5-2018

Computational Investigation of the Morphological Design Dimensions of Historic Hexagonal-Based Islamic Geometric Patterns

Mostafa Waleed Hashem Alani
Clemson University

COMPUTATIONAL INVESTIGATION OF THE MORPHOLOGICAL DESIGN DIMENSIONS OF HISTORIC HEXAGONAL-BASED ISLAMIC GEOMETRIC PATTERNS

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Planning, Design, and the Built Environment

---

by
Mostafa Waleed Hashem Alani
May 2018

Accepted by:
Dr. M. Carlos Barrios Kleiss, Committee Chair
Dr. Akel I. Kahera
Dr. Cliff Ellis
Joseph Choma

ABSTRACT

This dissertation examines the morphology of Islamic Geometric Patterns (IGP). Using mixed methods, including the simulation of historical designs and content analysis, this dissertation explores the question of how it is possible to mathematically describe the IGP. The study argues that the compositional analysis of geometry is not solely sufficient to investigate the design characteristics of the IGP, and the underlying mathematics and computational nature of the IGP should be considered when investigating historical IGP.

The study presents a parametric description method that captures the reality of the IGP in numeric form and utilizes the form to derive representational codes that include the information necessary to construct a geometry. The representational codes are utilized to further investigate the actual and virtual design space of the IGP, aiming at identifying morphological similarities between historical designs.

This research challenges the long-standing paradigm that considers compositional analysis to be the key to researching historical IGP. Adopting a mathematical description shows that the historical focus on existing forms has left the relevant structural similarities between historical IGPs understudied.

The research focused on the historical, hexagonal-based IGP and found that hexagonal-based IGP designs correlate to each other beyond just the actualized dimension and that deep, morphological connections exist in the virtual dimension. Using historical evidence, this dissertation identifies these connections and presents a categorization system that groups designs together based on their ‘morphogenetic’ characteristics.

DEDICATION

To my parents, Samerah Alfaraj and Waleed Alani.

To my wife, Rousel; and to my son and daughters, Ibrahim, Sarah, and Rania.

ACKNOWLEDGMENTS

I would like to express my gratitude to all of my committee members. I would like to deeply thank my doctoral dissertation chairperson, Dr. Carlos Kleiss, for his guidance at both the academic and personal levels as well as his understanding and continued support. I am thankful I had the opportunity to work with him. I would also like to thank Dr. Akel Kahera, who greatly inspired my work in the area of Islamic architecture. In addition, I would like to thank Joseph Choma for his great support and enthusiasm about my area of research, especially morphology. I would also like to thank Dr. Cliff Ellis for his insights and valuable inputs regarding research design. Furthermore, I would like to thank Dr. Mickey Lauria, the director of the PDBE program, for the many opportunities provided and his support. I would also want to acknowledge my friends at the PDBE program who have been with me through this incredible journey.

Finally, I would like to thank the Higher Committee of Education Development in Iraq, the office of the Iraqi Prime Minister, and Aliraqia University for providing me this wonderful opportunity and for funding my doctoral research.

TABLE OF CONTENTS

Table of Contents (Continued)

Table of Contents (Continued)

LIST OF TABLES

Table Page

LIST OF FIGURES

CHAPTER ONE

ISLAMIC ARCHITECTURE IN THE DIGITAL AGE

1.1 INTRODUCTION

The science and technology of the digital age is revolutionizing architecture (Kolarevic 2004). Digitals, both computerized and computational, advance the research and design practice of architecture by opening new opportunities to explore complex formal compositions and have recently shifted focus from the traditional “representational” nature of architecture toward design “formalism” (Oxman, Oxman 2014).

When it comes to the research and design of Islamic architecture, digitals are used primarily as an alternative to conventional tools such as pen and paper. Consequently, the discipline is overly dependent on approaches that focus on the formal representation of historical models. However, limiting Islamic architecture to particular compositional characteristics neglects the intellectual process responsible for producing designs.

The inquiry should go beyond existing examples and examine “the emergence and evolution” of architectural forms. Such an approach provides new research opportunities and reestablishes an “open-ended” search for the forms that make Islamic architecture an active contributor to global architecture (Rabbat 2004).

1.2 BACKGROUND

Historically, Islamic art and architecture took advantage from the mathematics of its age. The enormous diversity of complex forms that exist in Islamic art and architecture are products of mathematical and geometrical advancements as discussed in available historic documentation. One such document is Risâla fimâ yahtâju al-sâni'u min a'mâl al-handasa (On the Geometric Constructions Necessary for the Artisan), by Abu al-Wafa' al-Būzjānī, (998). This manuscript shows that mathematicians collaborated with artisans to explore new relationships and prefect designs. Moreover, in the Fi tadakhul al-ashkal al-mutashabiha aw mutawafiqa (On interlocking similar or congruent figures) manuscript, the author demonstrates awareness of several mathematical relationships such as the Pythagorean theorem and binomials (Chorbachi 1989). Yet, when it comes to the research and design of Islamic architecture, mathematics is mainly discussed in terms of proportion with less focus on the contribution of mathematics in the design process.

European scholars conducted several surveys that examined architectural sites where Islamic monuments reside in the nineteenth and early twentieth centuries. They produced chronological and geographical classifications of building typologies and styles that essentially focused on the formal compositions of Islamic architecture (Rabbat 2012). The discourse that followed these "Orientalism" studies mainly took two different approaches. One emphasized regional differences, while the other attempted to reproduce romantic architecture that reflects the past —nationalism and neo-Islamism. These approaches, because they only show consideration to particular formal styles, are both criticized of viewing Islamic architecture as a "stagnant" product that has ceased evolving (Rabbat 2004). With some exception of attempts by Rifat al-Chaderchi and Kamal el-Kafrawi, who –as Nasser Rebbat argues—that actively engaged the design practiced through examining and understanding historic models, and produced designs that adopt the architectural style of their age (Rabbat 2012).

1.3 THEORETICAL FRAMEWORK

The study and research of historic Islamic architecture are influenced by both positivism and relativism paradigms. European scholars from the nineteenth century took a positivist approach when examining historic Islamic architecture for the IGP. Several architectural monuments, especially in Spain, Egypt, the Holy Land, and Turkey, have been measured, documented, and dated. Later, this information was compiled to produce regional catalogs of selected works that introduced Islamic architecture to Europe with some attempts to analyze the underlying grids of the IGP. For instance, Owen Jones, in his book The Grammar of Ornament, published in 1856, presents the first systematic formal approach for researching the IGP (Jones 1868). Prisse d'Avennes, in L'art arabe, published in 1877, was the first to observe that the underlying grid of the IGP was based on scientific knowledge shared between various Islamic cities through design scrolls (d'Avennes 1877). Jules Bourgoin, who did not have access to the historical scrolls, had classified the patterns into categories according to their inner grid (Bourgoin 1879). This approach was used later by other European researchers in North Africa and Spain (Necipoğlu, Al-Asad 1995).

On the other hand, other research argued that there is a spiritual meaning behind the art of geometric patterns. However, tracing back this approach takes us no further than the early part of the past century, during which Titus Burckhardt (Burckhardt 2009). Hossein Nasr, and, later, Keith Critchlow (Critchlow, Hossein-Nasr 1976) interpreted the geometric patterns and their hidden circles as a symbol of Islam referring to al-Tawhid, —the monotheism. This approach has been criticized of being highly subjective and lacking the historical evidence to support its argument (Chorbachi 1989). None of the discovered design manuscripts mention such an interest. The only argument that has the historic evidence to support it is that the IGP are the products of scientific advancements in mathematics and geometry.

Neither positivism nor relativism alone is well suited for this research. The understanding of IGP should emerge from the actualized data and can be used to "critically test and develop ideas about the existence and nature of the phenomena" (Groat, Wang 2002). The ontological foundation of this research acknowledges the existence of independent reality and views it as stratified. This means that reality is not only what is observed but is the result of a deeper-level process. This process is responsible for producing the multiplicities of observed reality (DeLanda 2002). Therefore, to understand the reality, the design process must be investigated. This ontological view of reality aligns with the writings of French philosopher Gilles Deleuze, who pioneered a theory of how forms come into existence—morphogenesis. Deleuze argued that the forms we observe in reality are an "actualized" state of an idea while the generative process is capable of producing other possibilities, what Deleuze identifies as "virtual" realities. This potential population of virtual design multiplicities precedes actualized design singularity and is perceived as just as real as the actual.

Epistemologically, this research assumes that reality can be known using a wide range of research tools that are both objective and subjective. Methodologically, the research process involves both quantitative and qualitative methods to collect and analyze data from multiple resources. This provides a deeper understanding of the problem being addressed.

1.4 STATEMENT OF THE PROBLEM

The dominant approach to studying IGP is representational in nature and focuses on the formal characteristics of historic singularities, with less focus on the mathematics and relations between design parts. Consequently, research on IGP does not incorporate the computational mechanism that is responsible for producing the design multiplicities in the investigation process. This representational approach is clearly evident in studies aimed at establishing systems of categorization to group together designs that share similar characteristics through the identification of an underlying grid system. The result is classification of the IGP into several categories. The designs included in each category range from designs that share the same repetition structures such as groups of square- or hexagon-based designs (Bourgoin 2012, Broug 2013a, Jones 1868)to designs that share the same system of proportions, such as in Issam El-said's study (El-Said, El-Bouri & Critchlow 1993a).

However, a few studies have moved beyond this traditional formal approach—and its 'Orientalist roots'—to emphasize the relationship between mathematics and historic IGP. One such study is by Wasma al Chorbachi, where the author examined the geometry in the *On Interlocking* manuscript and identified the formula used to generate the design. By manipulating the formula, she was able to derive several new design variations(Chorbachi 1989). Although this study examined a specific design, it provides an approach that is concerned with utilizing a scientific method rather than merely focusing on the formal qualities.

However, the mathematical approach to classifying the patterns is primarily based on symmetry. The first study that scientifically investigated IGP was conducted by Edith Müller (Müller 1944, Necipoğlu, Al-Asad 1995). Müller analyzed the symmetry of the patterns based on group theory. This research was followed by a publication by Sayed Abas and Amer Salman (1995) who attempted to identify a method to categorize the design of IGP. They acknowledged the important contribution of group theory in studying the patterns and used scientific notation to identify individual geometric designs. Mohamed Ould Djibril developed a computational method for identifying the symmetry group of the patterns (Djibril, Thami 2008) However, these studies did not investigate internal geometric designs. Rather, they focused on repetition and design propagation using symmetry. Consequently, it remains unapparent how designs that share the same symmetry may relate to or be differentiated morphologically from each other.

1.5 RESEARCH QUESTION

The research addresses the question of how to incorporate mathematics and morphology to describe the actual and virtual design space of IGP and identify and graph the relations among design parts? It then utilizes this description, in light of historical evidence, to address the question of what morphological correlations exist among historic design singularities?

1.6 SIGNIFICANCE OF THE RESEARCH

"The precise definition of an ellipse introduces us to all ellipses in the world."

D'Arcy Thompson (1917).

The research and design of Islamic art and architecture must "catch up" and take advantage of the technological advancements of the digital age (Keshani 2012). This requires the development of an "infrastructure" that incorporates the computational mechanism that produces design multiplicities in the investigation process to explore historical designs in a way that goes beyond archiving information—digitization—and can be used as analytical instrument.

This study eliminates the traditional boundary that focuses on either chronological and geographical development or mere geometrical analysis and seeks to provide a computational lens to investigate the historical evidence of surviving historic IGP that exploit innovative tools and the algorithmic nature of IGP. The goal is to provide an alternative understanding of historical IGP based on mathematics and morphology to complement conventional formal understanding that is aimed at establishing a new platform for engaged research on and design of the patterns.

The significance of the identification of design formalism of IGP is that it enables the construction of databases of representations of design singularities, which provide an extensive source of information and connect knowledge on multiple levels. For instance, a single geometry can be examined regarding its design morphology, geographic location, and chronological order. These representations serve to investigate and analyze the morphology of historical designs empirically for possible correlations. In other words, this research provides a non-linear reading of the history of IGP that complements historians' approaches.

Although the focus of this investigation is on IGP, the underlying goal is to provide a method to actively engage the design of Islamic architecture, based on mathematics and morphology, in order to construct a version of history that represents the digital age through incorporating innovative computational tools into the design process. Eventually, this will reduce the gap between the contemporary world's practice of architecture and Islamic architecture by allowing the latter to contribute to current design practices.

1.7 METHODS OVERVIEW

This research utilizes mixed methods in two sequential phases. In phase one, simulation modeling is employed to develop a parametric description that describes the formalism of IGP. This description is used to construct the representational code of historical designs. In the subsequent phase, content analysis is utilized to study and compare the representational codes, searching them for possible correlations.

CHAPTER TWO

ISLAMIC GEOMETRIC PATTERNS

2.1 INTRODUCTION

The goal of chapter two is to discuss the design characteristics of the IGP and identify their chronological and geographical development through examining surviving monuments and historical manuscripts. The chapter provides a discussion on the parallel development in mathematics and its relationship to geometric patterns. Finally, the chapter identifies and discusses related literature for both the formal and mathematical approaches.

2.2 GEOMETRIC MODE OF ISLAMIC PATTERNS

Patterns are a common feature of Islamic architecture and exists in a variety of shapes and types. In general, Islamic patterns have been classified into two main categories: Arabic calligraphy and arabesque (Burckhardt 2009 p.52, Abas, Salman 1995). Calligraphy is the art of Arabic writing in which various types of Arabic calligraphy that belong to different ethnic groups within the Islamic world are used for architectural decoration (Burckhardt 2009). For instance, Kufi style, which consists of simple rectangles and squares, is employed to create façade decoration (figure 2.1).

Arabesque, on the other hand, has two modes: “stylized plant forms” and geometrical patterns (figure 2.2). The plant forms mainly consist of curvilinear elements forming “vines” and other floral forms such as leaves that show rhythm with some degree of symmetry (Abas, Salman 1995, Burckhardt 2009 p.62).

Figure 2.1 illustration of the Arabic calligraphy using Kufi style (Burckhardt 2009 p.44)

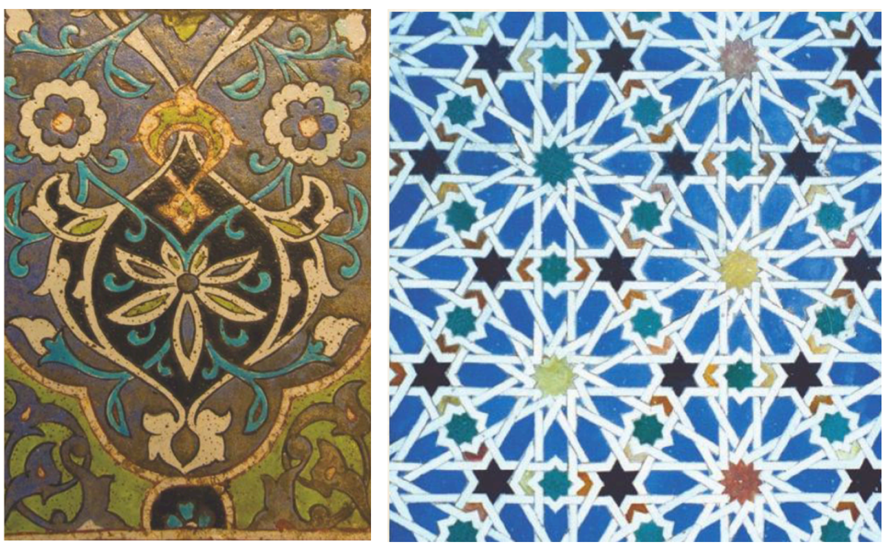


Figure 2.2 Left: illustration of floral design from Bursa, Turkey. Right: Geometric pattern from Granada, Spain(Burckhardt 2009 p.66, p.68)

Geometric patterns, however, are commonly constructed from several polygons or other regular figures (Burckhardt 2009). They consist of a "repeated unit" and a "repetitive structure" (El-Said, El-Bouri & Critchlow 1993a, Abas, Salman 1995). The repeated unit is the minimal possible region that contains the basic geometrical composition, where the repetitive structure is a product of systematically reiterating the repeated unit to fill the space. The shape of the repeated unit dictates the periodicity of the structure. Thus, both periodic and quasi-periodic patterns exist due to the stacking capabilities of the selected repeated unit.

