummager) is an interactive content based image retrieval system that executes an image search based on a query image [130]. It is built with C# and requires the .Net framework 3.0 and Windows Operating System to run [130, 131]. We used Img(Rummager) for the SpCD implementation because it was used

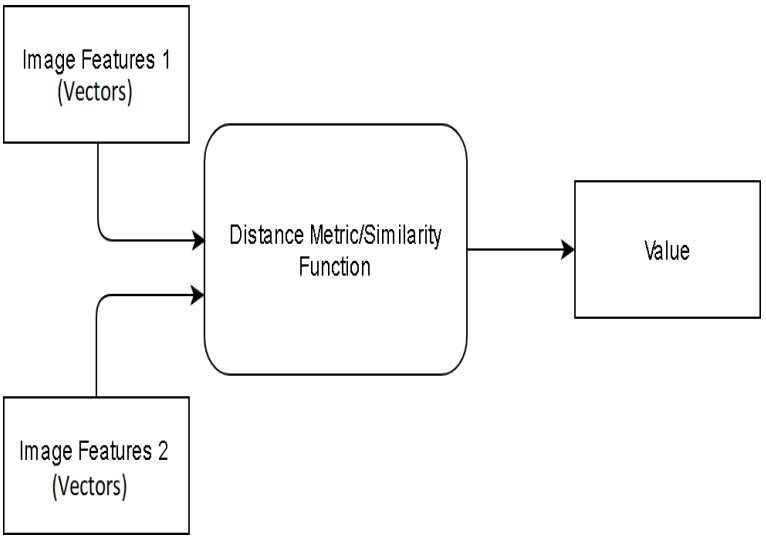


FIGURE 6.6: Image feature matching using SpCD.

successfully to evaluate the similarity of syntactically generated images in [77]. We used the BCSGI to generate five images for each gallery, see Figures 6.7–6.11. The similarity of these images in each gallery was tested using SpCD.

6.2.2 Dataset Description

The application of a BCSG rule adds at least one 1×1 square during derivation, as such the images are not always the same size when the derivation is completed. Hence, the images in Figures 6.7–6.11 which formed the dataset of this experiment were scaled to the same size. This is for presentation purposes and to ensure the images have the same statistical and scaling properties. Each gallery in Figures 6.7–6.11 was generated using different grammar.

The images are generated within some defined bounds (lower and upper limits) with a small gap in the building levels. The building level in this part is the number of shapes added upon to generate an image or a corner (part) of an image as usual. The difference between one picture and another is either in the placement of colours or the level or way at which they are built. These galleries formed the dataset for this experiment.

6.2.3 Results and Discussion

6.2.3.1 Results

The ranking according to the SpCD is such that the image with the smallest SpCD value is the most similar to the target image, while the image with the largest SpCD value is the least similar to the given image.

Tables 6.1–6.5 summarise the outcome of the SpCD similarity experiment on the images in Figures 6.7–6.11 respectively. For each given table, the first column and row represent the

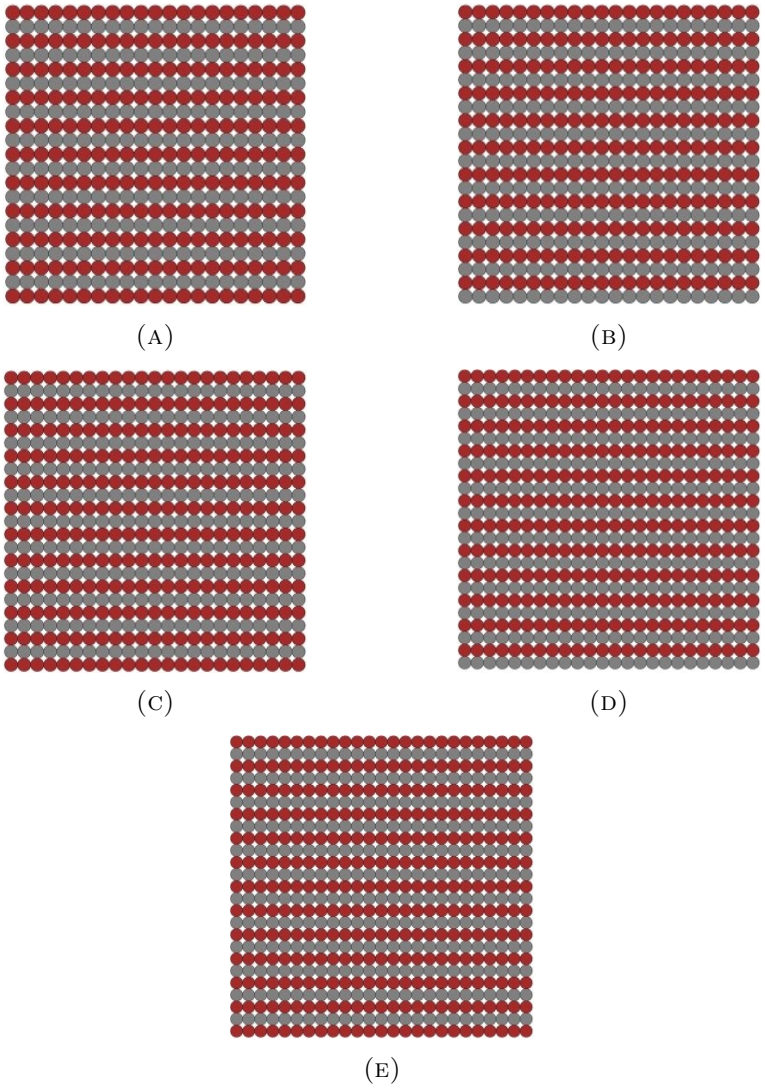


FIGURE 6.7: $G_{\text{Carpet}-\text{A}}$ same colour placement with different building levels.

images labelled (A) to (E). The image ranking is from 1 (most similar) to 5 (least similar). For instance, 2 (0.557) means 2 is the image ranking and 0.557 is the SpCD value. The experiment was conducted in two ways: Firstly we scaled the images to the same size before applying the SpCD, secondly we applied the SpCD on the original image size without scaling them.

6.2.3.2 Discussion

Consider Tables 6.1–6.5. The SpCD values clearly show that the images are similar but not identical. The images are generated with a little difference in the building level within some defined bounds, that is lower and upper limits. The difference between one image and another is either in the placement of colours or the building level, see Figures 6.7–6.11. The images in Figure 6.7 were kept at different building levels with a colour dominating a row at a time. The SpCD values showed that the images are not the same, as shown in Table 6.1.

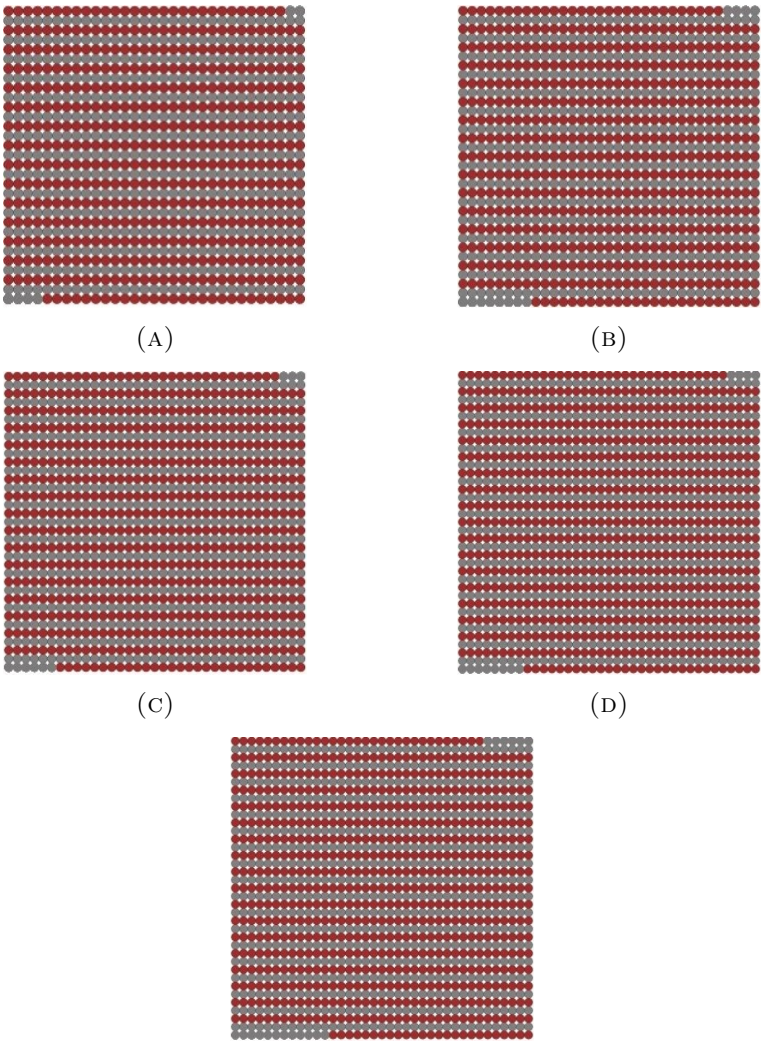


FIGURE 6.8: $G_{\text{Carpet}-\text{B}}$ different building levels with the light colour on the lower right corner twice the light colour on the upper right corner.

On the other hand, the light colour on the upper right corner of the images in Figures 6.8 were intentionally made to appear twice its length at the lower left corner of the image in Figure 6.8 with different building levels too. Also, the images in Figure 6.9 were intentionally generated with different building levels, with the light and dark colours dominating the upper right and lower left corners of the image or vice versa or one colour at both corners of the image. While the upper left corner was mirrored on the lower right corner. The images in Figure 6.10 were also generated with different building levels with different colour placement on the image parts. The SpCD values also showed that the images are similar but not identical. See Tables 6.2–6.4 respectively.

Meanwhile, the images in Figure 6.11 are generated with the same building level but different colour placement on the image parts. The colour placement in the images is such that the lower right corner consists of light and dark colours which fill the upper left and lower right corners in a way a colour is not allowed to dominate a row or column respectively, see Figure 6.11. The SpCD values clearly showed that the images are similar but not identical even though they are

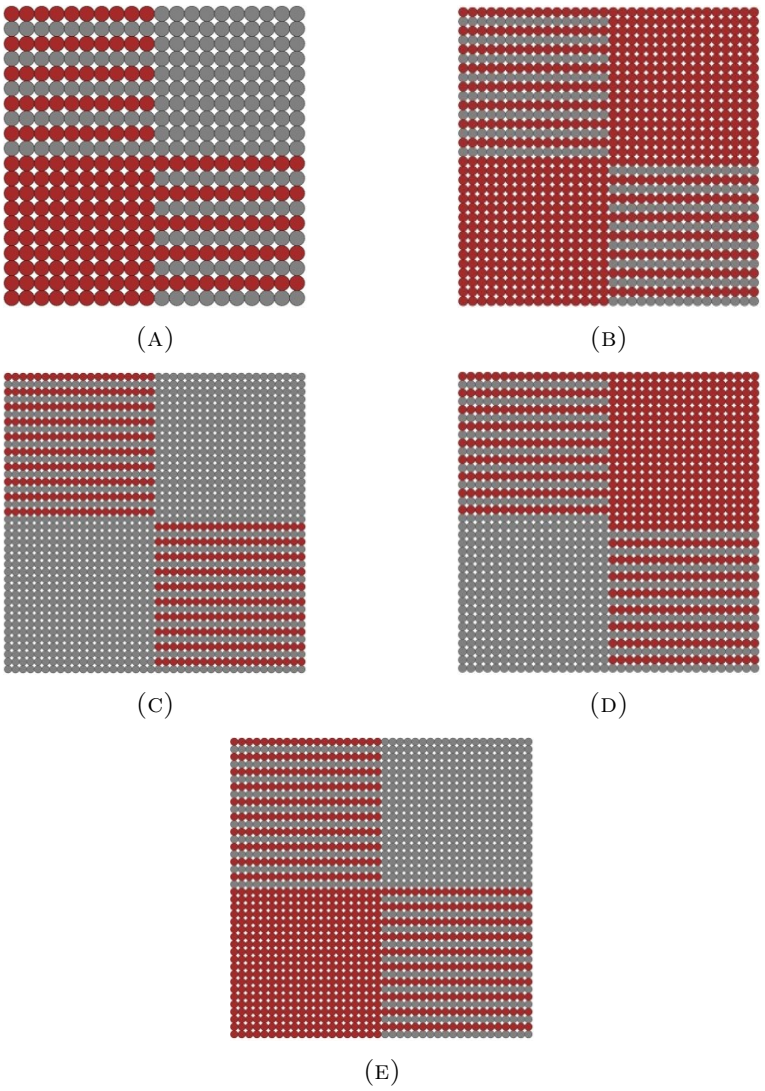


FIGURE 6.9: $G_{\text{Carpet}-\text{C}}$ different colour placement with different building levels.