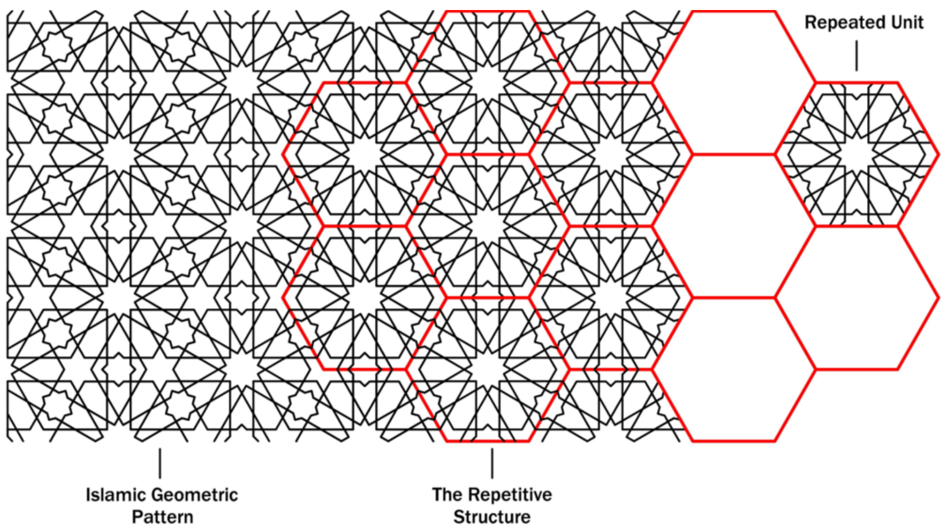


Figure 2.3 Construction of IGP from a repeated unit using a hexagonal structure.

In general, IGP have four recognizable characteristics: symmetry, flow, unboundedness, and interlacing (Abas, Salman 1995). Symmetry is a dominant characteristic of IGP. In fact, 17 types of wallpaper symmetry have been identified in Alhambra Palace alone (Abas, Salman 1995, Müller 1944, Grünbaum, Grünbaum & Shepard 1986a, Pérez-Gómez 1987). Regarding hexagonal patterns, symmetries of type P3, P3M1, P31M, P6, and P6M geometric designs have been identified on various monuments (Abas, Salman 1995).

The flow characteristic of the IGP refers to the continuity of the geometric elements. It causes the eye to follow the lines and observe a variety of compositions and structures (Abas, Salman 1995). Within the flow, designers often utilize "visual anchors", which are typically composed of star components (Broug 2013b). Unboundedness, on the other hand, refers to the ability to recursively and infinitely extend a design by stacking repeated units or expanding the structure in the case of quasi-periodic patterns (Al Ajlouni 2012).

Finally, the interlacing characteristic is found when the rectilinear elements that form patterns overlap (Burckhardt 2009). This characteristic emphasis the feature referred to as uqda in Arabic, or girih in Persian, which means a knot resulting from the interlacement of two lines (Necipoğlu, Al-Asad 1995). These knots can be emphasized or deemphasized based on the type of embellishment chosen by the artisan(Broug 2013b). Thus, in some cases the same geometric design embellished in terms of lines, as in interlaced geometric patterns, or in terms of geometric composition.

It is also common for the same design to appear in different centuries, or appear in different geographic regions. For instance, consider the design in the left of figure 2.2. This design exists in Alhambra palace in Granada, Spain and a similar design found also in Konya, Turkey. However, the question here is how frequent such replications are?

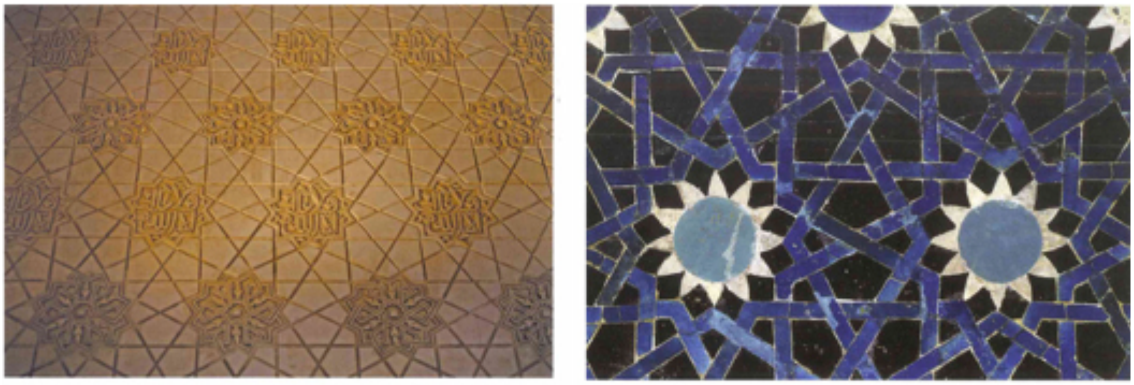


Figure 2.4 Identical designs. Left: design from Alhambra palace Granada, Spain. Right: design from Konya, Turkey(Broug 2013b p.115)

With the formal utilization of digitals, such similarities remain unapparent and can only be identified through manual comparison of historical designs. Computational utilization of digital tools, however, advances investigation and can detect such similarities in a much more efficient manner.

2.3 CHRONOLOGICAL AND GEOGRAPHICAL DEVELOPMENT

Geometry was widely used in ancient Mesopotamia and Egypt for land measurement, building construction, and astronomical calculations. The Greeks built upon this knowledge with Euclid's studied, further discovered, and documented the geometries in a systematic manner. Later, the manuscripts were dispersed in the region and were available to Islamic civilizations (Wilson 1988). Islamic art utilized this knowledge and developed geometric patterns with sophisticated mathematics.

During his expedition in Iraq (1911-13), Ernst Herzfeld, an archeologist and scholar of Islamic architecture, identified the earliest geometric ornamentations in the surviving monuments of the city of Samara, dated to the 9th century; these include Dar al-Khilafa (Palace of the Caliph), constructed 836-42, and other houses in the city dated to the 9th century (Necipoğlu, Al-Asad 1995). These designs are considered to be older than the tiles of the Great Mosque of Kairouan in Tunisia, which is dated to the 856-73[1], and the geometric plasterworks found in the arches of the inner court of the Ibn Tulun Mosque in Egypt (879). Therefore, the consensus is that geometric patterns originated and developed in Abbasid capitals of Baghdad and Samarra, taking advantage of the 9th -century mathematical advancement in the city of Baghdad, and then spread in the region. In fact, the tiles of the Great Mosque of Kairouan were designed and built in Baghdad and shipped to Tunisia (Broug 2013b).

Later examples found in the Maqbara-i Isma'il Samani in Bukhara (914-43) and the Jurjin Mosque in Isfahan were built from brick with no star composition. However, Mazar-i 'Arab 'Ata in Tim, Uzbakistan (977-78) and another royal mausoleum and city minaret in Uzgand (1012-13) show the earliest geometric star designs built from brick (Necipoğlu, Al-Asad 1995).

The geometric designs then spread in the region, appearing in several cities during the Seljuk Dynasty (1040-1157), such as the great mosque in Seljuk's capital Isfahan (1072-92). However, the prime surviving examples in Seljuk are the two tombs towers in Kharraqan (1067-68 and 1093) (Stronach, Young 1966). Each of the towers has eight facades covered with a variety of geometric brick designs. A transition from traditional brick patterns to glass brick can be seen in the Gunbad-I surkh (1147-1148), Gunbad-I

[1] The mosque itself was constructed in 670. However, the tiles were designed and constructed in the time of Emir Abu Ibrahim Ahmed, who governed from 856-73 in the Qubla wall. The design was made in Baghdad and shipped to Tunisia. The initial intent was for it to be used in the palace of Abu Ibrahim Ahmed. However, he changed his mind and requested that these designs be placed in the Qubla wall of the Great Mosque of Kairouan (Eric p.37).

Qabud (1196-97), and Mu'mina Khatun (1186) towers of Maragha and Nakhehivan in Iran, where color was introduced for the first time in geometric designs(Necipoğlu, Al-Asad 1995).

The successors of the Seljuk Dynasty in the region, (i.e., Rum Seljuk (1081-1307), Zangids (1127-1222), Ayyubids (1169-1260), and Mamluks (1250-1517) also utilized geometric patterns in their designs. A Rum Seljuk example is the minbar of the Ala' al-Din Mosque in Konya Turkey (1155). A Zangid example is Nur al-Din Zangi Mosque in Hama (1163-1164). Moreover, the majority of Mamluks monuments show great variety in the use of geometric designs (Necipoğlu, Al-Asad 1995).

Scholars(Herzfeld 1942, Necipoğlu, Al-Asad 1995)have argued that Baghdad remained the center of innovation even after losing its political importance during the Seljuk Period. Thus, even in the mid-13th century, several sophisticated designs, such as the geometric designs in Madrasa al-Mustansiriyya (1233) and Abbasid Palace (1255), can be found. It was not until the Mongol invasion to Baghdad (1258) that the city's importance began to decline. Later, during the Mongol-Ilkhanid Period, decorative design became more focused on floral designs. However, there are examples employing geometric compositions in decoration, such as Khanqah-i Shaykh 'Abd al-Samad, constructed between 1304 and 1325 (Broug 2008). Later, the existence of geometric patterns in the eastern part of Islamic lands mainly remained apparent in the buildings of the Timurid Dynasty (1370-1506 CE), where several monuments were decorated with geometric designs (Necipoğlu, Al-Asad 1995).

In the western Islamic world, several monuments from the Almoravid Dynasty (1053-1150) utilize two-dimensional and three-dimensional geometric designs, such as the Qarawiyyin Mosque (1135-1144) and the Marrakish Qubba (1066-1142). In neighboring Spain, the Spanish Umayyad for a long time did not use geometric patterns in their monuments. It was not until Almoravid unified North Africa and Spain that geometric designs began to appear in Spain, reached their peak in Alhambra Palace during the Nasrid Dynasty (1232-1492) (Necipoğlu, Al-Asad 1995).

Therefore, close examination shows that geometric patterns emerged in Iraq in the 9th century and then appeared in several Abbasid buildings in different regions by the 10th century. They then appeared in several Seljuk monuments in Iran and Iraq during the 11th century. Several Seljuk successors used them as well. Later, geometric designs mainly exist in Mamluks, Nasrid, and Timrud monuments.

2.4 PRIMARY SOURCES: HISTORICAL MANUSCRIPT AND DESIGN SCROLLS

Although the design process of IGP has historically been surrounded by secrecy and is inherited by artisans from their masters (Necipoğlu 1992), there is some surviving evidence on how the geometry is designed. Two historical manuscripts and three design scrolls were retrieved and are available today (Necipoğlu, Al-Asad 1995) These are:

1. Risâla fimâ yahtâju al-sâni'u min a'mâl al-handasa (Book on the Geometric Constructions Necessary for the Artisan) by Abu'l Wafa al-Buzjani's from the 10th century. Referred to as On Geometric Constructions in the rest of this dissertation.

2. Fi tadakhul al-ashkal al-mutashabiha aw al-mutawafiqa (On Interlocking Similar and Congruent Figures) by anonymous author from the 13th century. Referred to as On Interlocking in the rest of this dissertation.
3. The Topkapi Scroll by anonymous author from the late 15th the early 16th century.
4. The Tashkent Scrolls by anonymous author from the late 15th to the early 16th century.
5. The Mirza Akbar Scrolls by Mirza Khan from the 19th century.

Al-Buzjani's On Geometric Constructions manuscript demonstrated to artisans the rules by which they can operate instruments to precisely construct different geometrical compositions and addressed the difficulties that artisans may face. This manuscript presents a step-by-step (algorithmic) procedure of constructing circles, identifying points, and creating lines. Al-Buzjani began by explaining the instruments used to design the geometry—gunya, mistar, and alburcar—and how to calibrate these instruments. In the subsequent chapter, he discussed fundamental rules that each artisan should master (e.g., how to divide lines and angles into equal parts and how to determine the center of a circle). In the rest of the book, he explained how to construct different geometrical figures. This text reveals the process of thinking employed in design that, in its core, is based on mathematical proof. The text simplifies this for the artisan through steps of geometric construction. As mentioned by Al-Buzjani himself, he deliberately "excluded [from the book] the causes and proofs, to make it easier for the artisan to understand" (al-Būzjānī 998).

The On Interlocking manuscript, on the other hand is a collection of notes compiled by the anonymous author. The manuscript reveals that the conversation between mathematician and artisans continued centuries after Al-Buzjani as mathematician Abu Bakr al-Khalil is cited on multiple occasions(Özdural 2000). This manuscript shows the awareness and application of mathematical relations in geometric designs such as the Pythagorean theorem and the 2nd degree binomial (Chorbachi 1989). However, mathematicians chose to use the "cut and paste" method when teaching artisans. The author takes an approach similar to Al-Buzjani's by omitting theoretical proof that may have been complicated and difficult for artisans to comprehend (Özdural 2000).

The primary evidence that exists from later periods consists mainly of design scrolls. The Timurids Scroll, the so-called Topkapi Scroll, dates to the late 15th to the early 16th century and contains 114 drawings. Unlike the previous two historic texts, this scroll includes with no commentary that explains the steps of the design process. The scroll acts as a guidebook that presents the modular design method, focusing on the grid system and repeats united and utilizes symmetry to populate the design and fill spaces with geometric patterns (Necipoğlu 1992 p.54, Necipoğlu, Al-Asad 1995).

A parallel scroll is the Tashkent Scroll, which is also dated to the late 15th to the early 16th century. Like the Topkapi Scroll, the Tashkent Scroll shows finished models with no explanatory text. This scroll also focuses on the concept of the repeated unit and the grid system. With the Topkapi Scroll, these two historical documents bridge the gap between two-dimensional and three-dimensional geometric design in Islamic architecture through the demonstrated muqarnas drawings. When compared to actual monuments, the scroll reveals information on how two-dimensional drawings are treated in real three-dimensional geometry (Necipoğlu 1992 p.50).

The last known evidence is the so-called Mirza Akbar Scrolls; these scrolls are a collection of drawings designed by Persian state architect Mirza Khan and include plans, muqarnas, and geometric and calligraphic decoration. They show that the scroll tradition continued into the 19th century.

2.5 MATHEMATICS AND GEOMETRIC DESIGNS

In approximately 832, the al-Ma'mun Caliph established an academy of science called Bayt al-hikma (the House of Wisdom). This institution took over the translation of books from other civilizations in a wide range of subjects. Several books were translated to Arabic from other languages, including Greek Euclidian writings on geometry, as shown in the fihrist (index) of ibn al-Nadim of the translated books. Thus, Abbasid gained knowledge on geometry as early as the 9th century (Al-Khalili 2011).

Scholars in Bayt al-hikma differentiated yet also connected theory and praxis. For instance, the 9th-century philosopher Abū Naṣr Muḥammad ibn Muḥammad al-Fārābī (Alpharabius) in his book Ihsa al-ulum (Survey of Science) divided mathematics into fields of specialized topics in which each has al-nazari and al-amali (theoretical and practical divisions) (Necipoğlu, Al-Asad 1995).

Original contributions in a variety of sciences began to appear after the end of the translation period. Mathematics in particular flourished as new advancements were achieved. For instance, a scholar in Bayt al-hikma, Muḥammad ibn Mūsā al-Khwārizmī, invented algebra by “synthesizing” Greek geometric knowledge and the Indian decimal system. Al-Khwārizmī, in his book al-kitāb al-mukhtaṣar fī ḥisāb al-ğabr wa’l-muqābala (The Compendious Book on Calculation by Completion and Balancing), presented a unified process for problem solving, eventually delivering a new revolutionary “form of mathematical thinking” (Al-Khalili 2010).

Al-Khwārizmī’s contributions to arithmetic and trigonometry are equally important. For instance, his writings in arithmetic were widely read in Europe during medieval times. In fact, “algorithm,” which is derived from his last name al-Khwārizmī, was used to refer to the subject of arithmetic before gaining its modern meaning. Moreover, he further contributed to trigonometry by producing spherical trigonometry, as discussed in his book Zīj al-Sindhind. Al-Khwārizmī’s work influenced several mathematicians such as Thābit ibn Qurra, Sinān ibn al-Fatḥ, and Abu’l Wafa al-Buzjani.

Al-Būzhjānī in particular, who was a famous mathematician and astronomer from the 10th century and produced notable work on mathematics, is considered an important link between the use of mathematics and the design of IGP. He participated in conversions with craftsman, teaching them the correct, precise way of creating geometry. In his book, On the Geometric Constructions, he aimed to facilitate the design of geometry without using complicated mathematical proofs and reasoning -al-barahin wa alillal (al-Būzjānī 998).