TABLE 6.1: Similarity of images in Figure 6.7

1. Scaled Size					
Image	(A)	(B)	(C)	(D)	(E)
(A)	1 (0.000)	2 (9.991)	4 (11.936)	5 (13.774)	3 (10.868)
(B)	5 (9.991)	1 (0.000)	2 (5.299)	3 (6.134)	4 (8.743)
(C)	5 (11.936)	3 (5.299)	1 (0.000)	2 (4.857)	4 (9.751)
(D)	5 (13.774)	4 (6.134)	2 (4.857)	1 (0.000)	3 (5.854)
(E)	5 (10.868)	3 (8.743)	4 (9.751)	2 (5.854)	1 (0.000)
2. Original Size					
Image	(A)	(B)	(C)	(D)	(E)
(A)	1 (0.000)	2 (0.397)	3 (2.339)	4 (4.255)	5 (8.531)
(B)	2 (0.397)	1 (0.000)	3 (1.497)	4 (3.029)	5 (7.325)
(C)	4 (2.339)	3 (1.497)	1 (0.000)	2 (0.803)	5 (2.992)
(D)	5 (4.255)	4 (3.029)	2 (0.803)	1 (0.000)	3 (1.810)
(E)	5 (8.531)	4 (7. 325)	3 (2.992)	2 (1.810)	1 (0.000)

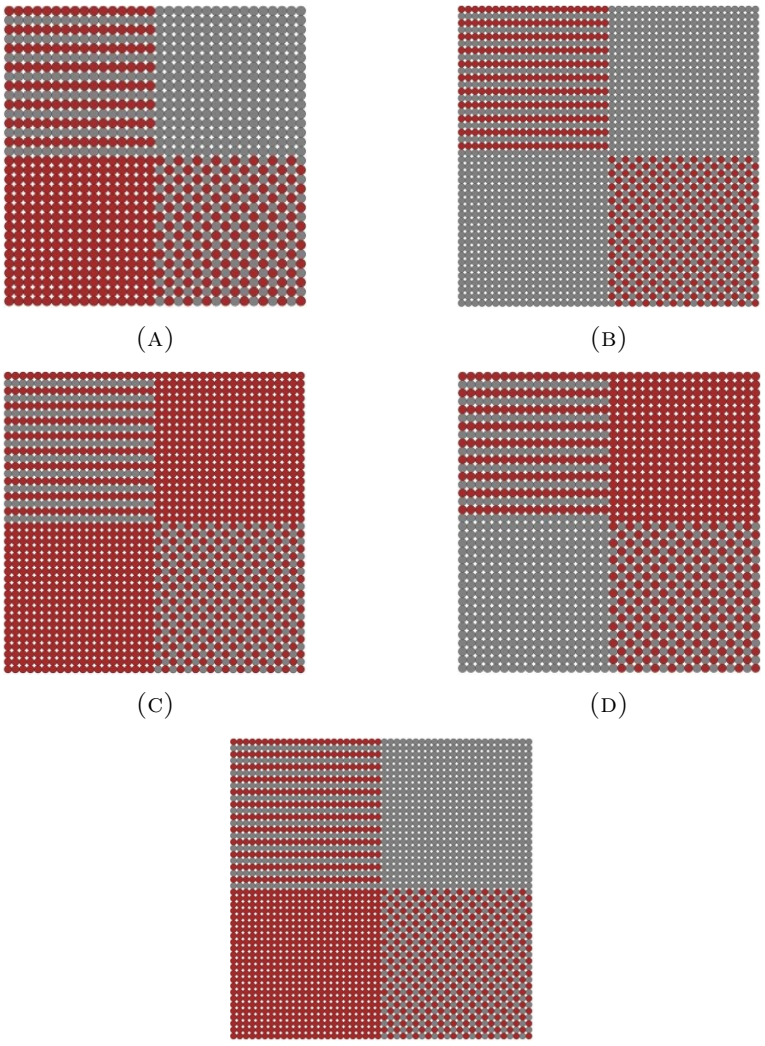


FIGURE 6.10: $G_{\text{Carpet}-\text{D}}$ different colour placement with different building levels.

TABLE 6.2: Similarity of images in Figure 6.8

1. Scaled Size					
Image	(A)	(B)	(C)	(D)	(E)
(A)	1 (0.000)	5 (2.452)	4 (2.289)	3 (2.112)	2 (1.906)
(B)	5 (2.452)	1 (0.000)	2 (0.992)	4 (2.311)	3 (1.993)
(C)	5 (2.289)	2 (0.992)	1 (0.000)	4 (1.342)	3 (1.034)
(D)	4 (2.112)	5 (2.311)	3 (1.342)	1 (0.000)	2 (0.323)
(E)	4 (1.906)	5 (1.993)	3 (1.034)	2 (0.323)	1 (0.000)
2. Original Size					
Image	(A)	(B)	(C)	(D)	(E)
(A)	1 (0.00)	2 (0.457)	3 (0.541)	4 (0.660)	5 (0.721)
(B)	2 (0.457)	1 (0.000)	3 (0.520)	4 (0.580)	5 (0.600)
(C)	5 (0.541)	2 (0.520)	1 (0.000)	3 (0.529)	4 (0.533)
(D)	5 (0.660)	4 (0.580)	3 (0.529)	1 (0.000)	2 (0.400)
(E)	5 (0.721)	4 (0.600)	3 (0.533)	2 (0.400)	1 (0.000)

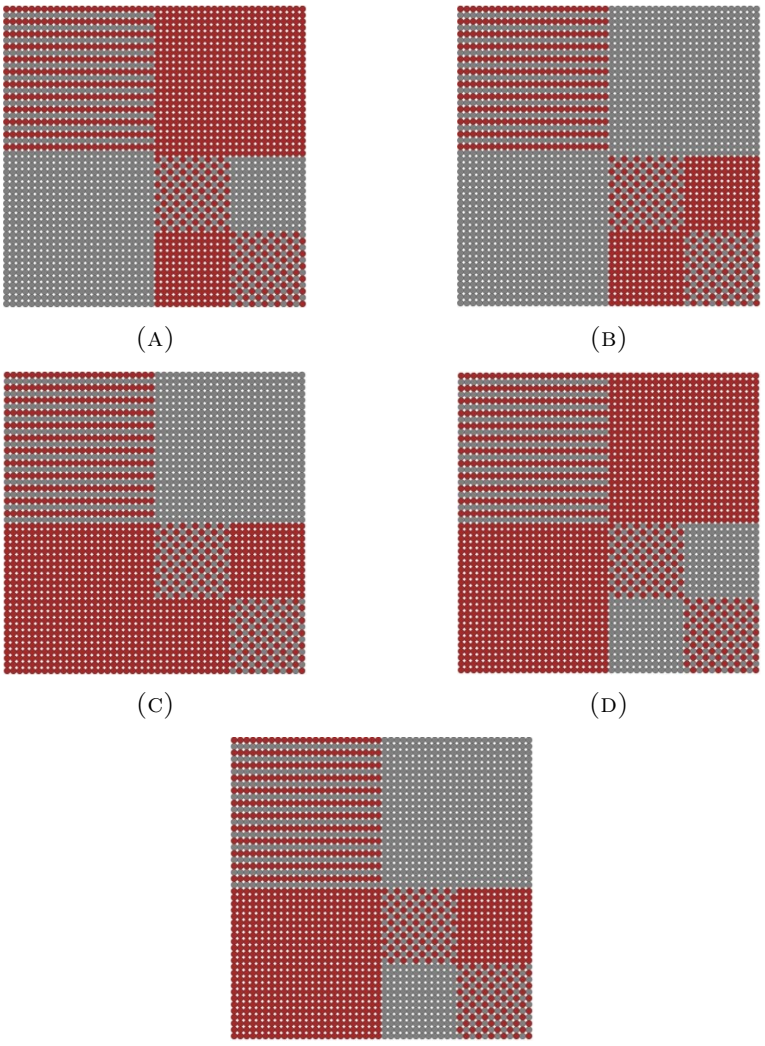


FIGURE 6.11: $G_{\text{Carpet}-\text{E}}$ different colour placement with same building levels.

TABLE 6.3: Similarity of images in Figure 6.9

1. Scaled Size					
Image	(A)	(B)	(C)	(D)	(E)
(A)	1 (0.000)	3 (41.458)	4 (52.323)	5 (70.816)	2 (7.692)
(B)	2 (41.458)	1 (0.000)	5 (73.393)	3 (44.688)	4 (45.008)
(C)	4 (52.323)	5 (73.393)	1 (0.000)	2 (44.649)	3 (50.469)
(D)	4 (70.816)	3 (44.688)	2 (44.649)	1 (0.000)	5 (73.100)
(E)	2 (7.692)	3 (45.008)	4 (50.469)	5 (73.100)	1 (0.000)
2. Original Size					
Image	(A)	(B)	(C)	(D)	(E)
(A)	1 (0.000)	4 (33.500)	3 (32.249)	5 (46.254)	2 (10.230)
(B)	5 (33.500)	1 (0.000)	4 (25.131)	2 (13.874)	3 (21.102)
(C)	5 (32.249)	3 (25.131)	1 (0.000)	2 (22.172)	4 (25.558)
(D)	5 (46.254)	2 (13.874)	3 (22.172)	1 (0.000)	4 (37.789)
(E)	2 (10.230)	3 (21.102)	4 (25.558)	5 (37.759)	1 (0.000)

TABLE 6.4: Similarity of images in Figure 6.10

1. Scaled Size					
Image	(A)	(B)	(C)	(D)	(E)
(A)	1 (0.000)	4 (51.026)	3 (36.208)	5 (69.772)	2 (12.490)
(B)	4 (51.026)	1 (0.000)	5 (68.603)	2 (48.043)	3 (49.905)
(C)	2 (36.208)	5 (68.603)	1 (0.000)	3 (38.979)	4 (42.725)
(D)	4 (49.772)	3 (48.043)	2 (38.979)	1 (0.000)	5 (75.698)
(E)	2 (12.690)	4 (49.905)	3 (42.725)	5 (75.698)	1 (0.000)
2. Original Size					
Image	(A)	(B)	(C)	(D)	(E)
(A)	1 (0.000)	3 (27.446)	4 (32.966)	2 (26.975)	5 (58.121)
(B)	2 (27.446)	1 (0.000)	4 (43.501)	3 (32.008)	5 (47.744)
(C)	3 (32.966)	4 (43.501)	1 (0.000)	2 (26.513)	5 (47.075)
(D)	3 (26.975)	4 (32.008)	2 (26.513)	1 (0.000)	5 (69.357)
(E)	4 (58.121)	3 (47.744)	2 (47.075)	5 (63.357)	1 (0.000)

TABLE 6.5: Similarity of images in Figure 6.11

1. Scaled Size					
Image	(A)	(B)	(C)	(D)	(E)
(A)	1 (0.000)	3 (54.081)	4 (76.962)	2 (48.484)	5 (80.177)
(B)	3 (54.081)	1 (0.000)	2 (47.405)	5 (81.354)	4 (55.260)
(C)	5 (76.962)	3 (47.405)	1 (0.000)	4 (53.494)	2 (11.474)
(D)	3 (48.484)	5 (81.354)	4 (53.494)	1 (0.000)	2 (45.427)
(E)	5 (80.177)	4 (55.260)	2 (11.474)	3 (45.427)	1 (0.000)
2. Original Size					
Image	(A)	(B)	(C)	(D)	(E)
(A)	1 (0.000)	3 (43.193)	4 (64.644)	2 (40.382)	5 (68.950)
(B)	3 (43.193)	1 (0.000)	2 (37.942)	5 (69.170)	4 (43.278)
(C)	5 (64.644)	3 (37.942)	1(0.000)	4 (44.559)	2 (8.580)
(D)	3 (40.382)	5 (69.170)	4 (44.559)	1 (0.000)	2 (40.225)
(E)	5 (68.950)	4 (43.278)	2 (8.580)	3 (40.225)	1(0.000)

structurally similar. See Table 6.5. All these are possible because we can control the derivation process of these images using the bag context.