It has been argued that another mathematician, Omar Khayyam (1048-1131 CE), participated in such conversations with artisans(Özdural 1995)In an untitled treatise[2] written after 1073, Khayyam explained the cubic equation in practice. Some of the solutions that Khayyam presented were actually used and shown in the On Interlocking manuscript (Özdural 1995). Moreover, the On Interlocking manuscript cites another mathematician, Abu Bakr al-Khalil, as providing a mathematical solution to geometric designs and participating in conversations with artisans.

Mathematics is exploited in a reverse manner (i.e. to convert geometry into numbers) for the purpose of cost estimation, as shown in the Ghiyāth al-Dīn Jamshīd Masʿūd al-Kāshī's miftah al-hisab in which he demonstrates a method for estimating the cost of building a muqarnas. Gülru Necipoğlu observed that "Arithmetic and geometry were two independent but interchangeable modes of expression for the same mathematical concept, one based on the language of numbers the other on the geometric forms" (Necipoğlu, Al-Asad 1995). This holds true in the case of IGP, especially in the early stages when mathematics played a significant role in the establishment and development of the patterns. However, little information is available about the relationship between math and IGP, and the recently discovered design scrolls show dependence on modular-based catalogs.

2.6 FORMAL APPROACH TO IGP

The geometric mode of Islamic patterns received attention from European scholars in the 19th century, mostly because of the "practical agenda" of the Industrial

---

[2] Unlike Al-Būzhjānī's manuscript, Khayyam's treatise does not address the artisan in a direct manner. Alpay Özdural explained that it is uncertain that this particular treatise was directed to artisans, but evidence infers this.

Revolution (Necipoğlu, Al-Asad 1995). Owen Jones, in his book The Grammar of Ornament, which was published 1856, classified the patterns based on the development of ethnic groups as Arab, Persian, Turkish, Indian, and Moresque. Jones' practical agenda led him to take a historical approach when dealing with IGP, similar to the one he uses when dealing with designs from other cultures in the same book. Jones presented the first systematic approach to research on IGP. Jones attempted to formulate a series of propositions by surveying existing designs to create new designs (Jones 1868).

Priss d'Avennes, in L'art Arabe, published in 1877, recognized that the underlying, complex structure of the patterns was based on scientific knowledge shared between various Islamic cities through scrolls. D'Avennes produced catalogs that presented Islamic architecture to Europe. Jules Brourgoin, who did not have access to the historical scroll, classified the patterns into categories not based on the observed appearance of designs but according to the inner grid system: "Hexagon, octagon, dodecagon, star rosette combination of two types, square octagon combination, heptagon, and pentagons." Brourgoin also had a practical agenda with his work of opening the new "infinite possibility" of design. In the 20th century, several studies examined North African designs between 1911 and 1975, utilizing the same approach to formally analyzing the underlying grid systems (Necipoğlu, Al-Asad 1995).

Later, a study by Issam El-Said examined the proportions of the IGP. With a sample of 29 hexagon-based geometric patterns sampled from different regions in the Islamic world, he focused on periodic patterns and identified three categories based on the repeated unit shape and systems of proportion: square patterns based on the root of two, hexagonal patterns based on the root of three, and patterns based on double hexagons (El-Said, El-Bouri & Critchlow 1993b). Like his predecessors, El-Said also had a design agenda focused on proportions to identify “beautiful design”.

2.7 MATHEMATICS-BASED APPROACH TO IGP

The scientific study of IGP can be classified into two categories: studies that focus on the symmetry of IGP and studies that focus on repeated units. Although these approaches are not mutually exclusive, some studies focus on one aspect more than the other.

In 1944, Edith Muller wrote a dissertation on the Moorish ornamentation in Alhambra Palace, employing group theory and crystallography to systematically annotate the patterns. She conducted symmetry analysis and discovered 11 types of symmetries (Müller 1944, Necipoğlu, Al-Asad 1995). Also around the same time, studies on symmetry were conducted by Soviet scholars, such as Gaganov and Baknaov, who had partial access to the historical design method through the Tashkent Scroll. In their work, they also focused on the underlying structure of symmetry groups (Necipoğlu, Al-Asad 1995). In 1986, Branko Grünbaum, Zdenka Grünbaum, and G.C. Shepard further examined Alhambra Palace and discovered 13 types of symmetry groups (Grünbaum, Grünbaum & Shepard 1986b). In 1987, R. Perez-Gomez and J. Montesinos found four missing groups to complete the 17 wallpaper types group theory (Pérez-Gómez 1987). Syed Jan Abas and Amer Shaker Salman, in their book “Symmetries of Islamic Geometrical Patterns,” presented a comprehensive examination of symmetry groups in IGP for geometries beyond Alhambra Palace (Abas, Salman 1995).

Other studies focus on the repeated pattern and its geometric motif. In 1989, Wasma Chorbachi presented a method for designing new IGP that is strongly tied to historical creative design methodology. Chorbachi examined the On Interlocking manuscript, identifying the design formula and manipulating to create new designs(Chorbachi 1989). Haresh Lalvani presented a grid of fixed subdivision for the "fundamental unit" then populating it with motifs to generate the pattern (Lalvani 1989).

The studies that follow are primarily focused on design exploration. Ahmad Aljamali, Craig Kaplan, and Ali Izadi proposed different methods and developed computer programs for design exploration. By defining parameters and manipulating the values of those parameters, they derived new designs (Kaplan, Salesin 2004, Aljamali, Banissi 2003, Izadi, Rezaei & Bastanfard 2010, Riether, Baerlecken 2012).

2.8 SUMMARY

IGP emerged from the intellectual center of the 9th and 10th centuries in the Abased capitals of Baghdad and Samara. The designs employed the most innovative mathematical knowledge of the time to produce a cultural heritage that spread throughout the Islamic world for centuries. Abu'l Wafa al-Buzjani' book On the Geometric Constructions gives important clues about the methodology employed in deriving geometry. The thought process Al-Buzjani employed is known today as "algorithmic design thinking", which identifies a step-by-step procedure of form generation to explore variations and increases the accuracy of the final product. In other words, al-Buzjani took advantage of advancements in mathematics –the language of his age— to create IGP.

Looking closely at the literature from the 19th and 20th century onward, one can clearly see two approaches: formal and scientific. The formal approach is more about practical geometry and conventional design tools. The scientific approach, on the other hand, utilizes mathematics and symmetry and focuses more on the generation of design from scratch with less interest in utilizing the methods for research historical IGP.

CHAPTER THREE

MORPHOLOGICAL DESIGN THINKING

3.1 INTRODUCTION

This chapter discusses the theory of morphogenesis and its relation to architecture. The goal is to expand on the underpinning theoretical framework of this dissertation and to elaborate on the appropriateness of the selected methodology. The chapter also provides foundational definitions and discusses the line of research that utilizes mathematics to describe forms.

3.2 DIGITAL MORPHOGENESIS

The writings of French philosopher Gilles Deleuze (Deleuze 1994, Deleuze, Guattari 1988, Deleuze 1993) from the second half of the 20th century had an impact on the use of digitals in architectural design. In his book, Difference and Repetition, Deleuze developed a theory of how forms come into existence—morphogenesis—and aimed to identify ways of novel creation. Deleuze argued that the forms we observe in reality are an "actualized" status of an idea and that the generative process is capable of producing other possibilities of what Deleuze identifies as "virtual" realities[3]. To Deleuze, an actualized form carries morphogenetic possibilities that have not yet been actualized. He further argued that this potential population of virtual design multiplicities precedes the singularity of actualized design and, therefore, the virtual is just as real as the actual.

Deleuze argued that the actualization of a form happens through the process of "'differen*c*iation". The result features "extensive"' qualities that give objects their

[3]Virtual reality is often used to describe "substitute reality", not to be confused with this concept; virtual reality here refers the space of possible ideas that can be actualized (Lynn, Kelly 1999).

distinct properties such as height or length. On the other hand, the process of "differen*t*iation" determines the virtual space of a design concept and includes "intensive" qualities. Intensive qualities refer to internal properties that cannot be changed unless the structure of the object is changed. Both "differen*t*iation" and "differen*c*iation" are processes by which an idea "incarnates" itself into the physical world.

Later, Deleuze's work made its way to architectural philosophy through the writings of Greg Lynn. Folding in Architecture by Greg Lynn (1993), which is based on Deleuze's Le pli, is considered one of the first attempts to theorize digital architecture. Lynn proposed the manipulation of formal representations using digital tools, fundamentally challenging the dominant representational logic of traditional architecture (Oxman, Oxman 2014).

Lynn's writings laid the foundations for the emergence of more specific theories centered on procedural processes and mathematical form generation that turn the focus from the "curvilinearity" and "blobby" forms of folding toward digital design thinking. These theories emphasize "formalism," or the "mechanisms" that govern the structure of relations within an architectural form rather than formal compositional aspects (Oxman, Oxman 2014, Kolarevic 2004). In other words, it is a shift from form "making" toward form "finding" (Kolarevic 2004).

3.3 MORPHOLOGY

Morphology is defined as "the scientific study of the forms" and emphasizes continuity and form mutation. The word was originally derived from the ancient Greek morphē and means form. However, morphology encapsulates the notion of animation[4]. According to the Oxford English Dictionary, morph as a verb means “change smoothly from one image to another by small gradual steps” (Stevenson 2010,P. 1151)

The interest in continuity in architecture was born as a response to deconstructionism (Oxman, Oxman 2014). In Folding in Architecture, Greg Lynn called for a reconsideration of the architectural form, replacing the “fragmented” with “fluidity” and at the same time taking advantage of advancements in computing technology of the nineties of the past century. In fact, Lynn’s writings laid the foundations for a series of publications concerned with theorizing the use of digitals in architecture.

Morphology is a term widely used in biology and refers to studying the form of living organisms and making connections between their structures. Morphology, and other relevant biological terms such as “genotype” and “phenotype”, were brought to architecture for the purpose of design exploration of different form configurations (Hillier, Hanson 1989, Steadman 1983). Two influential works by Albrecht Dürer (1528) and D’Arcy Thompson (1917) show the significance of morphological thinking. Dürer morphed an image of the human face through manipulating a hypothetical grid that he established, producing a series of faces with the aim of understanding how different forms related to each other. In On Growth and Form, D’Arcy Thompson compared the shapes of different species. In one example, he deformed the shapes of mammals’ skulls to transform one into the other. Thompson argued that there is something essential in all

[4] According to Greg Lynn, the difference between animation and motion is that motion emphasizes “movement and action” while animation is more about “evolution of a form.” (Lynn, Kelly 1999)

related forms that is not changed by the deformation, which he called “topological similitude” (Thompson 1917)

Topology is another term that is associated with morphology and is important in understating how new forms may relate to each other. Topology is more about structural relations and less about formal distinction. For instance, rectangles and squares are topologically equivalent, but both differ from triangles. Changing the length and width of the rectangle does not change its topology; however, adding or deleting a segment results in a topological transformation (Kolarevic 2004). Therefore, even if an actualized status of a particular form differs from that of another form, they may still have the same structural relations between their design components that cannot be identified though metric measurements.

In digital design, topological thinking is employed to produce design multiplicities that allow the exploration of a family of solutions by performing sequential transformations that produce a large number of shape variations, Digital tools allows initiating the process of geometric metamorphosis, which adds time to the process; thus, it becomes possible to express the “keyshape” of the geometry, or the state of the geometry at a particular point in time (Kolarevic 2004) This provides a convenient way to explore design variations.

Morphology and its related concept of topology are fundamental concepts in the digital design process; it helps not only in exploring design multiplicities and form optimization but also in understanding forms’ origin, evolution, and devolution, which are important concepts in both the research and design of forms.

3.4 PARAMETRIC DESIGN

Scholars (Oxman, Oxman 2014, Terzidis 2006, Woodbury 2010) have called for distinguishing digital architecture from digital design. While digital architecture aims to use the powers of computerization for building complex forms, it is still an “emulation” of conventional design tools—paper and pencil (Woodbury 2010). Digital design systems, on the other hand, shift the use of digital technologies from mere drafting tools to design thinking tools.

Parametric design is an approach to digital design. In a parametric design system, the designer establishes relationships between the design parts, manipulating them to generate infinite morphological variations (Oxman, Oxman 2014).

The process of parameterization involves initiating parameters and establishing relations rather defining a specific form (Kolarevic 2004). Parameters are values that have an effect on the design output. They can be variable or constant, simple or complex, and have a direct or indirect effect on the final output. Gradual change of variable parameters evokes the metamorphosis process. Thus, it becomes possible to examine the entire population of a particular design morphology.

3.5 MATHEMATICAL DESCRIPTION OF FORM

Arthur Loeb presented a method that exploit mathematics for describing the undelaying mesh of tessellations. The method is based on the number of “rotocenter”, the number of folds in each pattern. For instance, Loeb uses 2 2' to refer to a frieze structure and 3 3' 3" to refer to hexagonal structures. Each mesh produces symmetrical cells that can host geometric designs; Loeb’s study focuses on the holding structures. He aimed to show how mathematics and analytics advances the design process beyond what is "intuitively evident" and that mathematician and designers may complement each other's work (Loeb 2012, Loeb 1978).

Another work by Lionel March and Philip Steadman utilizes mathematics to describe forms. This work seeks an "economical" description of form. March and Steadman presented two descriptions: one for regular forms and one for irregular forms. The first description method utilizes the "grid system of quadrant" based on the Cartesian coordinate system (March, Steadman 1974). The description is a sequence of points that construct an architectural space, room by room, or building by building. For instance, R1=[25, 275; 0, 550] is a description of a room in a building. Here, R1 refers to the room. The following numbers refer to the location of the point in the quadrant grid. The first two points, 25 and 275, refer to the X-axis while 0 and 550 refer to Y-axis of the room. Subtracting X-axis values or Y-axis values from each other results in the width and length of the room, respectively.

To describe irregular forms, they proposed either inscribing the lines on the Cartesian coordinate system to position the constructing points or using the length of the line segments (r) and the angles between the lines (∅) to construct the form in the following way:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{r_1} & \mathbf{r_2} & \mathbf{r_n} \\ \emptyset_1 & \emptyset_2 & \emptyset_n \end{bmatrix}$$

Haresh Lalvani (Lalvani 1989) presented a "shape code" that describes Islamic geometry. He developed a grid system of fixed points with different "subdivisions" of a fundamental unit. He positioned geometric composition on the subdivisions of this grid to create designs. The parts of the subdivision involved in creating the design define the "shape code" of the particular design. Although the method establishes an interesting relationship between design components, the presented code is diagrams the relations between the points; with complex and highly segmented designs, writing the code comes closer to drawing the design that it describes.

3.6 SUMMARY

The use of digitals is changing ways of thinking about architecture; research and design are no longer about a specific, actualized form but rather the process that generates the form and is capable of generating morphological multiplicities. Knowing what is possible in the virtual space and comparing it to the actualized designs can reveal information about the selection process(Steadman 1983).

To discuss the morphology of IGP, a method that goes beyond the observed level and describes the virtual and actual space of design is needed. In other words, the method should allow the exploration of design morphology and symmetries and, at the same time, be capable of manipulating the actualized design. Thus, various states of a particular design can be detected, linked, and further examined.

CHAPTER FOUR

MIXED METHOD APPROACH

4.1 RESEARCH DESIGN

This research addresses the question of how to incorporate mathematics and morphology to describe the actual and virtual design space of IGP. It then utilizes this description, in light of historical evidence, to address the question of what are the morphological correlations between historic design singularities. Consequently, this research is descriptive and exploratory in nature. The objective is to describe IGP and then investigate similarities between historical IGP.

The concept of morphology operationalized in two dimensions: the actual and the virtual dimensions. The fixed design characteristics of historically existing hexagon-based IGP indicated mathematically using measurable attributes of points and line segments. The actual design space is utilized to derive a mathematical definition that encompasses the virtual design space and, in turn, is confirmed by utilizing this definition to code the historic singularities.

The design of this research is sequential. In phase one, a parametric description method is developed based on the examination of existing historic geometric patterns. The description aims to provide a unified method for describing existing IGP variations. The parametric description method is utilized to create databases of coded representations of historic designs. In the second phase of the research, content analysis is employed. The representational codes are compared to each other to identify similarities in both the actual and virtual dimensions.