One may argue that the images in Figures 6.7–6.11 are closely related or similar in the order at which they appear. Contrary to this claim, the SpCD values showed that they are not always similar in the order at which they appear. For instance, it could be argued that the image in Figure 6.7(B) is closer to the image in Figure 6.7(D), but the SpCD values showed that Figure 6.7(B) is closer to Figure 6.7(E), see Table 6.2. This could be because SpCD, among other functions, checks for the spatial distribution of the image colour information which may be different from the way humans see images.

However, a demonstration of SpCD on the images with their original size, that is without scaling the images showed that the SpCD values are lower than the SpCD values of the scaled images. It was observed also that the ranking according to SpCD values of the images without scaling with regards to a selected target image is not always the same as the ranking according to SpCD

values of the scaled images in a gallery to the selected target image of the same gallery, see Tables 6.1–6.5. This could be as a result of the loss of resolution or pixel during scaling.

6.3 Summary

In this part, we have used SpCD as a means of checking that the images generated by BCSGs are similar but not identical. While we may not strongly conclude that the SpCD alone can be used to determine the similarity of syntactically generated images, we can safely say, however, from our experiment that SpCD is suitable for determining the similarity of images generated by BCSGs. Since these BCSG generated images are similar but not identical according to our experiment, it is, therefore, safe to suggest that these images will be suitable in application areas where similar images are needed, like visual password scheme. Next, we present visual password scheme in Chapter 7.

Chapter 7

Visual Password Scheme Using BCSGs

This chapter is an extended version of the paper presented at the 19th International Conference on Intelligent Systems Design and Applications (ISDA) Conference 2019

7.1 Introduction

This chapter is motivated by the outcome of the similarity of images generated by BCSGs in Chapter 6. The goal of this chapter is to implement the similar but not identical images identified in Chapter 6 as a distractor in a prototype Visual Password Scheme (VPS). The prototype VPS is used to measure the user experience as to whether users can remember their passwords immediately after enrolment and one week later. This is to ascertain whether the similar but not identical images generated using BCSGs are good as a distractor for visual password scheme or not. The prototype visual password scheme is also built for shoulder surfing and guessing attacks resistance.

7.2 Visual Passwords Schemes

Visual Password Schemes (VPSs) involve the use of images for enrolment and verification. They are classified into recognition or cued recall [81, 166]. The former allows a user to choose a pass image from a list of distractor or decoy images during enrolment and verification. The later provides users with a hint so that they can recall their password. There are a lot of works based

on the two classes of VPSs as discussed in Section 2.8 of Chapter 2, and more are still being developed. Our kind of VPS is built as a recognition based system. This is because studies have shown that pictures are recognized with 98 percent accuracy more than words and sentences after a long time [62, 108, 141, 142]. It is intended to improve on the existing works on the aspects of guessing and shoulder surfing attacks resistance, as most recognition based VPSs are vulnerable to guessing and shoulder surfing attacks. This is demonstrated using the similar but not identical images generated using BCSGs.

7.3 The Prototype VPS

Here, we present the building blocks of the prototype VPS in Section 7.3.1

7.3.1 Structure

The prototype VPS is divided into three major parts, namely, enrolment, authentication and the admin. Each part is discussed briefly.

1. **Enrolment:** This part is where the generation of images (distractors or decoy) and the registration of password begins. The generation process is done by the BCSG interpreter that is embedded in the VPS. The BCSG interpreter models shape grammar rules into an image. A user is allowed to select the kind of predefined initial image. Every initial image has its own predefined rules which can generate infinitely different similar images. The system uses the selected initial image as a sample to generate nine distractors using the predefined rules. The nine distractors are randomly displayed on the screen. A user selects the image of choice to use as a password from the randomly displayed images. The system shuffles the images on the screen and requests that the user confirms image selection. The user is asked to supply an identification (ID) number as a means of a unique identifier just in case the user wants to change his password.

   The selected image is converted to a vector using the spatial colour distribution descriptor (SpCD) [131] and stored as an Extensible Markup Language (XML) file. The XML file is a file that defines a set of rules for encoding documents in a format that is both human and machine readable. The initial shape that was used to generate the image and the ID number are also saved.

2. **Authentication:** This part involves the verification of the image password selected by the user. Nine distractors are displayed on the screen. The system uses the saved initial shape in 1. and the appropriate predefined rules to generate the equivalent image distractors generated during enrolment. The nine images are displayed on the screen. One of the nine images must correspond exactly to the image password selected during enrolment.

   When a user selects an image, the system converts it to a vector using SpCD and matches the same with the saved vector in the XML file. Then the system shuffles the images and

requests that the user selects another image. This process is repeated three times. If the result of the three attempts are the same, that is, the vector matches the one in the XML file for the three attempts, then access is granted.

When a user does not remember his or her password, the system requests that the user supplies his ID number inorder to change the password.

3. **Admin:** This part overwrites any account. That is, it can deny or grant access at any time. Next, we present the architecture of the system in diagram (see Section 7.3.2).

7.3.2 System Architecture

In this section, the system architecture is shown in Figure 7.1. The architecture represents how each of the components of the prototype VPS communicates to each other. Next, we discuss the human computer interaction (HCI) design of the prototype VPS.

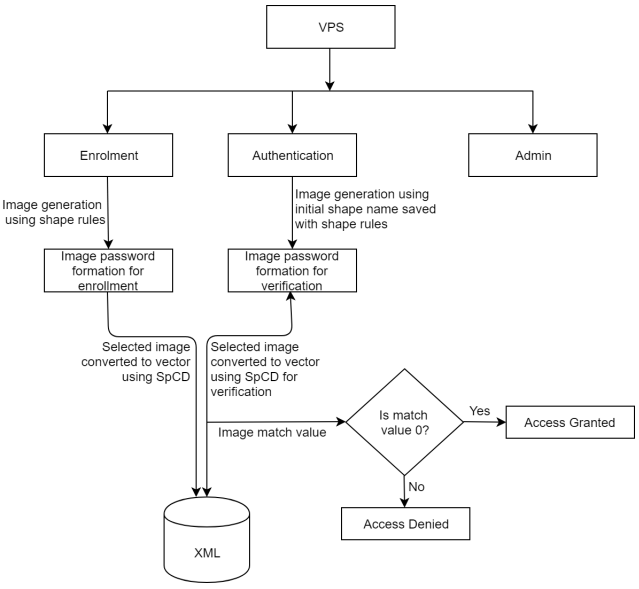


FIGURE 7.1: The VPS architecuture

7.3.3 HCI design

According to cognitive psychology, the way in which humans receive, process and store information, solve problems and acquire skill is very crucial [142]. Research in this area has shown that recognition is easier than recall, as such, humans remember images easily after a long period [142]. In order to design an interactive and an easy to use system, our type of VPS interface is designed having in mind the way in which humans receive, process and store information in order enhance recognition of image password [81].

7.3.3.1 Layout

The VPS layout is designed based on the International Telecommunication Union (ITU E.1.161) standard and recommendation for the arrangement of a digit, letters, and symbols on the telephone and other devices that can be used for gaining access to a telephone network [70]. The standard number for push buttons are ten digits, 0 to 9. The standard arrangement is 3 x 4 array as shown in Figure 7.2. The picture boxes (Image 1 to Image 9) in the 3 x 3 array are mimicked from the push buttons arrangement in Figure 7.2 leaving out the last row. Each picture box is of size 130 pt x 130 pt. The 3 x 3 array is organised in a (550 pt x 550 pt) frame.

The picture boxes are represented on the screen as shown in Figure 7.3. During rendering, the system automatically picks the size of the device screen and displays the frame at the center of the screen.

1 2 3
4 5 6
7 8 9
0

FIGURE 7.2: Push buttons arrangement

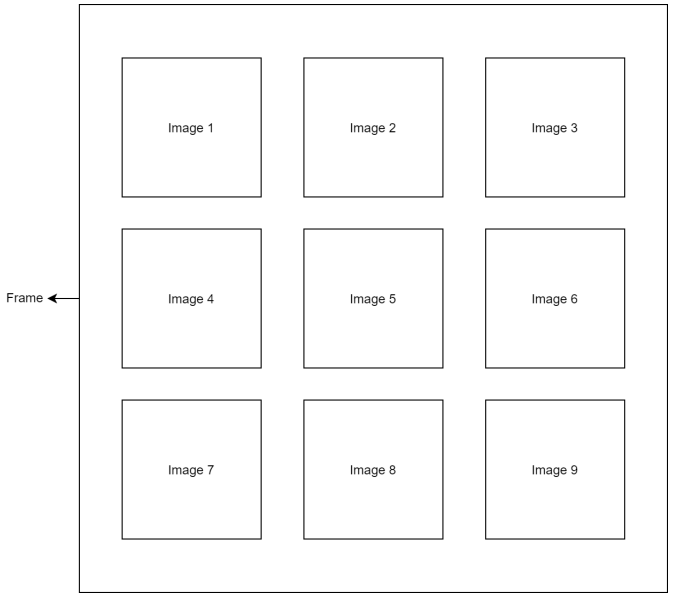


FIGURE 7.3: The VPS Layout

7.3.3.2 GUI

Designing the graphic user interface (GUI) is very crucial as the choice of colour, font type, and font size can affect users interest [6]. According to colour psychology [142], the majority of people see blue as their favourite colour. This is because most colour blind people can see the colour blue and it is associated with nature (e.g. clean water, clear sky, etc.), and more. This motivated our choice of blue as the font colour on a white background. The Sans Serif bolded

font size 14 is used to assist users with visual impairments. The Sans Serif font is a category of the font that does not use serif, small lines at the ends of characters. Which is sometimes confusing for people with visual impairment. Figures 7.4 represents the login GUI designs of the VPS.

Login GUI Design The login design is where the verification is done. It is made up of four parts. The four parts listed below are summarised in Figure 7.4.

1. Admin – denies and grants access to users at any time.
2. Login frame – holds and displays the nine distractors or decoy images for verification.
3. Create New Password – allows a user to generate and select new image password.
4. Forgot Password – requests users identification number for image password change.
5. Number of Trials – keeps count of login attempts by a user.

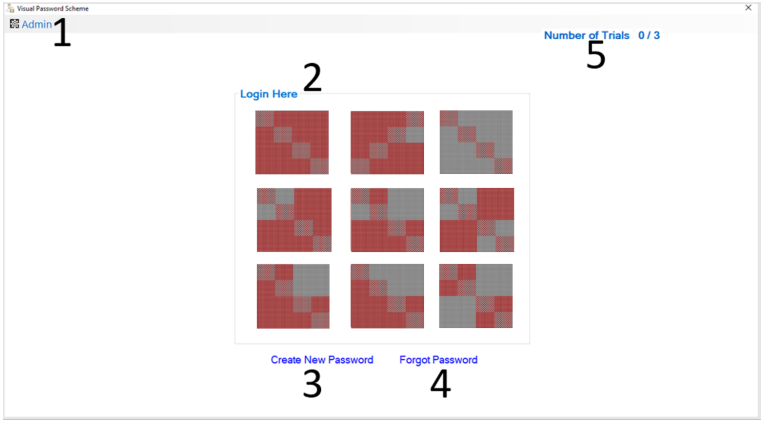


FIGURE 7.4: The VPS Login design

Enrolment GUI Design This module is made of three major parts, which are listed below:

1. Select Shape – allows a user to choose the type shape he wants to use for the image generation.
2. Image Type – allows the user to choose the type of image he wants to generate using the selected shape.
3. Image Frame – holds the randomly displayed nine distractor images.

7.3.4 Password Design

Here, password formation is done by first, partitioning the picture into 64 (8x8) equal blocks. Then, the image is transformed into a 64 coefficient matrix using SpCD. A zigzag scan is performed on the 64 coefficient matrix inorder to reduce it to a vector which is saved for verification (See Figure 7.5).

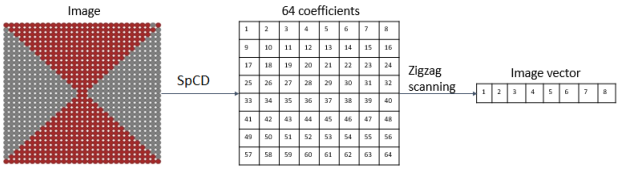


FIGURE 7.5: The VPS password formation

7.3.5 Strength Analysis

Here, we discuss the ability of the prototype VPS for shoulder surfing and guessing attacks resistance. Shoulder surfing occurs when an attacker learns a user's password by watching the user login. That is, an attacker takes up a position where the user's login details can be seen while he login. The attacker can achieve this aim by watching with the optical eyes, a special camera, video recorder or binocular, etc [168]. Guessing or Brute force attack involves trying every possible combination or trials until the correct password is found [59, 168]. Shoulder surfing and guessing attack resistance have improved significantly over the years [12, 59, 82, 121, 136, 168].