4.2 PHASE ONE: PARAMETRIC DESCRIPTION

Simulation is defined as the representation of a system in reality using modeling (Marans, Stokols 2013, Groat, Wang 2002). A model, on the other hand, is also a system of “potential” reality (Marans, Stokols 2013 p.30). Systems generally consist of identifiable components that interact with each other to actualize reality. The “states” of a system at particular point in time are a single representation of that system (Marans, Stokols 2013 p.105). A simulation model captures all possible arrangements of the components within the system, or the “state history”, in a sequenced manner (Marans, Stokols 2013 p.195).

In this research, the system being represented is IGP, and the model used is the deterministic mathematical modeling (Groat, Wang 2002, Marans, Stokols 2013). Mathematical models, on the other hand, “capture real-world relationships in quantifiable abstract values” (Groat, Wang 2002 p.360). These are abstract models that adhere to “mathematical principles” (Marans, Stokols 2013 p.32). This research employs mathematical models to construct a unified parametric model that encompasses all possibilities and produces representational codes of IGP. The model is deterministic because it produces a unique “output” of IGP for each set of “input”.

The simulation system is constructed through the observation of reality and aims to provide a comprehensive representation of these realities (Groat, Wang 2002 p.352). Herbert Simon argued that a simulation model needs to consider the “agreed-upon assumptions and specifications” to ensure the accuracy of the representation and identify a “bounded domain of the system” (Groat, Wang 2002 p.367, Simon 1996 p.42). In the case of IGP, the bounded domain corresponds to the recognizable characteristics of symmetry, flow, and unboundedness (Abas, Salman 1995 p.4). These characteristics identify, in a general manner, the shared “agreed-upon” features of IGP.

The validity of the identified model is confirmed by comparing the result of the model with the “real world”; the simulation model can be “calibrated” and validated to maximize its ability to reflect reality (Marans, Stokols 2013 p.195). Consequently, the description method can be tested on historically existing designs (Groat, Wang 2002 p.365). Furthermore, the accuracy of the model depends to a high degree on the collected data. Therefore, the data collection process targeted all surviving identifiable hexagon-based patterns (further explained in section 4.4). However, investigating historical data requires considering “selective survival,” which refers to the fact that “some objects survive longer than others” due to the type of material used, which could cause loss of data (Singleton Jr, Straits & Straits 1993 p.411). To overcome this, the research takes advantage of the fact that the same IGP were implemented using different materials. Thus, the study accounts for the undelaying design—“ground geometry”—regardless of the materials used or the type of embellishment, which reduces the effect of systematic loss of a particular design due to its construction material.

Chapter 5 discusses the first phase in detail. In the second phase, the study utilizes mathematical model to develop representational codes of historic designs for the purpose of conducting content analysis.

4.3 PHASE TWO: CONTENT ANALYSIS

The second phase goal is to identify the morphological correlations between existing historical singularities of IGP. To this end, the second phase of this research employs representational codes to conduct content analysis. Content analysis is a useful method for examining textual and visual materials through providing a systematic technique that transforms materials into quantifiable data (Singleton Jr, Straits & Straits 1993 p.420). The process of conducting content analysis (i.e., identification of the "content categories," "recording units," and "system of enumeration") is discussed in Chapter 6 of this dissertation.

4.4 DATA COLLECTION

This research employs non-probabilistic purposive sampling that tracks surviving designs. The literature review played a central role in guiding the data search process. Chronologically, the period from the ninth to the 15th century is identified as the era of "invention" (Abas, Salman 1995 p.8). Geographically, close examination of the literature reveals that IGP were developed in the Abbasid Dynasty in Baghdad and Samara and then dispersed into other regions, later reaching the Mamluk, Timurid, and Nasrid Dynasties (Necipoğlu, Al-Asad 1995). Therefore, all designs that exist on monuments belonging to these dynasties were also considered.

This research focuses on hexagonal IGP. These types of geometric patterns have been widely used in the Islamic world since the early days of the patterns. Furthermore, Abas and Salman's study of symmetry showed that hexagonal IGP are the most frequently used periodic pattern (Abas, Salman 1995 p.138).

Data from historical buildings were collected through photographs gathered from books (Necipoğlu, Al-Asad 1995, Hill, Grabar 1967, Broug 2008, Broug 2013b, El-Said, Parman 1976), journal articles (Creswell 1919, Stronach, Young 1966), library archives (the Aga Khan Documentation Center at MIT and the Creswell Photographic Archive at The Ashmolean Museum), and authoritative websites (Archnet and dome websites). Appendix A shows the full list of the collected hexagonal patterns.

In the early stages, designs were limited in number, and the literature (Necipoğlu, Al-Asad 1995, Broug 2013b)identifies the monuments by their original names, dynasty, and geographic location. Multiple sources were examined, and the designs were collected and arranged chronologically. The literature discusses designs after the early stages in terms of the governing dynasty and mentions some monuments as examples and discusses particular designs. In these cases, all buildings that were constructed or renovated during a dynasty were examined. Further reading regarding the history of the building were pursued when necessary to identify the authenticity of designs.

The collection process resulted in a total of 273 designs collected from mosques, madrasa, hospitals, mausoleums, and palaces. Figure 4.1 (top) shows the geographic distribution of the collected data. In the same figure (bottom), a bar chart demonstrates the total number of designs collected from each region. The colors on each bar refer to the proportion of designs that belong to different dynasties. The figure on the bottom left shows the number of each collected design in relation to chronological period.

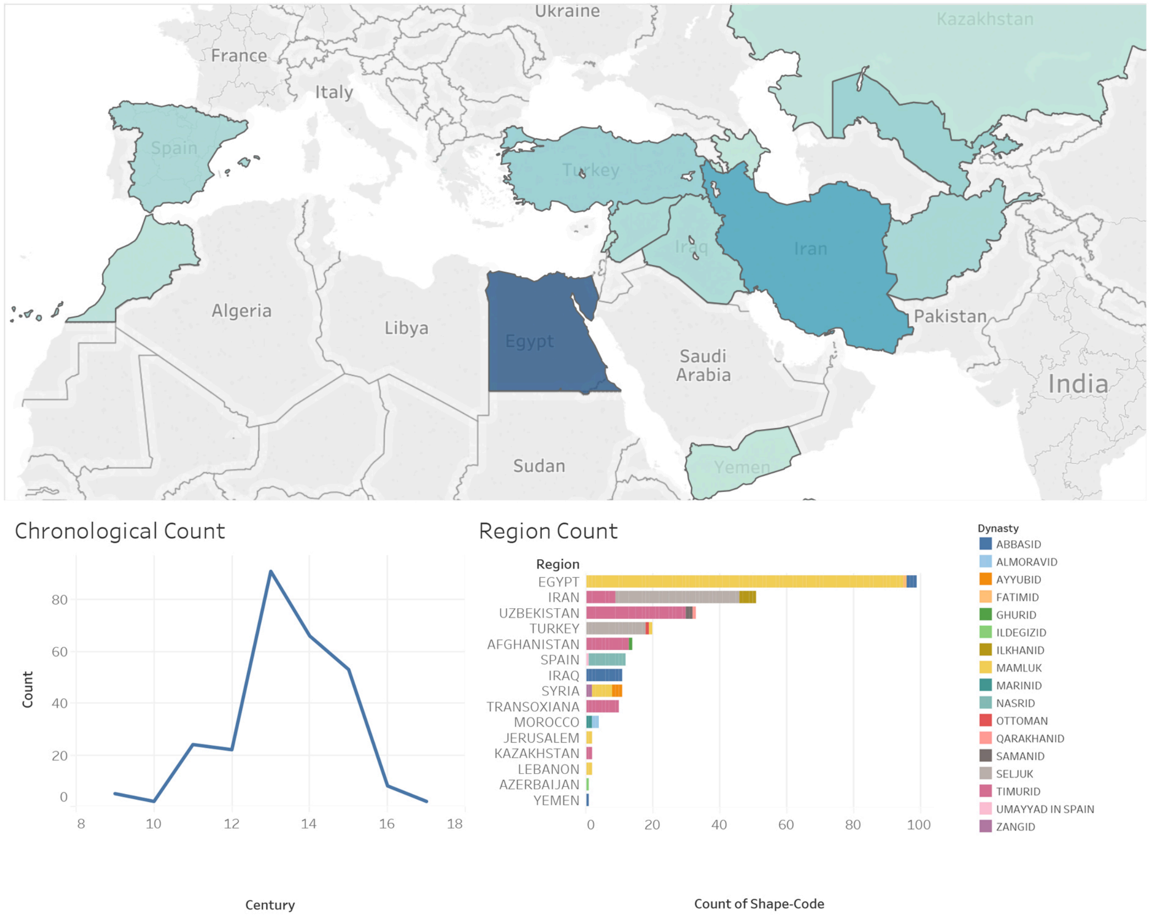


Figure 4.1 The geographic, chronological, and dynastic distribution of the collected data.

4.5 INSTRUMENTATION

The study collected the data in the form of digital photographs taken of either the IGP or the exterior or interior architectural surfaces of ancient monuments. In the latter case, hexagonal IGPs were identified in each photo and if more than one hexagonal geometric pattern existed, each was extracted in the form of a digital image. In some cases, more than one picture exists in different archives showing the same IGP. Thus, the surroundings of the IGP and erosion were examined to avoid confusion and inclusion of a single design more than once. Photographs were stored in a spreadsheet together with other information about designs, including dates, regions, towns, and governing dynasties.

Subsequently, the images were imported to the AutoCAD computer program as raster images and converted from digital photographs into CAD vectors. The process of conversion was conducted by tracing the photograph in AutoCAD. First, the geometric composition being repeated (“visual anchors”) was identified. Then, the geometric composition was bounded with a hexagon and repeatability to the neighboring cell was checked. Afterwards, the symmetry type and consequently the fundamental unit was identified (further explained in section 5.2, Chapter 5). Then, the geometry that fell into that fundamental unit was drawn and populated to the whole geometry. The next step involved checking the accuracy of the identified geometric composition by repopulating the hexagonal structures with the geometric composition. To increase accuracy when drawing the geometric patterns, the researcher referred, when possible, to the steps illustrated by Eric Broug(Broug 2008)for drawing whole designs or particular components. After the conversion process, each design was scaled so that each segment of the containing hexagonal unit was set to a length of 10 AutoCAD units.

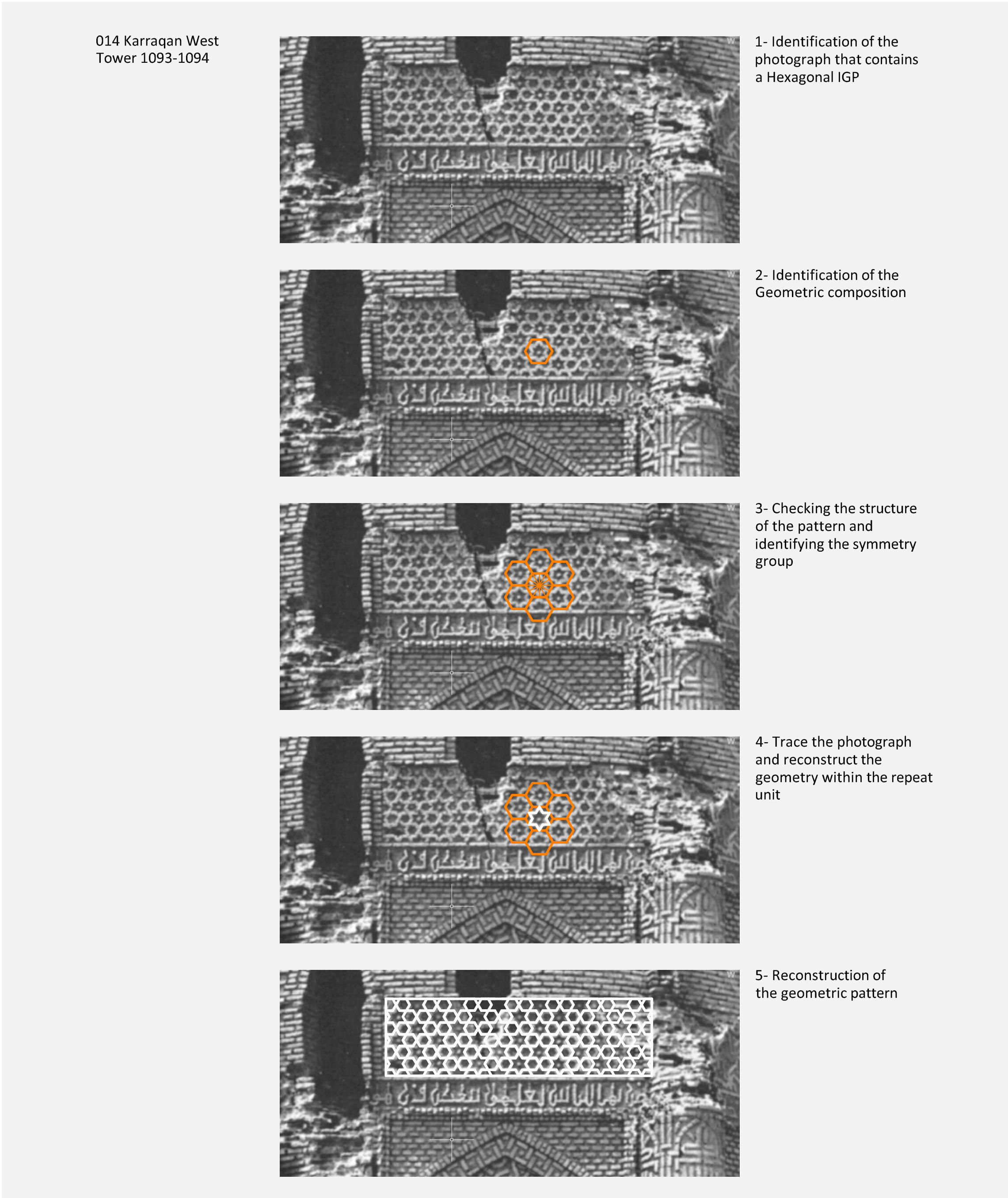


Figure 4.2: Conversation process from a digital photograph to CAD vector. Image used in this illustration is from David Stronach and T. Cuyler Young(1966).

CHAPTER FIVE

PARAMETRIC DESCRIPTION OF THE IGP MORPHOLOGY

5.1 INTRODUCTION

This chapter consists of three main sections. In the first section, geometric analysis concerning the understanding of the "reality" of the IGP is conducted. The second section discusses the development of the parametric morphological description. Lastly, the third part verifies the morphological description through the development of the simulation program.

5.2 ANALYSIS OF THE IGP

Periodic IGP consist of two main components: a repeat unit (RU) and a repetitive structure (El-Said, El-Bouri & Critchlow 1993a, Abas, Salman 1995). While the RU contains the primary geometric design to be populated, the repetitive structure stacks the RU to fill the space completely, leaving no gaps. Together, the RU and the repetitive structure determine the wallpaper symmetry group to which the pattern belongs.

Determining the wallpaper group is important for identifying the "fundamental unit" (FU). This unit represents the minimum geometric composition that is being systematically (Abas, Salman 1995 p.79). Thus, the employment of such a unit in the development of a geometric description produces shorter, more "economical" codes.
In the case of hexagonal patterns, there are five possible types of wallpaper groups: P3, P3M1, P31M, P6, and P6M[5]. Abas and Salamn identified the following steps by which the group can be distinguished (Abas, Salman 1995 p.108):

[5] 'The International Crystallographic Notation' was employed to label different types of symmetry groups.

- In the case of three-fold geometric designs, the existence of reflection symmetry within the FU must be checked:
    - If a reflection does not exist, the symmetry group is P3.
    - If a reflection does exist, the center of rotation needs to be confirmed and there are two possible cases:
        - If the 'center of rotation' appears only on reflection lines, the wallpaper symmetry group is P3M1.
        - If the 'center of rotation' not appears on reflection lines, the wallpaper symmetry group is P31M.
- In the case of six-fold geometric designs, the existence of reflection symmetry within the FU must be checked, and there are two possible scenarios:
    - If reflection symmetry does not exist, the symmetry group is P6.
    - If reflection symmetry does exist, the symmetry group is P6M.

Figure 5.1 explains the identification process of the FU. This procedure was applied to the collected data, and it was found that 93.43% of the collected IGPs fall within the P6M category (figure 5.2).