In order to prevent these attacks in our type of VPS, we implemented the similar but not identical images which act as distractors to the target image, and, to the human attacker who observes from a distance. In addition, the distractors are randomly displayed at every trial during authentication. This means that nine images are displayed on the screen, the user has to make three accurate trials for access to be granted. If the user fails at any stage of the three attempts, he will not be notified the particular stage at which he made the mistake.

Although, large password space contributes largely to the security of a system, which is the main defence against a brute force attack. However, most recognition based graphical passwords tends to have a small password space [58]. This is because picture passwords are mostly used in mobile devices, as such, password space must be limited [58]. An increased password space is not realistic for users [58].

Interestingly, it is more difficult to carry out a brute force attack against visual passwords than alphanumeric passwords [136].

7.4 Usability Study

In this section, we look into users ability to remember a password immediately after enrolment and the user's ability to remember the password at least one week later.

7.4.1 Participants

A hundred and eleven novice users, who are not familiar with the prototype VPS were trained to use the VPS. The participants were 67 male and 44 female with an average age of 25 years. These set of participants were taken because they are expert computer users who use computers for at least 4 to 5 hours a day, either for personal research or work activities.

7.4.2 Materials

Most of the computers used in testing the prototype VPS displayed nine images of equal size in a frame on the center of a screen of standard size. The nine images are randomly displayed at every trial. The system displays the images randomly to determine whether a novice could learn, remember, and select the image password successfully. The time it takes for a successful authentication is recorded by the system.

7.4.3 Procedure

The participants were divided into four groups. Each group was trained separately by the researcher. After the initial training and explanation of how the system works and how to identify an image. The researcher demonstrated the system by showing the participants how to create a password and how to log into the computer with the password.

Then, the participants were guided to create their password individually. All the participants created their password successfully. The participants were asked to use the password they created to login into the computer. At this time, the participants were not guided any more. A number of the participants were able to remember their passwords and login successfully immediately after the enrolment. The time for each successful login was recorded. The participants were interviewed to get their perception of the system.

Furthermore, one week later, a follow up was initiated. The participants were reminded to login to determine whether they could still remember their image passwords. The outcome is discussed in Section 7.4.4. The Chi-Square (X^2) test was used to analyse the association between a participants ability to login to the computer immediately and one week later. That is if those who successfully login immediately and one week later, did that by chance or they remembered their passwords and vice versa.

Hypothesis One

- H_0: The null hypothesis assumes that there is no association between a participants ability to login immediately and one week later,
- H_1: The alternative hypothesis claims that there is some association between a participants ability to login immediately and one week later.

7.4.4 Results

All participants were able to complete the process of login and authentication. The outcome is grouped and summarised in Table 7.1 also called observed data. The X^2 test is based on a test statistic that measures the divergence of the observed data from the values that would be expected or expected values (see Table 7.2). The expected value for each cell is generated from

the observed data in Table 7.1 as thus:

$$(row\_total * column\_total)/n, \qquad \text{(Equation 7.1)}$$

where n is the total number of observations in the table. If the expected value for each cell is greater than or equal to 5, then, the X^2 is good for the experiment, otherwise, another test statistic will be used. For example, the value in column two, row two of the expected values table (see Table 7.2) is calculated using Equation 7.1 to be

$$39 * 46/111 = 16.16216$$

TABLE 7.1: Observed Data

Immediately \ One week later	**No**	**Yes**	**Total**
No	24	15	39
Yes	22	50	72
Total	46	65	111

TABLE 7.2: Expected values

Immediately \ One week later	**No**	**Yes**	**Total**
No	16.16216	22.83784	39
Yes	29.83784	42.16216	72
Total	46	65	111

From Table 7.1, 24 participants are not able to remember their password immediately and one week later, 22 remembered their passwords immediately and forgot it one week later. 15 participants could not remember their password immediately but remembered it one week later, while 50 participants were able to login immediately and also one week later. From the results, there are 72 (64.9%) of the total number of participants who were able to login immediately while 39 (35.1%) could not log in immediately.

The results suggest that the majority of the participants were able to log into the system immediately. There were 65 (58.6%) of the total number of participants who were able to log into the system after one week successfully and there were 46 (41.4%) of the total number who were not able to log into the system after one week. There was a decrease in the total number of participants who were able to log into the system immediately compared to the total number that logged into the system after one week.

The X^2 test statistic is computed using Equation 7.2.

$$X^2 = \sum_{i}^{k} \frac{(O_i - E_i)^2}{E_i} \qquad \text{(Equation 7.2)}$$

where the observed value for each cell in Table 7.1 is denoted by O_i, and the expected value for each cell in Table 7.2 is denoted by E_i. The variable i is the specific cell in the table and k represents the total number of cells without the total column or row.

For example, using the Equation 7.2, we obtain the X^2 test statistic as thus

$$
\begin{aligned}
X^2 &= \frac{(24-16.16216)^2}{16.16216} + \frac{(15-22.83784)^2}{22.83784} + \frac{(22-29.83784)^2}{29.83784} + \frac{(50-42.16216)^2}{42.16216} \\
&= 3.80096 + 2.86991 + 2.05885 + 1.45703 \\
&= 10.18675
\end{aligned}
$$

Then, we proceed to calculate the probability value (p-value) of the X^2 test statistic. The p-value helps to support the claim as to whether to accept or reject the (H_0). The p-value is also evidence against the H_0. Therefore, H_0 is rejected if the p-value is ≤ 0.05. According to the experiment, the p-value = 0.00156 using the z table.

7.4.5 Discussion

The results of the X^2 test statistic show that there is a statistically significant ($X^2 = 10.18675$, p-value=0.00156) relationship or association between a participant's ability to log into the system immediately after enrolment and one week later. We, therefore, reject the H_0 and accept the alternative hypothesis H_1. We say that there is an association between the two variables. That is to say, those who remembered their passwords immediately and one week later was not by chance.

The odds (odds ratios[1]) of a successful login immediately are 3.64 times the odds of an unsuccessful login. The odds ratio is significant (p-value=0.002) with a confidence interval[2] of (1.61, 8.23).

7.5 Performance Analysis

We tested the prototype VPS performance in terms of its ability to produce the right result at the shortest possible time. We recorded the times it took for 20 successful logins on the machine the application was built and arrived at an average of 12.6 seconds and standard deviation of 3.2671. Then we compared the times to the average of 20.7 seconds of the 72 successful logins on the different machines with a standard deviation of 8.2619.

The essence is to know if the performance of the system is machine bias. That is if the performance in terms of time is dependent on the machine or not. The Independent Samples t test is used to compare the means of the two independent group of times (seconds) to determine whether there is statistical evidence that the associated time means are significantly different.

Hypothesis Two

- H_0: The null hypothesis assumes that the time means are not significally different

[1] The measure of an association between an exposure and an outcome.
[2] Used to estimate the precision of the odds ratio.

- H_1: The alternative hypothesis assumes that there is a signficant difference in the time means.

Therefore, H_0 is rejected if the p-value is ≤ 0.05.

7.5.1 Result

The independent t test statistic is calculated using the Equation 7.3.

$$t = \frac{\bar{X_1} - \bar{X_2} - (\mu_1 - \mu_2)}{s_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \quad s_p = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}} \qquad \text{(Equation 7.3)}$$

Where the variables $\bar{X_1}$ = mean of sample 1, $\bar{X_2}$ = mean of sample 2, $\mu_1 - \mu_2$ = difference in the two population means, s_1^2 = sample variance of sample 1, s_2^2 = sample variance of sample 2, n_1 = size of sample 1, and n_2 =size of sample 2.

Note: Sample 1 is for the main machine, sample 2 is for different machines. The experiment performed using the Equation 7.3 showed that the independent t test statistic is -4.3195 and the p-value = 0.000004.

7.5.2 Discussion

The results of the t test statistic show that t = -4.3195 and p-value = 0.000004 using the t table, this implies that, there is a statistically significant difference between the time means. We reject the null hypotheses H_0 and accept the alternative hypotheses H_1. This simply means that the run time of the prototype VPS can be affected by the machine used.

7.6 Summary

In this chapter, we implemented the prototype VPS using the BCSGs generated images. The similar but not identical images which act as distractors were randomly displayed to improve shoulder surfing and guessing attacks resistance. The prototype VPS was tested for the human usability, which showed that 72 (64.9%) out of 111 users were able to remember their image password immediately after login. While 65 (58.6%) out of 111 user were able to remember their image password one week after login. The results of the X^2 test statistic showed that there is a statistically significant (X^2 = 10.18675, p-value=0.00156) relationship or association between a participant's ability to log into the system immediately after enrolment and one week later. This suggests that those who remembered their passwords immediately and one week later was not by chance.

The performance of the prototype VPS in terms of run time was tested. It is also observed that the prototype VPS performance can be affected by the type of machine used. Next, is the conclusion in Chapter 8.

Chapter 8

Conclusion And Future Work

8.1 Conclusion

In recent times, similar image generation has become relevant as the need for its use in the application areas like visual password schemes as distractors have increased. A distractor image is a similar image in a visual password scheme that diverts the attention of a user from the selected image during login or incorrect image choices that looks similar to the selected image [109]. Research has shown that picture grammars such as shape grammars can generate similar images [52, 76]. Most shape grammar systems fail to generate only similar images in a controlled manner. For a shape grammar to generate similar images, it requires users to generate different images and manually select the similar images. Therefore, it is difficult to know if the next image to be generated will be similar to a selected image. Existing researches that use shape grammars for the generation of images cannot regulate how similar images can be generated in a controlled manner. This is why these grammars generate images that are entirely different from each other or images that some part is too small for the human eye to see.

To address some of these problems identified above, there is a need for an additive shape grammar class whose rules can be controlled during derivation and can allow any shape to be used to implement the images during rendering after derivation. These shape grammar rules are controlled in a way the difference between the generated images or subimages can also be controlled during derivation. This research used bag context shape grammars to generate similar images and has made the following contributions:

1. Defined the theoretical concepts and implementation of the notion of a new additive shape grammar systems, that allow any shape to be used to implement images during rendering and after a derivation. Since the additive shape grammars alone cannot generate images in a regulated manner, the need for bag context being added to the additive shape grammar rules (BCSGs) became necessary.
2. Defined the theoretical concepts and implementation of the notion of BCSGs for the generation of images in a regulated manner and proved that every permitting puzzle grammar

can be converted to a bag context shape grammar. Then we demonstrated the conversion process with examples. We then considered a set of images and demonstrated that bag context shape grammars can generate a set of images with fewer variables and rules.

3. Determined different methods to use BCSGs to make similar images. Such as keeping the building level of images to be the same while the colour placement is different and keeping the building level of images to be different while the colour placement is also different, during derivation by limiting the bounds (lower and upper limits) of the rules.

4. Implemented the theory in BCSGs into a prototype software called BCSGI. This tool was used to generate all the images in this research and can support the teaching of shape grammar or BCSG implementation in formal language classes or higher learning in general. It can also be a framework for researchers or developers who wish to contribute new ideas in the area of similar image generation or improve upon the tool for similar image generation.

5. Choose and implemented SpCD, a content based image retrieval method to mathematically check the similarity of images generated by BCSGs. SpCD proved to be good for checking the similarity of syntactically generated images. Therefore, this research recommends SpCD to be used for checking the similarity of syntactically generated images, particularly BCSGs.

6. Finally, implemented the similar images generated by bag context shape grammars in a prototype visual password scheme as distractors.

We, therefore, recommend the approach of BCSGs to the application areas where the generation of similar images are needed. In the development of new grammars or improvement of existing grammars. BCSGs can also be useful to other domain areas such as architectural design, engineering design, product design and decorative art.

8.2 Future Work

When a PBPzG is converted to a BCSG using the Lemma 3.11, the resultant BCSG has a bag position for each variable of the PBPzG. By inspecting the BCSG, one can often reduce the number of bag positions. The question is whether there is a general rule for reducing bag positions. Improvement and implementation of a more robust BCSGI that can run on different platforms with improved performance. This is because the BCSGI developed in this research is strictly a proof of concept. The implementation of the prototype visual password scheme is into a robust software tool.

Other developments in additive grammar models with permitting features such as cooperating context free array grammar systems [151], permitting features in P systems generating picture arrays [152], array P systems with permitting features [153], cooperating basic puzzle grammars systems [154], etc remain open for further research with regard to BCSGs.

Bibliography

[1] Agarwal, M., and Cagan, J. A blend of different tastes: The language of coffeemakers. *Environment and Planning B: Planning and Design 25*, 2 (1998), 205–226.

[2] Ahmad, S. *A Framework for Strategic Style Change Using Goal Driven Grammar Transformations*. PhD thesis, The University of Strathclyde, 2009.

[3] Ahmad, S., and Chase, S. Grammar representations to facilitate style innovation: An example from mobile phone design. In *Communicating Space(s)-Proceedings of the 24th eCAADe Conference, Volos, Greece* (2006), pp. 320–323.

[4] Ahmad, S., and Chase, S. Transforming grammars for goal driven style innovation. In *Predicting the future, Proceedings of the 25th Conference on Education in Computer Aided Architectural Design in Europe (eCAADe), Frankfurt am Main, Germany* (2007), pp. 879–886.

[5] Al-Kazzaz, D. A., and Bridges, A. H. A framework for adaptation in shape grammars. *Design Studies 33*, 4 (2012), 342–356.

[6] Alan, D., Finaly, J., Abowd, G. D., and Russel, B. *Human Computer Interaction*. Pearson Prentice Hall, 2004.

[7] Aliaga, D., Rosen, P., and Bekins, D. Style grammars for interactive visualization of architecture. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (2007), 786–797.

[8] Anil, K. J., and Aditya, V. Image retrieval using color and shape. *Pattern Recognition 29*, 8 (1996), 1233–1244.

[9] Anthony, C., and Robert, D. *Foundation Maths*, sixth ed. Pearson Education Limited, Edinburgh Gate, United Kingdom, 2016.

[10] Arida, S. Volkswagen Beetle Grammar. Computational design, Final MSc project, Massachusetts Institute of Technology (MIT), 2002.

[11] Aurich, V., and Weule, J. Non-linear Gaussian filters performing edge preserving diffusion. In *Mustererkennung 1995*. Springer, 1995, pp. 538–545.

[12] Awais, A., Muhammad, A., Kashif, H. M., and Ramzan, T. Secure graphical password techniques against shoulder surfing and camera based attacks. In *International*

Journal of Computer Network and Information Security, November 2016 (2016), IEEE, pp. 11–18.

[13] Baeze-Yates, R., and Valiente, G. An image similarity measure based on graph matching. IEEE String Processing and Information Retrieval, pp. 28–38.

[14] Baker, B. S. Tree transducers and tree languages. *Information and Control 37*, 3 (1978), 241–266.

[15] Barnsley, M. F. *Fractals everywhere*. Academic press, 2014.

[16] Beirão, J., and Duarte, J. Urban grammars: Towards flexible urban design. In *Proceedings of the 23rd International Conference on Education in Computer Aided Architectural Design in Europe (eCAADe)* (2005), pp. 491–500.

[17] Canny, J. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6 (1986), 679–698.

[18] Carles, V. R. Image-Based query by example using MPEG-7 visual descriptors. Master's thesis, Universitat Politècnica De Catalunya, Barcelona, 2010.

[19] Carson, C., Thomas, M., Belongie, S., Hellerstein, J. M., and Malik, J. Blobworld: A system for region-based image indexing and retrieval. In *International Conference on Advances in Visual Information Systems* (1999), Springer, pp. 509–517.

[20] Chase, S. C., and Liew, P. A framework for redesign using FBS models and grammar adaptation. In *Computer Aided Architectural Design Futures 2001*. Springer, 2001, pp. 467–477.

[21] Chin, R. C. *Product grammar: Construction and exploring solution spaces*. PhD thesis, Massachusetts Institute of Technology, 2004.

[22] Chomsky, N. Three models for the description of language. *IRE Transactions on Information Theory 2*, 3 (1956), 113–124.

[23] Chomsky, N. On certain formal properties of grammars. *Information and Control 2*, 2 (1959), 137–167.

[24] Chomsky, N. Formal properties of grammars, Handbook of Mathematical Pychology. *Wiley & Sons, New York 2* (1963), 232–418.

[25] Chomsky, N. Constraints on rules of grammars. *Linguistic Analysis 2* (1976), 303–351.

[26] Commission, U. F. T., et al. Consumer sentinel network data book for January–December 2014, 2015.

[27] Correia, R. C., Duarte, J. P., and Leitão, A. M. Malag: A discursive grammar interpreter for the online generation of mass customized housing. In *Proceedings of the 4th International Conference on Design Computing and Cognition* (2010).

[28] Crumley, Z., Marais, P., and Gain, J. Voxel-space shape grammars. In *WSCG 2012: 20th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Plzen, Czech Republic, 25 - 28 June* (2012), pp. 1–10.

[29] Daniel, G. A., Paul, A. R., and Daniel, R. B. Style grammars for interactive visualization of architecture. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS 13*, 4 (2007), 786–797.

[30] Dassow, J. Grammatical picture generation. *Retrieved from http://theo.cs.uni-magdeburg.de/lehre06w/picgen/grampicgen-text2.pdf Accessed 10 Novermber, 2016* (2006).

[31] Del Bimbo, A., Mugnaini, M., Pala, P., and Turco, F. Visual querying by color perceptive regions. *Pattern Recognition 31*, 9 (1998), 1241–1253.

[32] Drewes, F. Language theoretic and algorithmic properties of d-dimensional collages and patterns in a grid. *Journal of Computer and System Sciences 53* (1996), 33–60.

[33] Drewes, F. Tree-based picture generation. *Theoretical Computer Science 246*, 1 (2000), 1–51.

[34] Drewes, F. *Grammatical Picture Generation: A Tree-Based Approach (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc., 2006.

[35] Drewes, F., du Toit, C., Ewert, S., van der Merwe, B., and van der Walt, A. P. Bag context tree grammars. In *Proceedings of the 10th International Conference, DLT 2006, Santa Barbrara, CA* (June 26-29, 2006), Springer, pp. 226–237.

[36] Drewes, F., Ewert, S., Högberg, J., van der Walt, B., and van der Walt, A. Random context tree grammars and tree transducers. *South African Computer Journal 34* (2005), 11–25.

[37] Duarte, J. P. A discursive grammar for customizing mass housing: the case of Siza's houses at Malagueira. *Automation in Construction 14*, 2 (2005), 265–275.

[38] Ducla, S. BMW grammar. Computational design final project. MIT, 2002.

[39] Eakins, J., and Graham, M. Content-based image retrieval, October 1999. University of Northumbria at Newcastle URL: http://www.jisc.ac.uk/uploaded_documents/jtap-039.doc. Last Accessed on 01/04/2019.

[40] Ertelt, C., and Shea, K. Shape grammar implementation for machining planning. In *Proceedings of the 4th International Conference of Design Computing and Cognition* (2010).

[41] Ewert, S. *Random Context Picture Grammars*. PhD thesis, Stellenbosch University, Stellenbosch, 1999.

[42] Ewert, S. Random context picture grammars: The state of the art. *Manipulation of Graphs, Algebras and Pictures. Essays Dedicated to Hans-Jörg Kreowski on the Occasion of His 60th Birthday* (2009), 135–147.

[43] Ewert, S., Jingili, N., Mpota, L., and Sanders, I. Bag context picture grammars. *Journal of Computer Languages* (2019). doi:https://doi.org/10.1016/j.cola.2019.04.001.

[44] Ewert, S., and van der Walt, A. Generating pictures with random context. In *Proceedings of the South African Mathematical Society Congress, Bellville, South Africa. University of the Western Cape* (1996), pp. 143–148.

[45] Ewert, S., and van der Walt, A. Generating pictures using random forbidding context. *International Journal of Pattern Recognition and Artificial Intelligence 12*, 7 (1998), 939–950.

[46] Ewert, S., and van der Walt, A. Permission is necessary when generating pictures. In *Proceedings of the 13th Annual MSc and PhD Conference in Computer Science, Stellenbosch, South Africa* (1998), University of Stellenbosch, pp. 18–26.

[47] Fisher, R. *Statistical Methods for Research Workers*, vol. 13. Hafner, 1958.

[48] Flemming, U. More than the sum of parts: the grammar of Queen Anne houses. *Environment and Planning B: Planning and Design 14*, 3 (1987), 323–350.

[49] Frank, D., Habel, A., Kreowski, H.-J., and Taubenberger, S. Generating self-affine fractals by collage grammars. *Theoretical Computer Science 145*, 1&2 (1995), 159–187.

[50] Frank, D., Kreowski, H.-J., and Lapoire, D. Criteria to disprove context-freeness of collage languages. In *Proceedings of the 11th International Symposium on Fundamentals of Computation Theory* (September 1997), Krakow, Poland, pp. 1–3.

[51] Galloway, M. M. Texture analysis using gray level run lengths. *Computer Graphics and Image Processing 4*, 2 (1975), 172–179.

[52] George, S. *Shape: Talking about seeing and doing*. MIT Press, 2006.

[53] Gevers, T., and Smeulders, A. W. Pictoseek: Combining color and shape invariant features for image retrieval. *IEEE transactions on Image Processing 9*, 1 (2000), 102–119.

[54] Gode, C., and Ganar, A. Image retrieval by using colour, texture and shape features. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering 3*, 4 (2014), 8552–8559.

[55] Goswami, D., and Krishna, K. V. Lecture notes in formal languages and automata theory, November 2010. URL: https://www.iitg.ac.in/dgoswami/Flat-Notes.pdf. Last Accessed on 05/03/2019.

[56] Habel, A., and Kreowski, H.-J. Characteristics of graph languages generated by edge replacement. *Theoretical Computer Science 51* (1987), 81–115.

[57] Habel, A., Kreowski, H.-J., and Taubenberger, S. Collages and patterns generated by hyperedge replacement. *Languages of Design 1* (1993), 125–145.

[58] Haichang, G., Xiyang, L., Sidong, W., Honggang, L., and Ruyi, D. Design and analysis of a graphical password scheme. In *Fourth International Conference on Innovative Computing, Information and Control* (2009), IEEE, pp. 675–678.

[59] Haichang, G., Zhongjie, R., Xiuling, C., Liu, X., and Uwe, A. A new graphical password scheme resistant to shoulder-surfing. In *2010 International Conference on Cyberworlds* (2010), IEEE, pp. 18–23.

[60] Haralick, R. M., Shanmugam, K., et al. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 6 (1973), 610–621.

[61] Hassanat, A. B. Visual passwords using automatic lip reading. *arXiv preprint arXiv:1409.0924* (2014), 218–231.

[62] Hazamy, A. A. *Influence of Pictures on Word Recognition.* PhD thesis, Electronic Theses and Dissertations, Georgia Southern University, 2009.

[63] Herbert, T., Sanders, I., and Mills, G. African shape grammar: A language of linear Ndebele homesteads. *Environment and Planning B: Planning and Design 21*, 4 (1994), 453–476.

[64] Hiremath, P., and Pujari, J. Content based image retrieval using color, texture and shape features. In *Advanced Computing and Communications, 2007. ADCOM 2007. International Conference on* (2007), IEEE, pp. 780–784.

[65] Ho, Y. L., Ho, K. L., and Yeong, H. H. Spatial color descriptor for image retrieval and video segmentation. *IEEE Transactions on Multimedia 5*, 3 (2003), 358–367.

[66] Hoisl, F., and Shea, K. An interactive, visual approach to developing and applying parametric three-dimensional spatial grammars. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 25*, 04 (2011), 333–356.

[67] Hopcorft, J., Motwani, R., and Ullman, J. D. *Introduction to Automata Theory, Languages, and Computation.* Pearson Addison Wesley, 2006.

[68] Huijsmans, M. S. L. D. N., and Denteneer, D. Content based image retrieval: Klt, projections, or templates. *First International Workshop on Image Databases and Multimedia, Netherlands 1* (1996), 27–34.

[69] Iestyn, J., Chris, E., and George, S. Shapes, structures and shape grammar implementation. *Computer-Aided Design 111* (2019), 80–92. https://doi.org/10.1016/j.cad.2019.02.001.

[70] International-Telecommunication-Union. Series E: Overall network operation, telephone service, service operation and human factors. *Telecommunication Standardization Sector of ITU* (2004), 1–14.

[71] Isawasan, P., Muniyandi, R., Venkat, I., and Subramanian, K. Array-rewriting P systems with basic puzzle grammar rules and permitting features. In *International Conference on Membrane Computing, Cham, Switzerland* (2017), Springer, pp. 272–285.