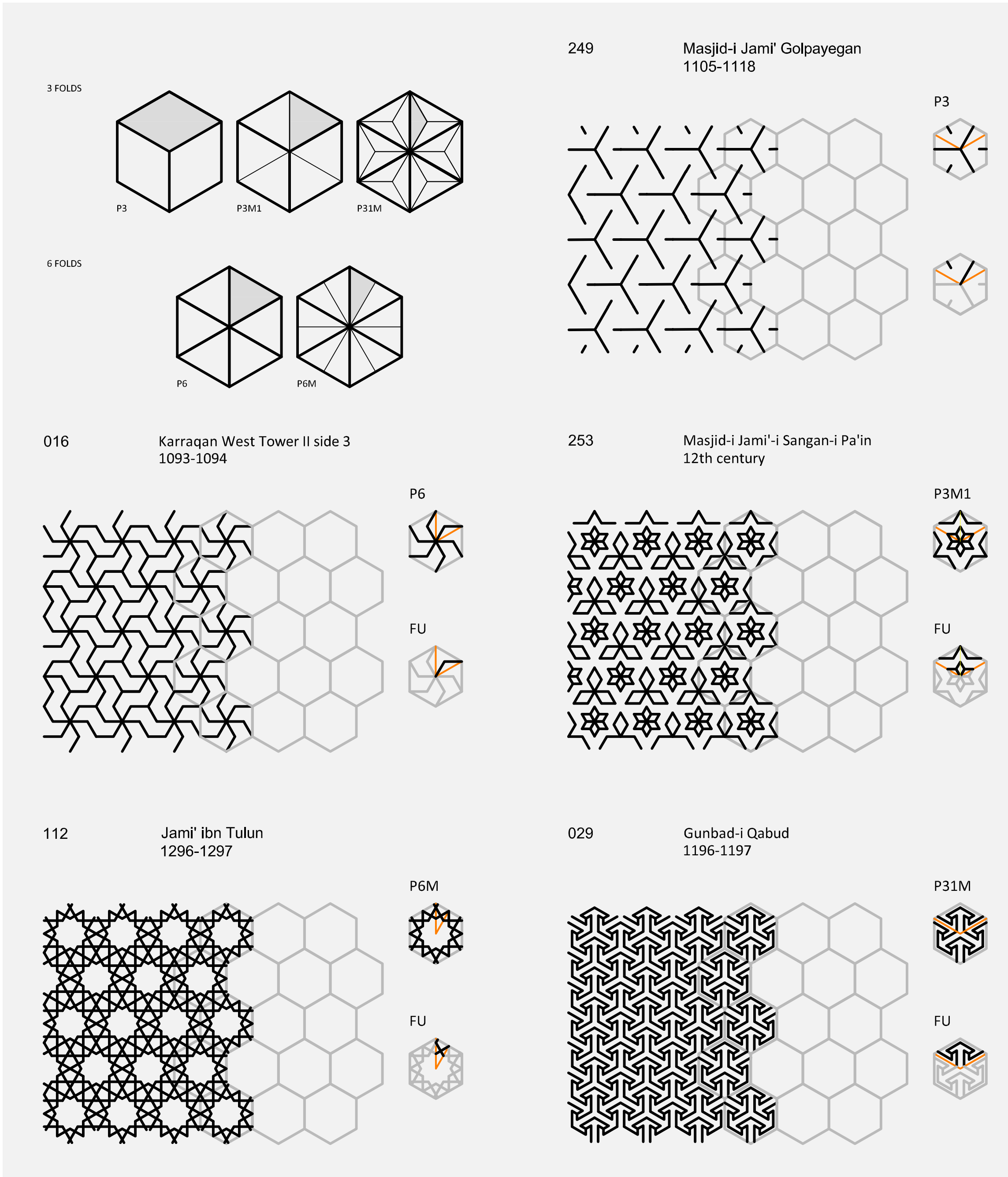


Figure 5.1: Examples of each of the five types of the symmetry wallpaper groups and the process of identifying the FU. Top left explains the FU by itself.

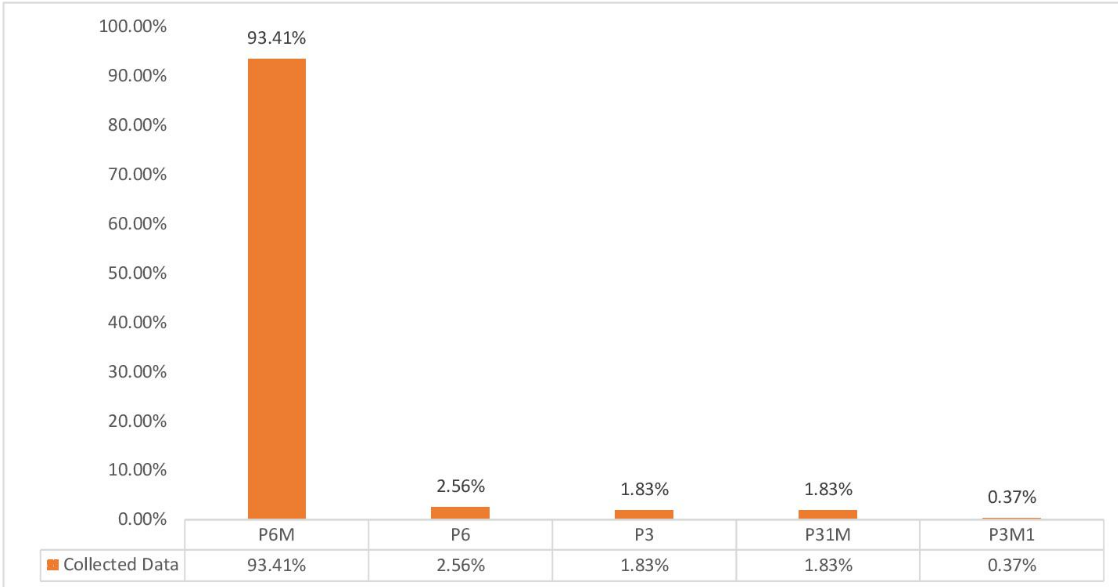


Figure 5.2: Percentage of each type of the five hexagonal wallpaper group symmetry categories in the collected data.

Having determined the symmetry group, the FU can be obtained. By identifying the points of intersection between the geometric component's lines and the intersection of the geometric lines with the hypothetical boundaries of the fundamental unit, the design can be decomposed into points series connected by line segments. The following point types can be identified (figure 5.3):

- Single connection point (SP): points in this category connect only to one another, forming a single segment within the FU.
- Double connection point (DP): points in this category connect to two other points, forming two segments within the FU.
- Triple connection point (TP): points in this category connect to three other points, forming three segments within the FU.

- Quadruple connection point (QP)[6]: points in this category connect to four points, forming four segments within the FU.

These points and their relationships are considered the basic constructional components of the morphological description.

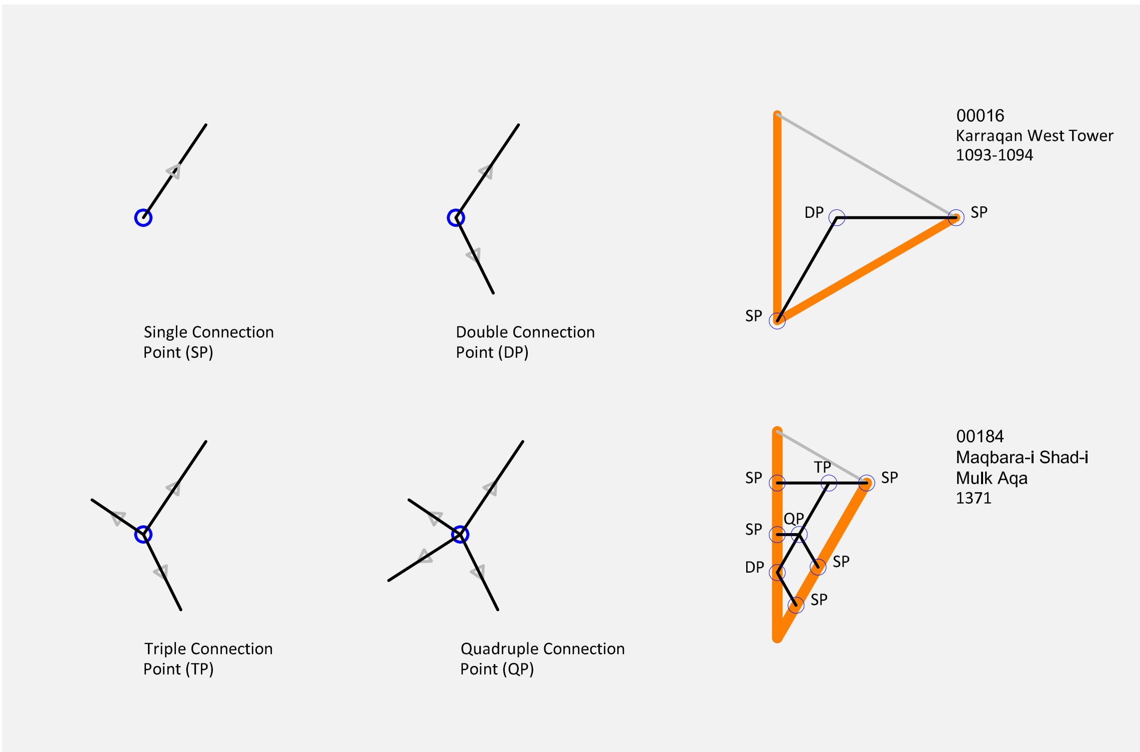


Figure 5.3: The illustrations on the left explain point categories. The illustrations on the right show two examples with different point types.

5.3 THE MORPHOLOGICAL DESCRIPTION

The morphological description exploits symmetry information and constructional components—points and their relationships— that fall within the FU to develop a

[6] Hereafter, triple and quadruple connection points referred to as T/QP.

“deterministic” simulation model that “outputs” geometric patterns based on an “input”’ of representational code, ultimately capturing the “reality” of the IGP in a numeric form.

The constructional components within the FU can be represented mathematically by referring to each of the constructional points in a sequenced manner in a similar way to Lionel March and Phillip Steadman’s method (March, Steadman 1974 p.182 & p.190). In general, and based on the design within the FU, two scenarios were identified: single and multiple sequence(s) of straight lines, hereafter referred to as polyline(s). In the first scenario, the FU contains only a single polyline that can be described by listing all of the constructional points that fall within the FU in a sequenced manner. For instance, the design that exists within the FU of the Ibn Tulun mosque (shown in figure 5.4) can be represented as:

$$\mathbf{PL_1 = [P_1\ P_2\ P_3]}$$

where PL refers to the polyline and P refers to the constructional points. The square brackets indicate the beginning and the end of a single polyline. However, this description only represents the geometry within the FU. To populate the description to the RU and the structure, symmetry information should be added. Thus, the previous code can be rewritten as:

$$\mathbf{P6M: PL_1}$$

Similarly, PL can be substituted by a list of the constructional points:

$$\mathbf{P6M: [P_1\ P_2\ P_3]}$$

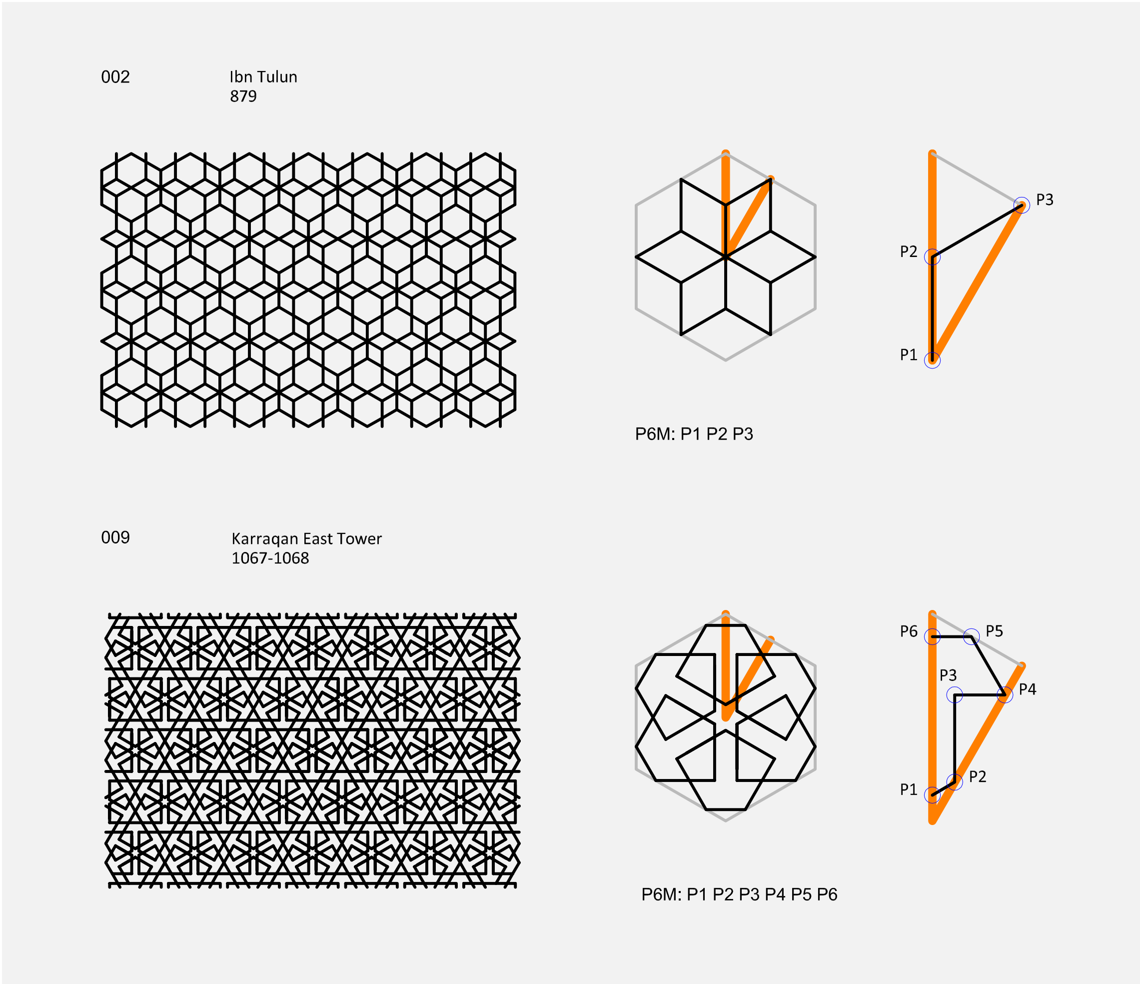


Figure 5.4 The top shows geometry from Ibn Tulun mosque, and the bottom shows geometry from the Karraqan East Tower. Both geometries are shown in terms of the whole pattern, the repeated unit, the fundamental unit, and the representational code.

Designs with more constructional points can be coded in a similar fashion as shown in the Karraqan East Tower geometry in figure 5.4 (bottom). Further, if the polyline closes on itself at any point but remains as a single polyline, the design can still be described in a similar way, yet the shared point is addressed twice in the description as it appears. For instance, the geometry found in Alhambra Palace (figure 5.5) can be described in the following possible ways:

$$\mathbf{P6M}: [\mathrm{P_1\ P_2\ P_3\ P_4\ P_5\ P_6\ P_3\ P_7\ P_8}]$$

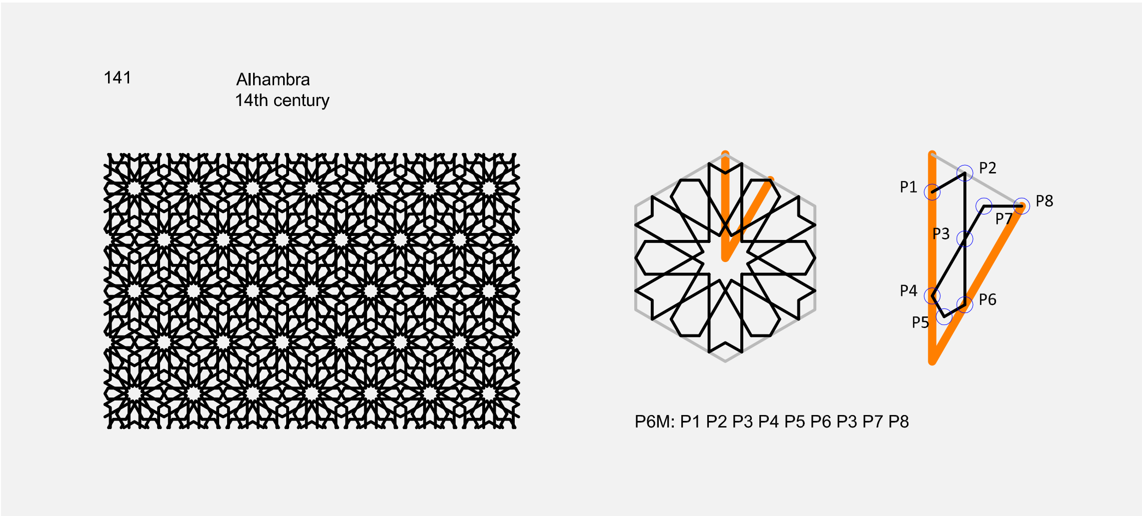


Figure 5.5: Single polyline scenario that closes on itself in a quadruple connection point.

Here $\mathbf{P_3}$ is listed twice, and in this particular instance, $\mathbf{P_3}$ is a quadruple connection point (QP)[7].

If there are multiple polylines within the FU (second scenario), each polyline is described by listing all points in a sequenced manner. If a shared point exists between two polylines, the point is addressed in each list. For instance, the geometry shown in figure 5.6 (top) can be described as:

$$\mathbf{P6M}: [\mathrm{P_1\ P_2\ P_3\ P_4}]\ [\mathrm{P_5\ P_3\ P_6}]$$

[7] The description code can be also written as: $\boldsymbol{P6M}: [\boldsymbol{P_1\ P_2\ P_3\ P_6\ P_5\ P_4\ P_3\ P_7\ P_8}]$, where the sequences of points that fall between the quadruple connection point are reversed. Chapter 6 addresses the sorting algorithm that was employed to identify all possible descriptions.

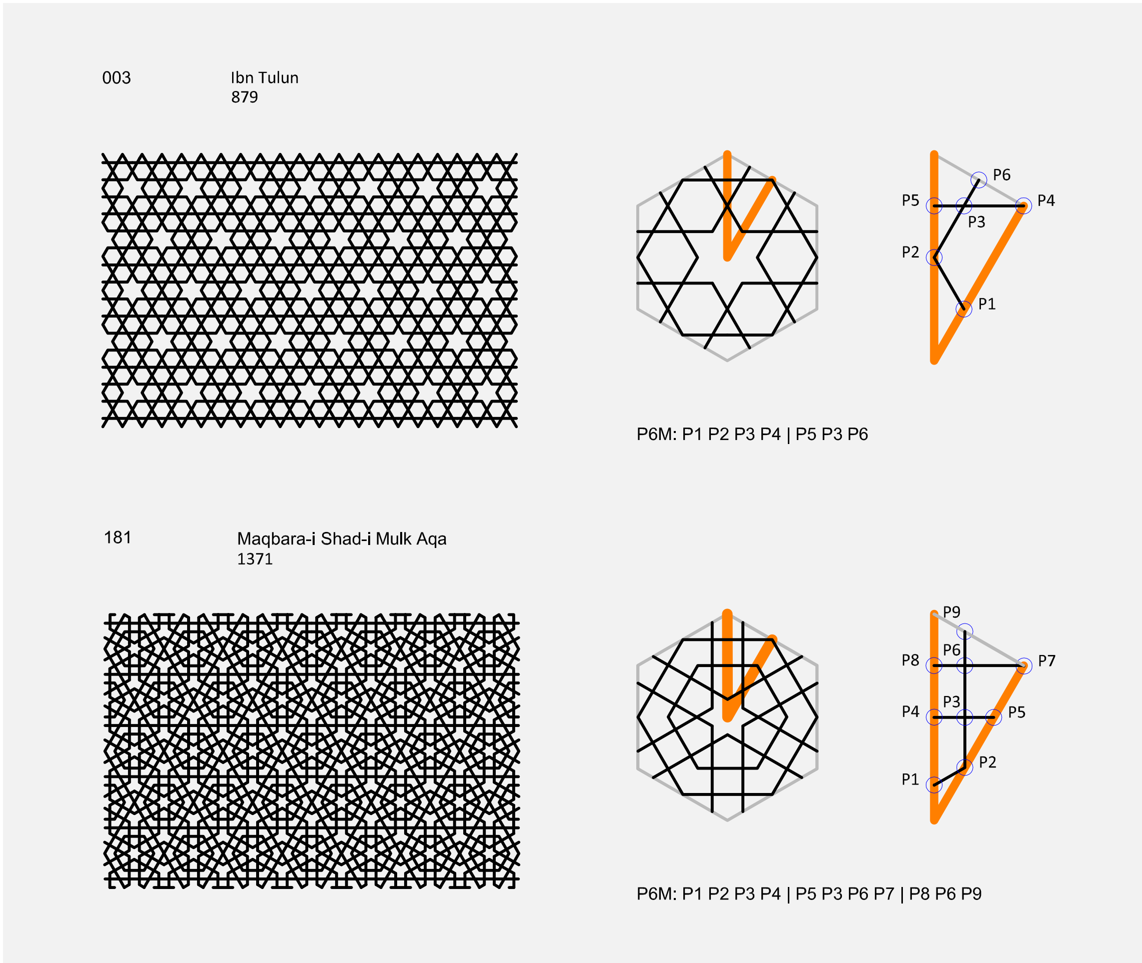


Figure 5.6: Multiple polyline scenarios. The top shows a design with two polylines within the FU, and the bottom shows a design with three polylines within the FU.

Furthermore, the actualization of the design requires clearly defining the exact location of the constructional points. These points can be defined using their coordinates on the Cartesian coordinate system. Therefore, the code for the design in figure 5.4 (top) can be expressed as:

$$\mathbf{P6M}\text{:}\ \begin{bmatrix} \mathbf{x}_{\mathbf{P}_1} & \mathbf{x}_{\mathbf{P}_2} & \mathbf{x}_{\mathbf{P}_3} \\ \mathbf{y}_{\mathbf{P}_1} & \mathbf{y}_{\mathbf{P}_2} & \mathbf{y}_{\mathbf{P}_3} \end{bmatrix}$$

Similarly, the design in figure 5.6 (top) can be expressed as:

$$\mathbf{P6M}: \begin{bmatrix} \mathbf{x}_{\mathbf{P_1}} & \mathbf{x}_{\mathbf{P_2}} & \mathbf{x}_{\mathbf{P_3}} & \mathbf{x}_{\mathbf{P_4}} \\ \mathbf{y}_{\mathbf{P_1}} & \mathbf{y}_{\mathbf{P_2}} & \mathbf{y}_{\mathbf{P_3}} & \mathbf{y}_{\mathbf{P_4}} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathbf{P_5}} & \mathbf{x}_{\mathbf{P_3}} & \mathbf{x}_{\mathbf{P_6}} \\ \mathbf{y}_{\mathbf{P_5}} & \mathbf{y}_{\mathbf{P_3}} & \mathbf{y}_{\mathbf{P_6}} \end{bmatrix}$$

Alternatively, polar coordinates can be utilized. These have proven to be more convenient due to the fact that they enhance the parametric aspect of the description by granting independent controls for the angle and distance for each point in a meaningful way (figure 5.7). For instance, changing how far a point is from the center of the RU requires manipulating only one parameter, while in the Cartesian coordinate system method, two inputs are required to reach the same output. Here, the distance of each point in the design is measured from the center of the RU; the angle between the distant line and the hypothetical horizontal line that passes through the center is also measured (figure 5.7). Therefore, the previous description of figure 5.4 (top) can be rewritten as:

$$\mathbf{P6M}: \begin{bmatrix} \mathbf{r}_{\mathbf{P_1}} & \mathbf{r}_{\mathbf{P_2}} & \mathbf{r}_{\mathbf{P_3}} \\ \emptyset_{\mathbf{P_1}} & \emptyset_{\mathbf{P_2}} & \emptyset_{\mathbf{P_3}} \end{bmatrix}$$

Similarly, the design in figure 5.6 (top) can be expressed as:

$$\mathbf{P6M}: \begin{bmatrix} \mathbf{r}_{\mathbf{P_1}} & \mathbf{r}_{\mathbf{P_2}} & \mathbf{r}_{\mathbf{P_3}} & \mathbf{r}_{\mathbf{P_4}} \\ \emptyset_{\mathbf{P_1}} & \emptyset_{\mathbf{P_2}} & \emptyset_{\mathbf{P_3}} & \emptyset_{\mathbf{P_4}} \end{bmatrix} \begin{bmatrix} \mathbf{r}_{\mathbf{P_5}} & \mathbf{r}_{\mathbf{P_3}} & \mathbf{r}_{\mathbf{P_6}} \\ \emptyset_{\mathbf{P_5}} & \emptyset_{\mathbf{P_3}} & \emptyset_{\mathbf{P_6}} \end{bmatrix}$$

In this code, r refers to a point's respective distance value from the origin while $\emptyset$ refers to the value of the respective angles in which the points are located (Figure 5.7). In this research, the values are measured within a hexagonal RU, with each side of the hexagon measuring 10 units. Figure 5.8 shows more examples with the actualized values. Hereafter, these codes are referred to as representational codes.

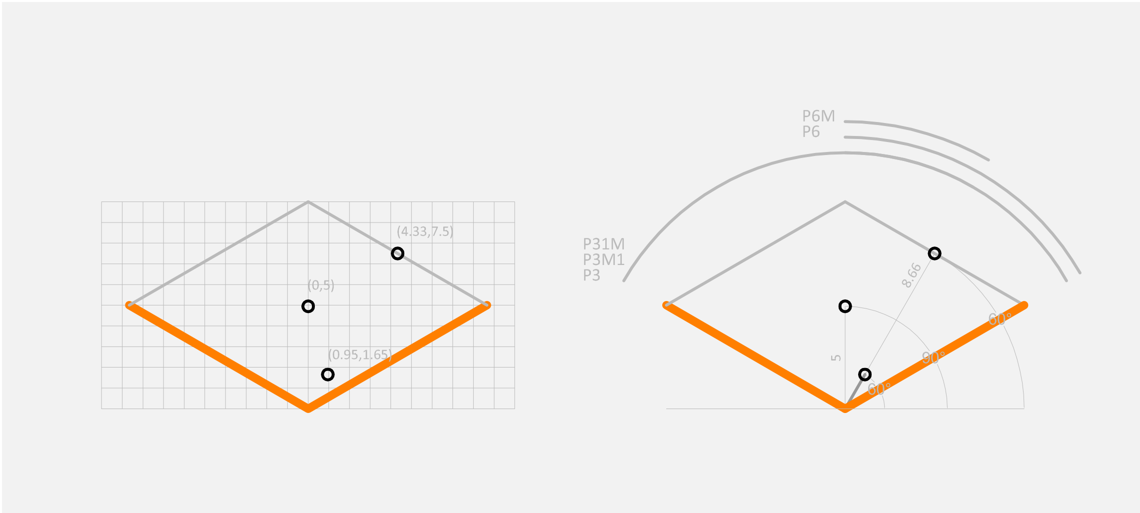


Figure 5.7: The process of point actualization. The Cartesian coordinate system vs. the polar coordinate system.

To develop a description model that represents the virtual space of IGP, the study employs abduction reasoning, where the derivation process of the description model moves from the actualized designs to construction of a model that encompass all IGP possibilities. This also aligns with the philosophical argument of Gilles Deleuze, who argued that actualized designs still carry "morphogenetic possibilities" within them. Thus, the coding process was carried out for all of the 273 collected designs to extract a description model that represents IGP morphology. Figure 5.9 shows an identified polylines behavior pattern that exists within a historical IGP. That is, a design can be constructed from at least a single polyline with at least two constructional points forming a single segment within the FU.

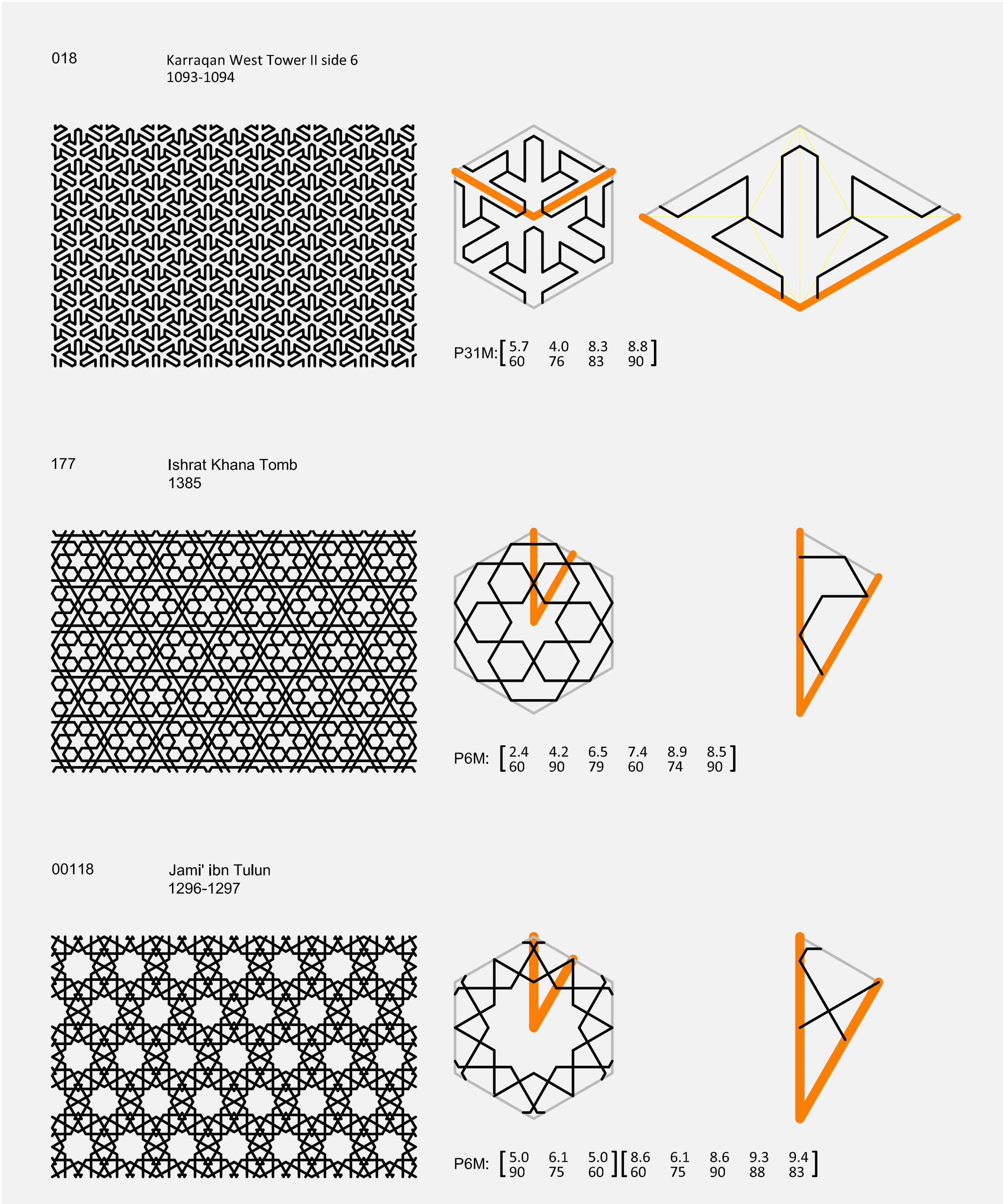


Figure 5.8 The pattern on left can be generated using the representational codes on the bottom right of each design.