[72] Ito, K., and Xiong, K. Gaussian filters for nonlinear filtering problems. *IEEE Transactions on Automatic Control 45*, 5 (2000), 910–927.

[73] Jäger, G., and Rogers, J. Formal language theory: Refining the Chomsky hierarchy. *Philosophical Transactions of the Royal Society B: Biological Sciences 367*, 1598 (2012), 1956–1970.

[74] Jain, A. K., and Vailaya, A. Image retrieval using color and shape. *Pattern Recognition 29*, 8 (1996), 1233–1244.

[75] Jiang, T., Li, M., Ravikumar, B., and Regan, K. W. Formal grammars and languages. In *Algorithms and Theory of Computation Handbook* (2010), Chapman & Hall/CRC, pp. 20–26.

[76] Jingili, N. *Syntactic Generation of Similar Pictures.* PhD thesis, School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg, 2019.

[77] Jingili, N., Ewert, S., and Sanders, I. Measuring perceptual similarity of syntactically generated pictures. In *8th International Conference on Simulation and Modelling Methodologies, Technology and Application (SIMULTECH), 30 July, Porto, Portugal* (2018). Accessible http://insticc.org/node/TechnicalProgram/simultech/presentationDetails/69065, [Last accessed 20-September-2018].

[78] Jowers, I., and Earl, C. Implementation of curved shape grammars. *Environment and Planning B: Planning and Design 38*, 4 (2011), 616–635.

[79] Jowers, I., Hogg, D. C., McKay, A., Chau, H. H., and De Pennington, A. Shape detection with vision: Implementing shape grammars in conceptual design. *Research in Engineering Design 21*, 4 (2010), 235–247.

[80] Kara, M. Identity Theft in United States of Amarica, Bureau of Justice Statistics, US. *Retrieved from http://www.ojp.gov/* (2015).

[81] Katsini, C., Fidas, C., Raptis, G. E., Belk, M., Samaras, G., and Avouris, N. Influences of human cognition and visual behavior on password strength during picture password composition. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2018), no. 87 in CHI '18, ACM, pp. 73–87.

[82] Kelley, P. G., Komanduri, S., Mazurek, M. L., Shay, R., Vidas, T., Bauer, L., Christain, N., Cranor, F. L., and Lopez, J. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In *2012 IEEE Symposium on Security and Privacy* (2012), IEEE, p. 212–226.

[83] Kendall, M. *The Advanced Theory of Statistics*, vol. 4. Macmillan, 1979.

[84] Knight, T., and Stiny, G. Making grammars: From computing with shapes to computing with things. *Design Studies 41*, Part A (2015), 8–28. https://doi.org/10.1016/j.destud.2015.08.006.

[85] Knight, T. W. Color grammars: the representation of form and color in design. *Leonardo 26* (1993), 117–124.

[86] Knight, T. W. Shape grammar and color grammar in design. *Environment and Planning B: Urban Analytics and City Science 21*, 6 (1994), 705–735.

[87] Knight, T. W. Shape grammars and color grammars in design. *Environment and Planning B: Planning and Design 21*, 6 (1994), 705–735.

[88] Knight, T. W. *Transformations in design: A formal approach to stylistic change and innovation in the visual arts*. Cambridge University Press, 1995.

[89] Knight, T. W. Introduction to shape grammars, MIT/Miyagi Workshop: Lecture notes, February 2000.

[90] Koning, H., and Eizenberg, J. The language of the Prairie: Frank Lloyd Wright's Prairie houses. *Environment and Planning B: Planning and Design 8*, 3 (1981), 295–323.

[91] Kreyss, J., Roeper, M., Alshuth, P., Hermes, T., and Herzog, O. Video retrieval by still-image analysis with imageminer. In *Electronic Imaging'97* (1997), International Society for Optics and Photonics, pp. 36–44.

[92] Krishnamurti, R., and Earl, C. Shape recognition in three dimensions. *Environment and Planning B: Planning and Design 19*, 5 (1992), 585–603.

[93] Kruger, H., and Ewert, S. Translating mutually recursive function systems into generalized random context picture grammars: reviewed article. *South African Computer Journal 2006*, 36 (2006), 99–109.

[94] Li, A. I. Integrating symbolic and spatial information in shape grammars, with an example from traditional Chinese architecture. In *Proceedings of the Fifth Conference of the Association of Computer Aided Architectural Design Research in Asia* (2000), pp. 245–253.

[95] Li, A.-K., Chau, H., Chen, L., and Wang, Y. A prototype system for developing two-and three-dimensional shape grammars. In *Proceedings of the 14th International Conference of Computer-Aided Architectural Design Research in Asia* (2009), pp. 717–726.

[96] Li, X., Schmidt, L. C., He, W., Li, L., and Qian, Y. Transformation of an EGT grammar: New grammar, new designs. *Journal of Mechanical Design 126*, 4 (2004), 753–756.

[97] Li, Z.-N., Zaiane, O. R., and Tauber, Z. Illumination invariance and object model in content-based image and video retrieval. *Journal of Visual Communication and Image Representation 10*, 3 (1999), 219–244.

[98] Lindenmayer, A. Mathematical models for cellular interactions in development I. filaments with one-sided inputs. *Journal of Theoretical Biology 18*, 3 (1968), 280–299.

[99] Ma, W.-Y., and Manjunath, B. S. Netra: A toolbox for navigating large image databases. *Multimedia Systems 7*, 3 (1999), 184–198.

[100] March, L., and Stiny, G. Spatial systems in architecture and design: Some history and logic. *Environment and Planning B: Planning and Design 12*, 1 (1985), 31–53.

[101] Martin, J. *Introduction to Languages and the Theory of Computation.* McGraw-Hill, New York, 2003.

[102] MathWorks. Image Comparison Using Statistical Parameters. *Retrieved from https://www.mathworks.com/help/images/image-mean-standard-deviation-and-correlation-coefficent.html Accessed October, 03* (2016).

[103] McCormack, J. P., Cagan, J., and Vogel, C. M. Speaking the Buick language: Capturing, understanding, and exploring brand identity with shape grammars. *Design Studies 25*, 1 (2004), 1–29.

[104] McKay, A., Chase, S., Shea, K., and Chau, H. H. Spatial grammar implementation: From theory to useable software. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 26*, 02 (2012), 143–159.

[105] Minh, D., Stefan, L., Duygu, C., Boris, N., Peter, W., and Mark, P. Interactive design of probability density functions for shape grammars. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 01–13.

[106] Mitchell, W. J. CAD as a social process. *In the Global Design: Proceedings of the Sixth International Conference on Computer-Aided Architectural Design Futures, National University of Singapore* (1995), 7–9.

[107] Norman, D. A. *The design of everyday things: Revised and expanded edition.* Basic books, New York, 2013.

[108] Okundaye, B. *A Tree Grammar-based Visual Password Scheme.* PhD thesis, School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg, 2015.

[109] Okundaye, B., Ewert, S., and Sanders, I. A novel approach to visual password schemes using tree picture grammars. In *Proceedings of the 2014 PRASA, RobMech and AfLaT International Joint Symposium* (2014), pp. 247–252.

[110] Orsborn, S., Cagan, J., Pawlicki, R., and Smith, R. C. Creating cross-over vehicles: Defining and combining vehicle classes using shape grammars. *AIE EDAM: Artificial Intelligence for Engineering Design, Analysis, and Manufacturing 20*, 03 (2006), 217–246.

[111] Ortega, M., Rui, Y., Chakrabarti, K., Mehrotra, S., and Huang, T. S. Supporting similarity queries in MARS. In *Proceedings of the Fifth ACM International Conference on Multimedia* (1997), ACM, pp. 403–413.

[112] Özkar, M., and Stiny, G. Shape grammars. In *ACM SIGGRAPH 2009 Courses* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 1–22.

[113] Paivio, A., Rogers, T. B., and Smythe, P. C. Why are pictures easier to recall than words? *Psychonomic Science 11*, 4 (1968), 137–138.

[114] Pauwels, P., Strobbe, T., Eloy, S., and De Meyer, R. Shape grammars for architectural design: The need for reframing. In *International Conference on Computer-Aided Architectural Design Futures (CAAD Futures 2015)* (2015), Springer, pp. 507–526.

[115] Post, E. L. Formal reductions of the general combinatorial decision problem. *American Journal of Mathematics 65*, 2 (1943), 197–215.

[116] Prentice, S., Tam, S., and Uh, J. Final project: Architectural grammars. *Massachusetts Institute of Technology, USA* (2011), 1–19.

[117] Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. *Numerical Recipes in C*, vol. 2. Cambridge University Press, 1992.

[118] Prusinkiewicz, P., Hanan, J., and Mech, R. An L–system–based plant modeling language. In *Applications of graph transformations with industrial relevance* (2000), Springer, Berlin, pp. 395–410.

[119] Przemyslaw, P., and James, H. Lindenmayer systems, fractals, and plants, November 1989. Lecture notes in Biomathematics. URL: http://algorithmicbotany.org/papers/lsfp.pdf. Last Accessed on 01-April-2019.

[120] Rani, S., Abdul, H., Chandrasekaran, M., and Subramanian, K. G. Stochastic puzzle grammars. *International Journal of Pattern Recognition and Artificial Intelligence 6* (1992), 257–272.

[121] Rasekgala, M., Sanders, I., Ewert, S., and Fogwill, T. Shape grammar model generating secure visual passwords: The move towards completely grammar based images. In *Second International Conference on Advances in Computing, Communication and Information Technology - CCIT 2014* (2014), Birmingham, UK, pp. 208–214.

[122] Renaud, K., and De Angeli, A. Visual passwords: Cure-all or snake-oil? *Communications of the ACM 52*, 12 (2009), 135–140.

[123] Revett, K. *Behavioral biometrics: A remote access approach*, vol. 1. John Wiley & Sons, 2008.

[124] Rob, D. Identity theft victims statistics. *Identity theft and scan prevention services official website, retrieved from http://www.identitytheft.info/victims.aspx* (2016).

[125] Roger, G. A transformation system for generating description languages of chain code pictures. *Theoretical Computer Science 68* (1989), 239–252.

[126] Rounds, W. C. Context-free grammars on trees. In *Proceedings of the first annual ACM symposium on Theory of computing* (1969), ACM, pp. 143–148.

[127] Rubner, Y., Tomasi, C., and Guibas, L. J. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision 40*, 2 (2000), 99–121.

[128] Rui, Y., Huang, T. S., and Chang, S.-F. Image retrieval: Current techniques, promising directions, and open issues. *Journal of Visual Communication and Image Representation 10*, 1 (1999), 39–62.

[129] Santiago, B., Gonzalo, B., and Gustavo, P. Visual copy & paste for procedurally modeled buildings by ruleset rewriting. *Computers & Graphics 37* (2013), 238–246.

[130] Savvas, A. C., Yiannis, S. B., and Mathias, L. Img(rummager): An interactive content based image retrieval system. In *SISAP '09 Proceedings of the 2009 Second International Workshop on Similarity Search and Applications, Prague, Czech Republic — August 29 - 30* (2009), IEEE Computer Society, pp. 151–153.

[131] Savvas, A. C., Yiannis, S. B., and Mathias, L. SpCD - Spatial Color Distribution Descriptor - a fuzzy rule based compact composite descriptor appropriate for hand drawn color sketches retrieval. In *2nd International Conference on Agents and Artificial Intelligence* (2010), Artificial Intelligence, pp. 58–63. Accessible http://hdl.handle.net/11728/10155, [Last accessed 20-September-2018].

[132] Shea, K., and Cagan, J. The design of novel roof Trusses with shape annealing: Assessing the ability of a computational method in aiding structural designers with varying design intent. *Design Studies 20*, 1 (1999), 3–23.

[133] Shepard, R. N. Recognition memory for words, sentences, and pictures. *Journal of Verbal Learning and Verbal Behavior 6*, 1 (1967), 156–163.

[134] Simon, H., Peter, W., Stefan, M. A., Gool, L. V., and Pascal, M. Grammar-based encoding of facades. *Comput. Graph. Forum 29* (2010), 1479–1487. https://DOI:10.1111/j.1467-8659.2010.01745.x.

[135] Speller, T. H., Whitney, D., and Crawley, E. Using shape grammar to derive cellular automata rule patterns. *Complex Systems-Champaign 17*, 1/2 (2007), 79.

[136] Sreelaja, N. K., and Sreeja, N. K. An image edge based approach for image password encryption. In *Security Communication Networks 2017* (2017), Wiley Online Library, p. 5733–5745.