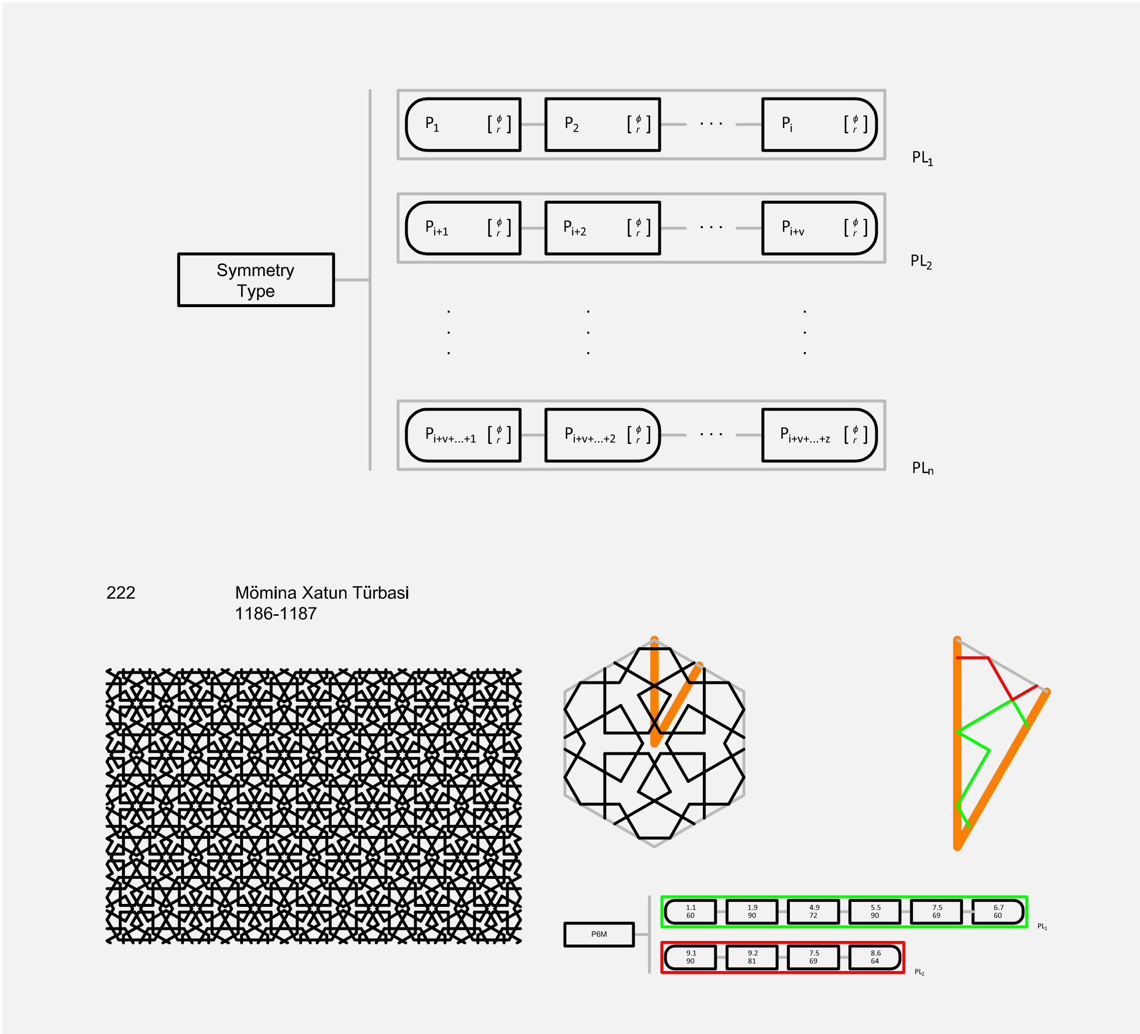


Figure 5.9 The top diagram shows the identified behavior pattern and parametric aspects of the IGP. The bottom is a historic example that demonstrates actualization of the description model above.

Any parametric description model that represents IGP needs to capture all scenarios and have the ability to be expand and to contain more polyline(s), while preserving the sequence of the constructional points and providing actualization information for each of the constructional points. Therefore, the code that captures the virtual morphological design space of an IGP can be expressed as:

$$\textbf{Symmetry Group}: \begin{bmatrix} \mathbf{r}_{\mathrm{P}_1} & \mathbf{r}_{\mathrm{P}_2} & \cdots & \mathbf{r}_{\mathrm{P}_\mathrm{i}} \\ \emptyset_{\mathrm{P}_1} & \emptyset_{\mathrm{P}_2} & \cdots & \emptyset_{\mathrm{P}_\mathrm{i}} \end{bmatrix} \begin{bmatrix} \mathbf{r}_{\mathrm{P}_{\mathrm{i}+1}} & \mathbf{r}_{\mathrm{P}_{\mathrm{i}+2}} & \cdots & \mathbf{r}_{\mathrm{P}_{\mathrm{i}+\mathrm{v}}} \\ \emptyset_{\mathrm{P}_{\mathrm{i}+1}} & \emptyset_{\mathrm{P}_{\mathrm{i}+2}} & \cdots & \emptyset_{\mathrm{P}_{\mathrm{i}+\mathrm{v}}} \end{bmatrix} \cdots$$

$$\begin{bmatrix} \mathbf{r}_{\mathrm{P}_{\mathrm{i}+\mathrm{v}+\cdots+1}} & \mathbf{r}_{\mathrm{P}_{\mathrm{i}+\mathrm{v}+\cdots+2}} & \cdots & \mathbf{r}_{\mathrm{P}_{\mathrm{i}+\mathrm{v}+\cdots+\mathrm{z}}} \\ \emptyset_{\mathrm{P}_{\mathrm{i}+\mathrm{v}+\cdots+1}} & \emptyset_{\mathrm{P}_{\mathrm{i}+\mathrm{v}+\ldots2}} & \cdots & \emptyset_{\mathrm{P}_{\mathrm{i}+\mathrm{v}+\cdots+\mathrm{z}}} \end{bmatrix}$$

where symmetry group in the above description refers to the symmetry type of the pattern; *i* refers to the total number of points in the first polyline; *v* refers to the total number of points in the second polyline; and *z* refers to the total number of points in the nth polyline.

5.4 THE SIMULATION PROGRAM

The quality of a description is determined by its ability to reflect reality. Therefore, to verify the ability of the code to describe the IGP, the researcher specifically developed a simulation program that reads the representational code and visualizes the design (figure 5.10). The inputs to the program are the representational code and the outputs are the visual images in the processing “display window”. Further, the program outputs a DXF file that can be imported to AutoCAD and compared with the associated design.

When running the program, the user is promoted to enter the representational code. After pressing the execute button, the code *string* of the input code is divided into two parts and stored in an *array* of *string*. The first *index* stores the symmetry type, and the second *index* contains points and their relations. The second part is later converted into another *array* of *string* that stores each point in the form of the angle and distance in a single *array index*. Next, the *function* that is responsible for drawing and displaying the code is called. This *function* reads each *index* of the second *string array* and converts it into a *float array* while performing a loop in the *array* to draw each point. If the end of a polyline indicates, the program skips a segment and thus establishes a new polyline if necessary. Subsequently, the IGP is displayed on the screen, and the user can export the geometry in a DXF format. Appendix B shows the program script.

Furthermore, the program provides additional morphing functionality that performed through changing the values of the representational code and redrawing the design as figure 5.10 shows.

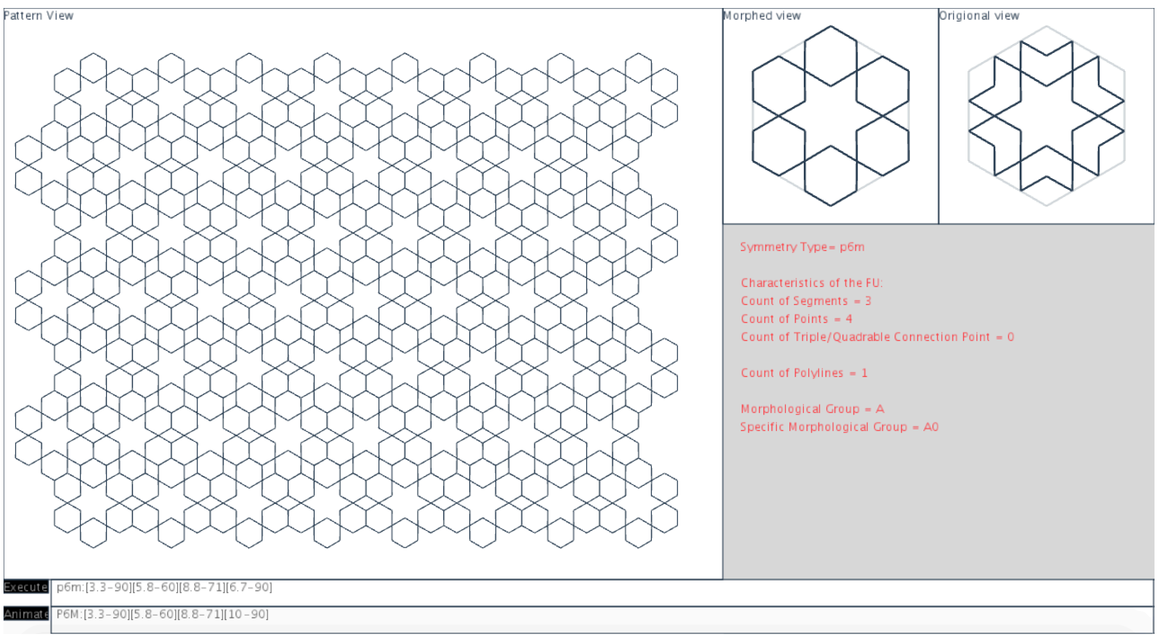


Figure 5.10 Explanation of the interface of the IGP explorer (the simulation program).

5.5 PILOT STUDY: MANIPULATING THE PARAMETRIC DESCRIPTION

A preliminary version of the parametric description method presented in this chapter was published in the Conference Proceedings of the 20th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA). In that paper, values within the representational code of the historical designs were manipulated, and new codes were derived. Figure 5.11 illustrates the morphing process of the geometry originally existing in Ibn Tulun mosque. When the results of the morphing process were compared with the representational codes of historic IGP, it was found that some of the newly derived codes exactly matched historic designs in other regions (figure 5.12). Therefore, two types of morphological correlations between the historic designs were identified: identical designs and structurally equivalent designs, which the study further investigates in the following chapter.

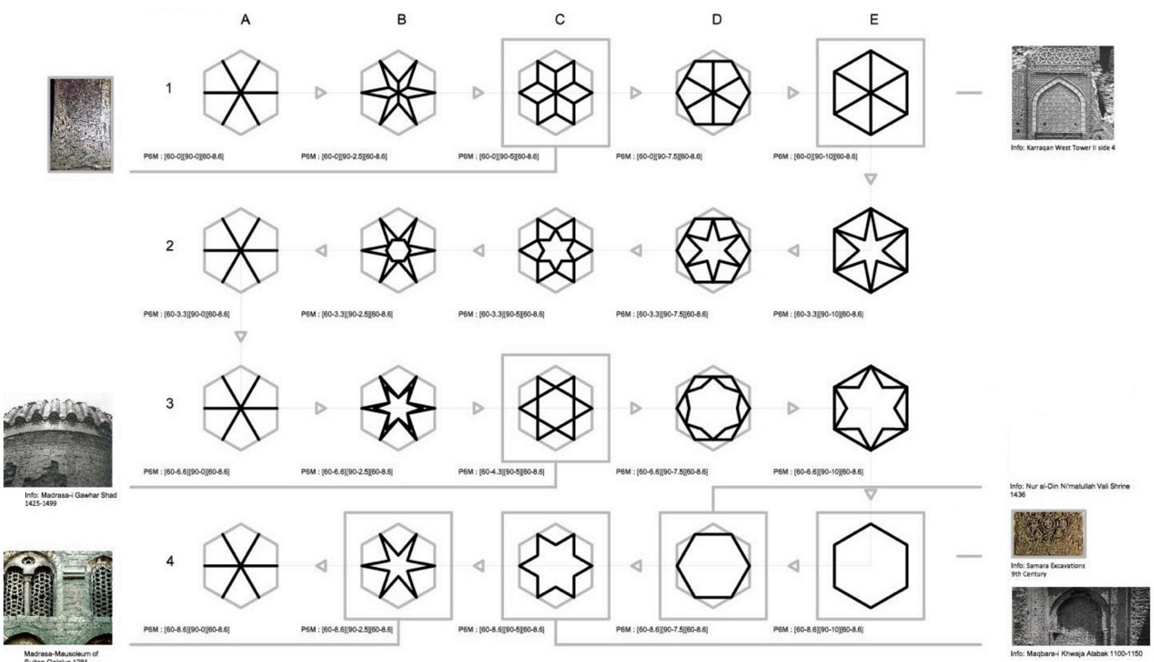


Figure 5.11 Following the arrow, this figure explains selected transformations of Ibn Tulun geometry through various topological states (Wade 2015, Stronach, Young 1966, Burckhardt 2009)

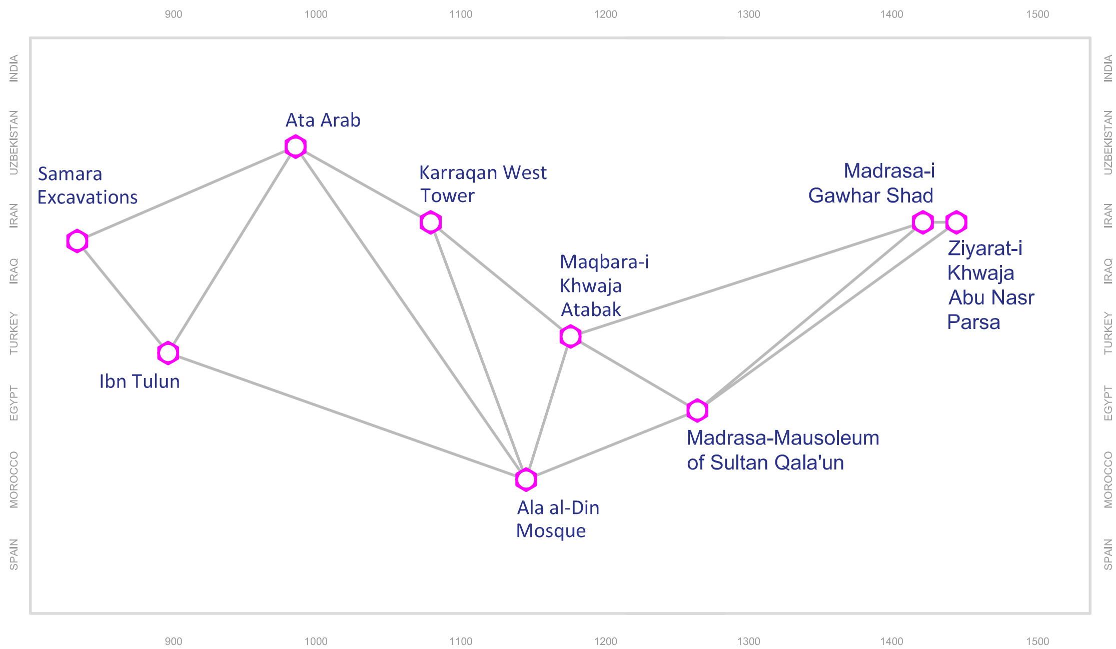


Figure 5.12 Across region morphological correlations among historical designs. The x-axis represent time, and the y-axis represents the geographic location arranged in an ordinal manner from west (bottom) to east (top).

CHAPTER SIX

THE MORPHOLOGICAL CORRELATIONS

6.1 INTRODUCTION

The goal of this chapter is to identify the morphological correlations that exist among various historical hexagon-based IGP. This chapter utilizes the developed representational code and begins by grouping hexagonal IGP into five groups based on the number of polylines that exist within the FU. Next, the chapter establishes the content category, discusses the search algorithms employed to investigate the representational codes, and presents the results of each content category. Finally, the chapter discusses the identified morphological correlations.

6.2 THE MORPHOLOGICAL GROUPS

The representational codes of historic designs were examined and, by counting the total number of polylines within the FU of each design, the collected data can be categorized into five groups as follows:

- MORPHOLOGICAL GROUP A (single polyline): The sequence of points forms a single polyline within the FU. Of the 273 examined IGP, 168 designs fall within this category. This category can be further subdivided based on the number of T/QP within the FU into six specific morphological groups (SMG): A0, A1, A2, A3, A4, and A8. Here, the letter A refers to the number of polylines, and the following number indicates the total number of the identified T/QP within the FU. The most frequent SMG is A0, with 152 designs (see figure 6.2 for examples of all SMG).