[137] Srihari, R. K., Zhang, Z., and Rao, A. Intelligent indexing and semantic retrieval of multimodal documents. *Information Retrieval 2*, 2-3 (2000), 245–275.

[138] Stanchev, P. General image retrieval model. In *Proceedings of the Conference of the Union of Bulgarian Mathematicians, Pleven* (1998), pp. 63–71.

[139] Stander, A., Dunnet, A., and Rizzo, J. A survey of computer crime and security in South Africa. In *ISSA* (2009), pp. 217–226.

[140] Stauder, J., Sirot, J., Le, B., Hervé, C. E., and O'Connor, N. E. Relating visual and semantic image descriptors. In *EWIMT 2004 - European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology, 25-26 November, London, UK* (2004), EWIMT 2004. Accessible http://doras.dcu.ie/390/, [Last accessed 20-September-2018].

[141] Stenberg, G. Conceptual and perceptual factors in the picture superiority effect. *European Journal of Cognitive Psychology 7*, 1, 1–35.

[142] Sternberg, R. J., and Sternberg, K. *Cognitive Psychology.* Wadsworth, 2011.

[143] Stiny, G. Introduction to shape and shape grammars. *Environment and Planning B 7*, 3 (1980), 343–351.

[144] Stiny, G. What is a design? *Environment and Planning B 17* (1990), 97–103.

[145] Stiny, G. Weights. *Environment and Planning B 19* (1992), 413–430.

[146] Stiny, G. How to calculate with shapes. *Formal engineering design synthesis 19* (2001), 13–27.

[147] Stiny, G., and Gips, J. Shape grammars and the generative specification of painting and sculpture. In *IFIP Congress (2)* (1971), vol. 2, pp. 125–135.

[148] Stiny, G., and Mitchell, W. J. The Palladian grammar. *Environment and Planning B: Planning and Design 5*, 1 (1978), 5–18.

[149] Stouffs, R. Description grammars : A general notation. *Environment and Planning B: Urban Analytics and City 45*, 1 (2016), 106–123.

[150] Stricker, M. A., and Orengo, M. Similarity of color images. In *IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology* (1995), International Society for Optics and Photonics, pp. 381–392.

[151] Subramanian, K., Ibrahim, V., and Csuhaj-Varjú, E. On the power of permitting features in cooperating context-free array grammar systems. *Discrete Applied Mathematics 161* (2013), 2328–2335.

[152] Subramanian, K., Ibrahim, V., Linqiang, P., and Atulya, K. N. Permitting features in P systems generating picture arrays. In *Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Application (BIC-TA 2012)* (2012), Advances in Intelligent Systems and Computing, Vol 201, Springer, India, pp. 27–38.

[153] Subramanian, K., Pradeep, I., Ibrahim, V., Linqiang, P., and Atulya, N. Array P systems with permitting features. *Journal of Computational Science 5*, 2 (2014), 243–250.

[154] Subramanian, K., Saravanan, R., and Helen Chandra, P. Cooperating basic puzzle grammar systems. In *Lecture Notes in Computer science* (2006), Springer–Verlag, pp. 354–360.

[155] Subramanian, K., Siromoney, R., Reda, V., and Saoudi, A. Basic puzzle grammars and isosceles right triangles. *International Journal of Pattern Recognition and Artificial Intelligence, 6*, 5 (1992), 799–816.

[156] Subramanian, K., Siromoney, R., Reda, V., and Saoudi, A. Basic puzzle languages. *International Journal of Pattern Recognition and Artificial Intelligence, 9*, 5 (1995), 763–775.

[157] Subramanian, K., Sriram, S., Song, B., and Pan, L. An overview of 2-d picture array generating models based on membrane computing. In *Reversibility and Universality, Essays Presented to Kenichi Morita on the Occasion of his 70th Birthday* (2018), pp. 333–356.

[158] Swain, M. J., and Ballard, D. H. Color indexing. *International Journal of Computer Vision 7*, 1 (1991), 11–32.

[159] Tapia, M. A visual implementation of a shape grammar system. *Environment and Planning B: Planning and Design 26*, 1 (1999), 59–73.

[160] Taycher, L., La Cascia, M., and Sclaroff, S. Image digestion and relevance feedback in the imagerover www search engine. In *Proceedings of Visual, San Diego* (1997), vol. 97, pp. 85–91.

[161] Thatcher, J. W. Generalized2 sequential machine maps. *Journal of Computer and System Sciences 4*, 4 (1970), 339–367.

[162] Trescak, T., Esteva, M., and Rodriguez, I. A shape grammar interpreter for rectilinear forms. *Computer-Aided Design 44*, 7 (2012), 657–670.

[163] van der Walt, A., and Ewert, S. A property of random context picture grammars. *Theoretical Computer Science 301*, 1 (2003), 313–320.

[164] van der Walt, A. P. Random context languages. *Information Processing 71* (1972), 66–68.

[165] Vendrig, J. Filter image browsing: A study to image retrieval in large pictorial databases. *Master's thesis, Department of Computer Science, University of Amsterdam, The Netherlands. February* (1997).

[166] Wiedenbeck, S., Waters, J., Birget, J.-C., Brodskiy, A., and Memon, N. Authentication using graphical passwords: Effects of tolerance and image choice. In *Proceedings of the 2005 Symposium on Usable Privacy and Security* (2005), ACM, pp. 1–12.

[167] Wiedenbeck, S., Waters, J., Birget, J.-C., Brodskiy, A., and Memon, N. Passpoints: Design and longitudinal evaluation of a graphical password system. *International Journal of Human-Computer Studies 63*, 1 (2005), 102–127.

[168] Wiedenbeck, S., WatersLeonardo, J., and Birget, S.-C. Design and evaluation of a shoulder-surfing resistant graphical password scheme. In *Proceedings of the working conference on Advanced visual interfaces, AVI 2006, Venezia, Italy, May 23-26* (2006), ACM, pp. 177–184.

[169] Xu, C., and Prince, J. L. Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing 7*, 3 (1998), 359–369.

[170] Yamamoto, Y., Morita, K., and Sugata., K. Contextsensitivity of two-dimensional regular array grammars. *International Journal of Pattern Recognition and Artificial Intelligence 3*, (3 & 4) (1989), 295–319.

[171] Yasunori, Y., Kenichi, M., and Kazuhiro, S. Context sensitivity of two-dimensional regular array grammars. *International Journal of Pattern Recognition and Artificial Intelligence 3*, (3 & 4) (1989), 295–319.

[172] You, K. C., and Fu, K.-S. A syntactic approach to shape recognition using attributed grammars. *IEEE Transactions on Systems, Man, and Cybernetics 9*, 6 (1979), 334–345.

[173] Young, I. T., and Van Vliet, L. J. Recursive implementation of the Gaussian filter. *Signal Processing 44*, 2 (1995), 139–151.

Appendix A

Source Code

Some sample code snippets of the stages of the image generation using the BCSGI tool are shown below.

```
For i2 = 1 To Val(txtindex.Text) ' initialize the initial bag
   bag.Add(Val(txtbag.Text))
Next
PictorialForm.Add(cmbLabel.Text)
For Each nonTerminal In PictorialForm
   If nonTerminal = "d" Or nonTerminal = "w" Then
   Else
      nonTerminalList.Add(nonTerminal)
   End If
Next
nonTerminalList.Sort()
```

Listing 1: Initialization stage

```
j = randomize.Next(0, nonTerminalList.Count - 1)
nonTerminal1 = nonTerminalList.Item(j) 'picks the nonterminal in the position j
nonTerminalList.RemoveAt(j)
For Each item As ListViewItem In list1.Items
   If item.SubItems(2).Text = nonTerminal1 Then
      ruleIndexes.Add(item.SubItems(0).Text)
   End If
Next
```

Listing 2: Outer collection stage

```
no = randomize.Next(0, ruleIndexes.Count - 1) 'randomly picks a rule number
RuleNo1 = ruleIndexes.Item(no)
ruleIndexes.RemoveAt(no)
```

Listing 3: Inner collection stage

```
For k = 0 To bag.Count - 1 ' compare for the bag
   If bag.Item(k) >= LowerLimit.Item(k) And bag.Item(k) <= UpperLimit.Item(k)
    Then
      k1 = 1
   Else
      k1 = 2
   End If
Next
If k1 = 1 Then
   If num5 = "d" Or num5 = "w" Then
      For nm = 0 To PictorialForm.Count - 1
         If PictorialForm.Item(nm) = nonTerminal1 Then
            PictorialForm.Item(nm) = num5
            ShapenamesList.Add(num6)
            character7 = ""
            num5 = ""
            Exit For
         End If
   Next
For k = 0 To bag.Count - 1 'do bag ajustment
bag.Item(k) = Val(bag.Item(k)) + Val(bagAdjust.Item(k)) 'bag adjustment
Next
End If
```

Listing 4: Deriavtion stage

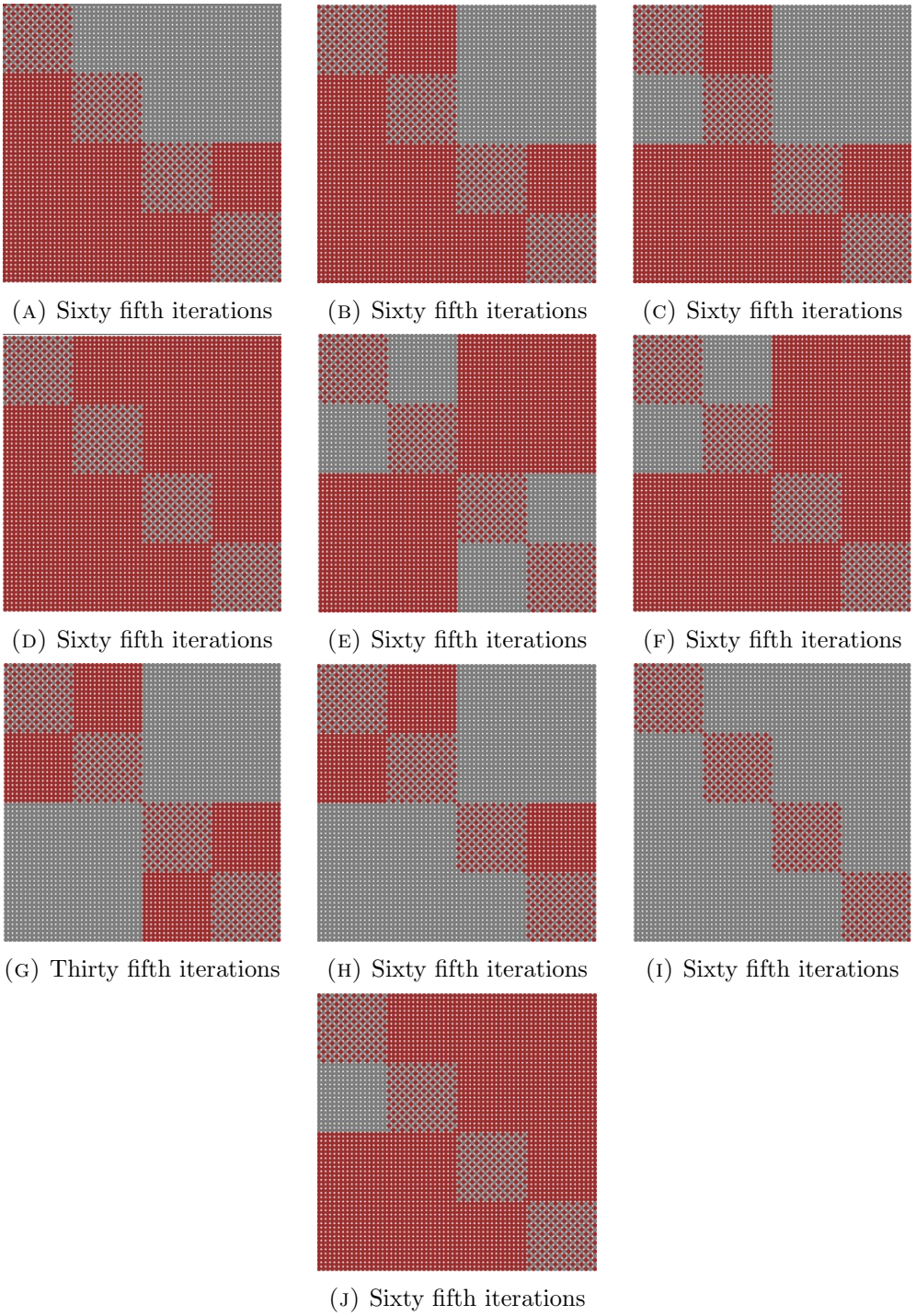
```
Dim img As New Bitmap(1000, 1000)
Dim myGraphics As Graphics = Graphics.FromImage(img)
pixmain.Image = img
```

Listing 5: Conversion stage

Appendix B

BCSG Generated Images

More images generated using the BCSGI tool are shown below. Some images generated with different colour placement and the same building levels are shown in Figure 1 while some images generated with the same colour placement but different building levels are shown in Figures 2 and 3.



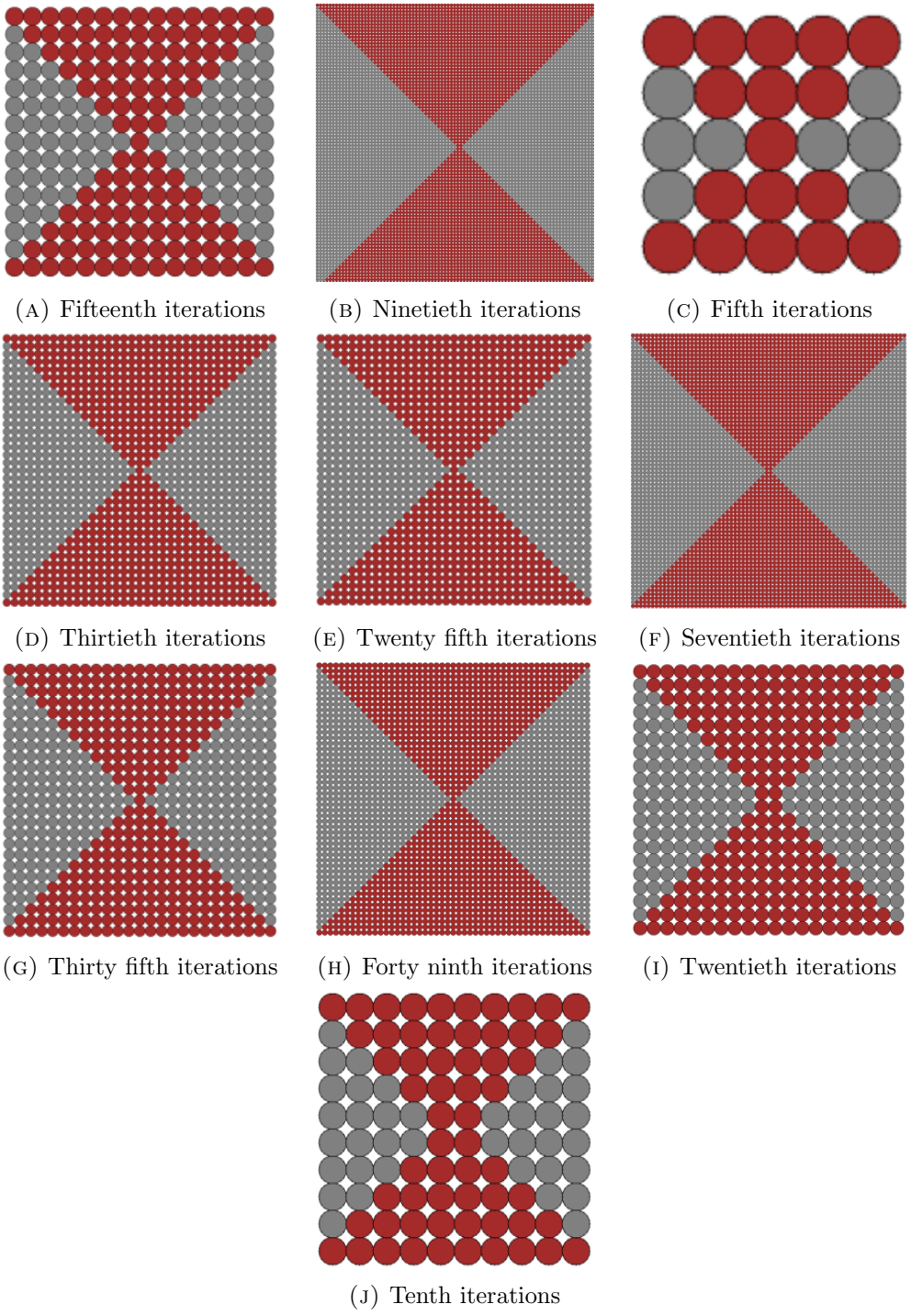
(A) Sixty fifth iterations (B) Sixty fifth iterations (C) Sixty fifth iterations

(D) Sixty fifth iterations (E) Sixty fifth iterations (F) Sixty fifth iterations

(G) Thirty fifth iterations (H) Sixty fifth iterations (I) Sixty fifth iterations

(J) Sixty fifth iterations

FIGURE 1: Carpet A: Images generated with different colour placement but the same building levels.



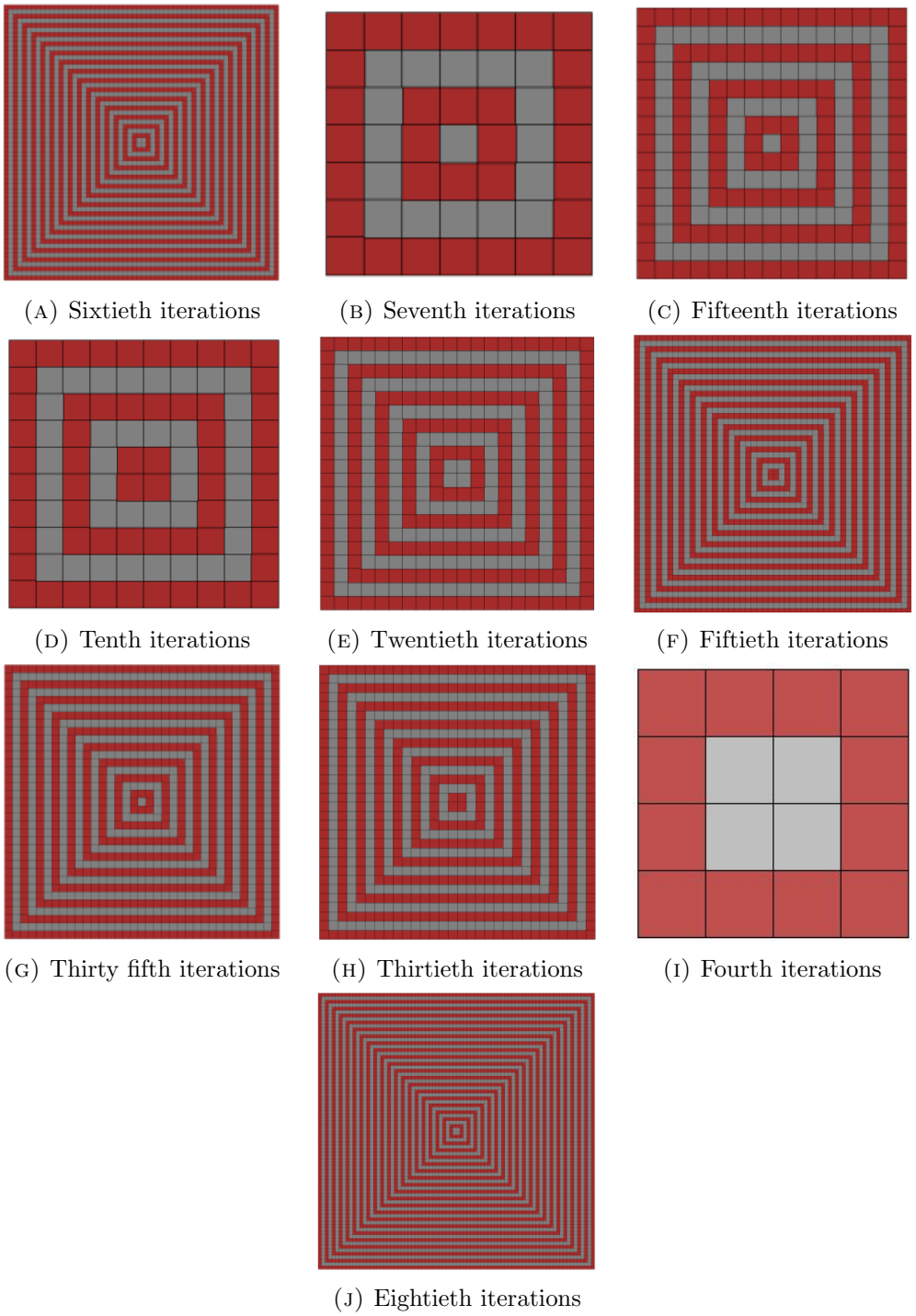
(A) Fifteenth iterations (B) Ninetieth iterations (C) Fifth iterations

(D) Thirtieth iterations (E) Twenty fifth iterations (F) Seventieth iterations

(G) Thirty fifth iterations (H) Forty ninth iterations (I) Twentieth iterations

(J) Tenth iterations

FIGURE 2: Carpet B: Images generated with the same colour placement but different building levels.



(A) Sixtieth iterations (B) Seventh iterations (C) Fifteenth iterations

(D) Tenth iterations (E) Twentieth iterations (F) Fiftieth iterations

(G) Thirty fifth iterations (H) Thirtieth iterations (I) Fourth iterations

(J) Eightieth iterations

FIGURE 3: Carpet C: Images generated with the same colour placement but different building levels.

Appendix C

Ethics Clearance Certificate

UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

Research Office

HUMAN RESEARCH ETHICS COMMITTEE (NON-MEDICAL)
R14/49 Ogbuokiri

CLEARANCE CERTIFICATE	**PROTOCOL NUMBER: H16/09/21**
PROJECT TITLE	Generation of similar images using bag context shape grammars
INVESTIGATOR(S)	Mr B Ogbuokiri
SCHOOL/DEPARTMENT	Computer Science and Applied Mathematics/
DATE CONSIDERED	23 September 2016
DECISION OF THE COMMITTEE	Approved
EXPIRY DATE	17 October 2019

DATE 18 October 2016 **CHAIRPERSON** (Professor J Knight)

cc: Supervisor : Professor S Ewert

DECLARATION OF INVESTIGATOR(S)

To be completed in duplicate and **ONE COPY** returned to the Secretary at Room 10004, 10th Floor, Senate House, University. Unreported changes to the application may invalidate the clearance given by the HREC (Non-Medical)

I/We fully understand the conditions under which I am/are authorized to carry out the abovementioned research and I/We guarantee to ensure compliance with these conditions. Should any departure to be contemplated from the research procedure as approved I/We undertake to resubmit the protocol to the Committee. **I agree to completion of a yearly progress report.**

Signature **Date** 21 / 10 / 2016

PLEASE QUOTE THE PROTOCOL NUMBER ON ALL ENQUIRIES

Appendix D

Approval Letter

UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

OFFICE OF THE DEPUTY REGISTRAR

16 September 2019

Blessing Ogbuokiri
Student Number 1545654
PhD Candidate
School of Computer Science and Applied Mathematics

TO WHOM IT MAY CONCERN

"Generation of similar images using bag context shape grammars"

This letter serves to confirm that the above project has received permission to be conducted on University premises, and/or involving staff and/or students of the University as research participants. In undertaking this research, you agree to abide by all University regulations for conducting research on campus and to respect participants' rights to withdraw from participation at any time.

If you are conducting research on certain student cohorts, year groups or courses within specific Schools and within the teaching term, permission must be sought from Heads of School or individual academics.

Ethical clearance has been obtained. (Protocol Number: H16/09/21)

Nicoleen Potgieter
University Deputy Registrar

Private Bag 3, Wits, 2050, South Africa | **T** +27 11 717 1204/8 | **F** +27 86 553 2271 | www.wits.ac.za

25 September, 2019.

Good day,

I am a PhD student at the School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg. I am conducting a research titled "Generation of Similar Images Using Bag Context Shape Grammars", as part of the research, I am required to develop a prototype visual password scheme using the similar images I generated with bag context shape grammars.

The prototype visual password scheme is tested for usability to ascertain if users will be able to remember their image password immediately after login and one week later.

I invite you to participate in the survey by using the prototype visual password scheme to enable me get the required feed back needed for the research. Your participation is strictly voluntary and no personal information is needed. Your participation will take about 10 minutes. Using the prototype visual password software indicates your consent to participate in the survey.

If you have any queries or concern, please contact me on (1545654@students.wits.ac.za) or my supervisors (mraborife@uj.ac.za and raseelo.moitsheki@wits.ac.za)

Thank you,

Blessing Ogbuokiri