- MORPHOLOGICAL GROUP B (double polylines): The sequence of points forms two polylines within the FU. A total of 88 designs fall within this category. Moreover, this category can be further subdivided into the following SMG: B0, B1, B2, B3, B4, and B6. B1 is the most frequent SMG, with 62 designs (figure 6.2).
- MORPHOLOGICAL GROUP C (triple polylines): The sequence of points forms three polylines within the FU. Only 15 cases fall within this category. The group can be subdivided into the following SMG: C1, C2, C3, and C5. C2 is the most frequent design, with 10 cases.
- MORPHOLOGICAL GROUP D (quadruple polylines): The sequence of points forms four polylines within the FU. Only one case from Madrasa al-Mustansiriyya was identified (figure 6.1).
- MORPHOLOGICAL GROUP F (sextuple polylines): The sequence of points forms six polylines within the FU. Only one case from Alhambra Palace falls within this category (figure 6.1).

Overall, 61.54% of the collected designs falls within group A, and 32.23% falls within group B. Only 6.23% falls into categories beyond two polylines. Figure 6.1 shows examples of each morphological group and presents the total number of cases in each SMG.

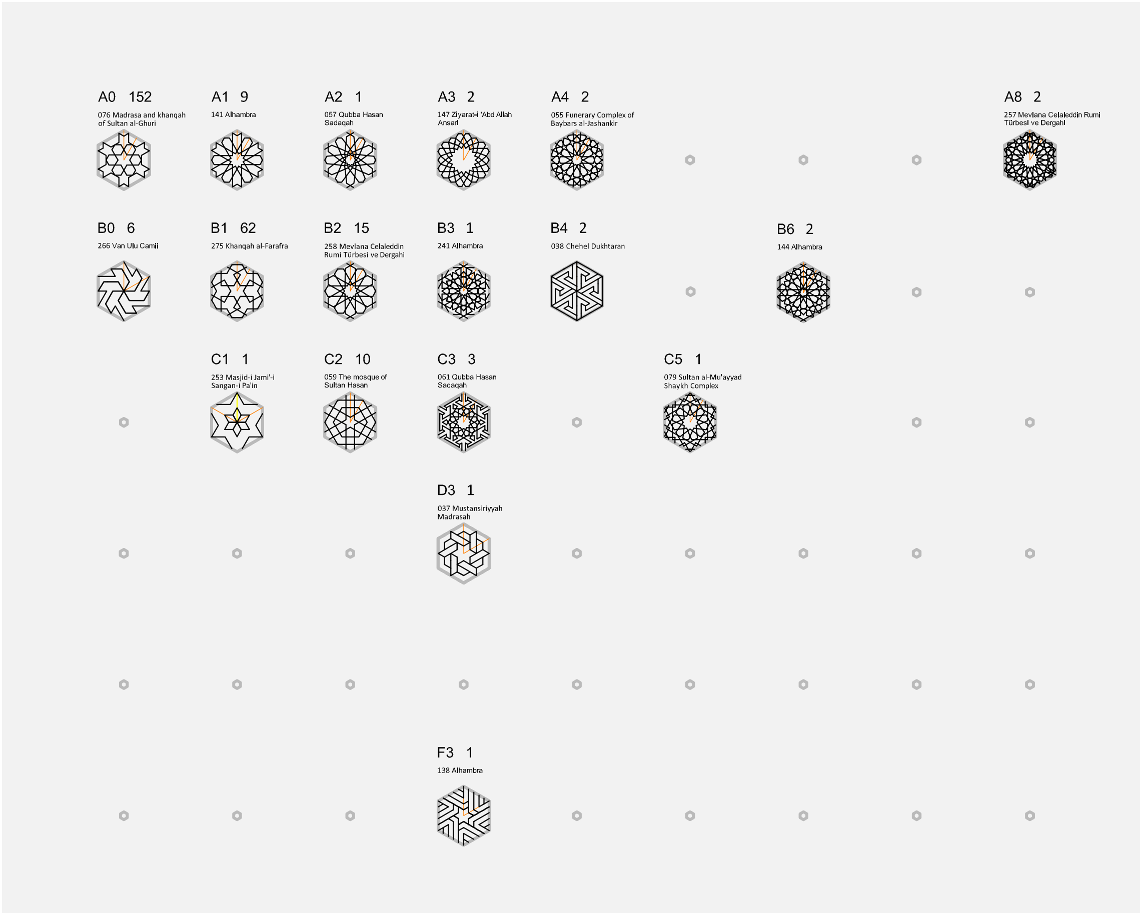


Figure 6.1: Examples of all SMG. *The y-axis represents the MG (number of polylines). The x-axis represents the total number of Q/TP within the FU. The intersection of the two axis defines the SMG. The intersection is represented by an example from the associated SMG. The frequency of each SMG is shown in orange at the top right of each geometry.*

6.3 THE CONTENT CATEGORIES

The identification of the content category is driven by the following questions: What is the frequency of the replicated designs in the collected historical IGP? Furthermore, does a structurally equivalent design exist? If yes, what is the frequency of such designs? Consequently, two main categories based on Deleuzian's actual-virtual conceptual framework were established: identical and structural equivalency. The identical category is concerned with identifying replicated designs; therefore, the recording unit in this category is the full match of the representational codes between the compared designs.

The structural equivalency category is concerned with identifying the existence of shared morphological configurations among historical designs. This category is further subdivided into four levels that each has its own recording unit. The representational codes of the actualized designs were examined on several levels in this category, moving gradually from the actual dimension toward the virtual design dimension. At each level, the comparison between the representational codes considered specific variables that have connections to the actualized dimension; in the subsequent level, fewer connections to the actualized dimensions were considered, moving gradually toward the virtual dimension (table 6.1).

			Indicators				
			Values of distance parameter	Values of angle parameter	Points Sequences	Specific Morphological Groups	Morphological Groups
Actual dimension	Identical Designs		✓	✓	✓	✓	✓
Virtual Dimension	Levels of structural equivalency	LV1	×	✓	✓	✓	✓
		LV2	×	×	✓	✓	✓
		LV3	×	×	×	✓	✓
		LV4	×	×	×	×	✓

Table 6.1: Variables considered in each content category. The check marks refer to the considered variables when comparing the recording units of the investigated designs.

6.4 THE SEARCH ALGORITHMS

A search algorithm was developed for each of the above content categories that compares an input of representational code with the database of the historically existing hexagonal IGP and output types of existing correlations. However, before the comparison process can take place, the representational codes must be sorted. Although each representational code always refers to a single output, a single IGP design can have more than one possible representational code that describes the design depending on the possible ways to sort the sequence of the constructional components. In the case of intersection, the sorting algorithm defines the possible paths that each polyline can take. Therefore, before comparing the codes using the matching algorithm, all possible representational codes must be identified. This step is important to control any coding inconsistencies caused by the researcher in regard to coding similar designs in a reverse order or the identification of polyline paths.

The following section discusses the sorting algorithm. Following this, matching algorithms and the results for all category are presented.

6.5 SORTING ALGORITHMS

Searching for identical designs or designs that are structurally equivalent of level one or two (these levels are discussed in the following section 6.6.2) requires the comparison of value and sequence of points information. Therefore, a sorting algorithm was developed for the following specific morphological groups: A0, A1, B0, B1, and B2. As multiple designs that share the same segment count exist within each group, these designs can be considered candidates for an identical or level one or two structural equivalency categories.

On the other hand, the A2, B4, C5, D3, and F3 groups include only one design, so it can be concluded that these designs cannot have identical designs or a level one or two structural equivalency. Other groups, A3, B3, A8, and C1, include designs that have a different number of segments; here, it is also possible to conclude that these designs cannot have an identical designs or a level one or two structural equivalency as the length of the representational code of the designs that falls into the same groups is different. The researcher was able to identify two identical designs for both the A4 and B6 groups. Group C3 includes both identical designs and designs that are different in segment numbers.

Group C2 has 10 designs that fall within four levels of segments: one design of 12 segments, two designs of ten segments, five of eight segments, and two of seven segments. The two designs of ten segments each fall within different symmetry groups and are thus neither identical nor structurally equivalent. For the five designs that have the eight segments, four were found to be identical, and the fifth falls within a different symmetry group. Only the two designs of seven segments required sorting. The researcher controlled the sorting of the codes by writing it in a selected predefined sequence. This is primarily because there are two cases on which to test the sorting algorithm.

Table 6.2 explains segment availability for each morphological group. The following sections discuss the sorting algorithm for A0, A1, B0, B1, and B2 groups.

MORPHOLOGICAL GROUP (MG)	SPECIFIC MORPHOLOGICAL GROUP (SMG)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	18	20	21	22	23	29	36	37
A	A0	55	49	30	6	10	2																
	A1					1		1	5	2													
	A2													1									
	A3													1			1						
	A4																			2			
	A8																					1	1
B	B0		2	3		1	1		1														
	B1			1	6	20	11	10	11	1													
	B2						1	1	1	2	2	6	2										
	B3													1					1				
	B4												1										
	B6																				2		
C	C1					1																	
	C2							2	5		2		1										
	C3														1	2							
	C5																	1					
D	D3									1													
F	F3										1												

Segment#

Count of Shape-Code 1 – 55

Table 6.2 The number of designs identified in each morphological group, broken down by the total number of segments in each design.

A0 SMG:

As discussed earlier, group A refers to a hexagonal IGP with a single polyline within the fundamental unit. The number "0" refers to the absence of any T/QP. In this case, there are only two possible ways to sort the representational code: 1) starting from P1 all the way to Pn, where n refers to the last point in the description; 2) the reverse of the first code, which is starting from Pn all the way to P1 (Figure 6.2).

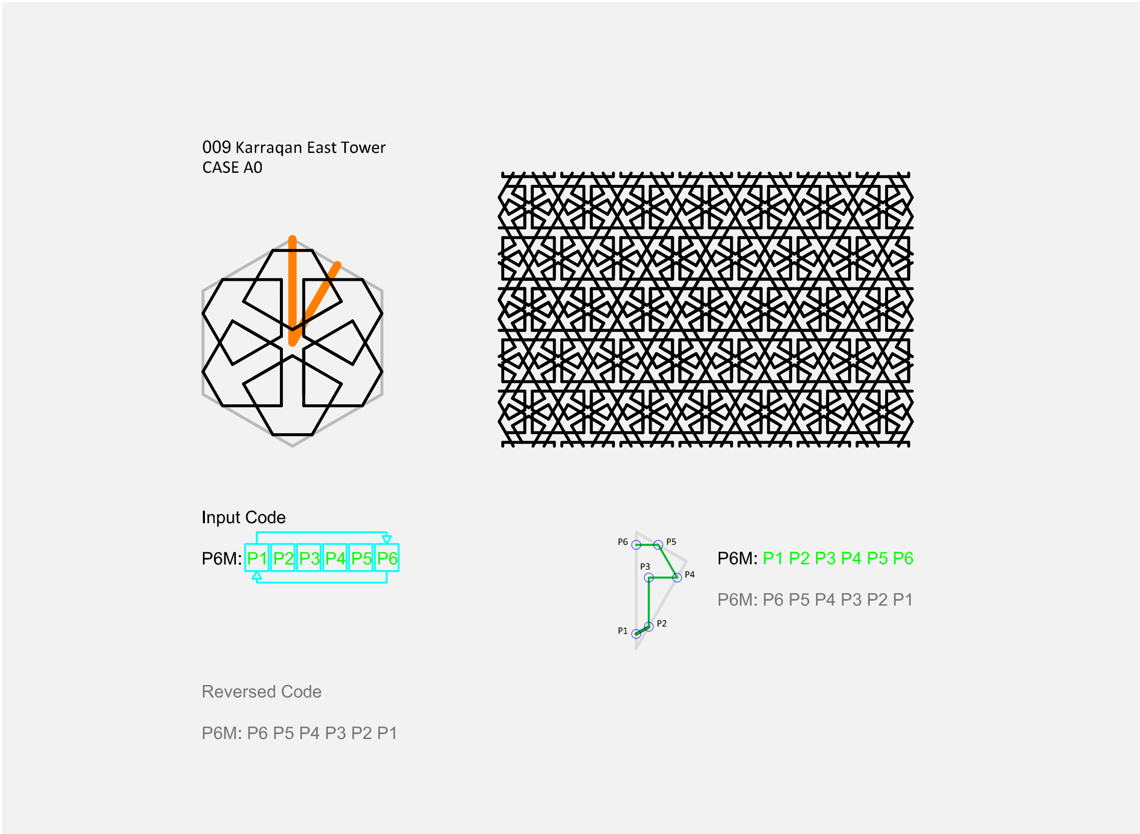


Figure 6.2 Sorting of the A0 SMG. The bottom right shows the two possible sortation of codes that fall within the A0 group

A1 SMG:

Similarly, A1 refers to the existence of a single polyline within the fundamental unit that has a single T/QP. Therefore, there are two possible paths for the polyline (shown on the bottom of figure 6.3). Consequently, there are four ways to write the code: 1) sorting the representational code starting from P1 all the way to Pn, where n refers to the last point in the description; 2) the reverse of the first code, 3) reversing the sequence of points contained between the T/QP following the steps below:

- Identify the T/QP, and

- Reverse the order of the constructional points that fall in between the T/QP (in figure 6.4, this step changes the path of the polyline from FU 1 to FU 2);

and 4) fourth possible representational code can be obtained by reversing the sequence of points in the third code.

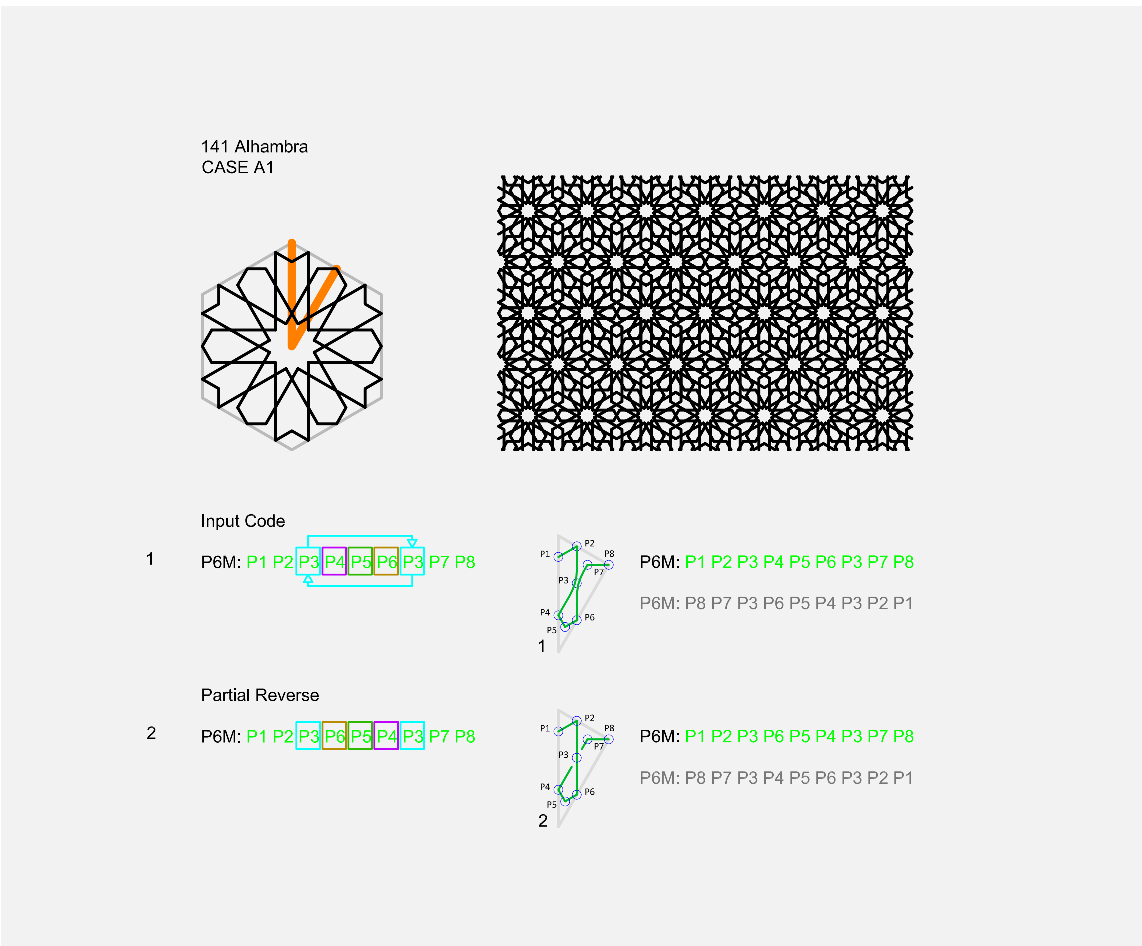


Figure 6.3 Sorting of the A1 SMG. The representational code of the design in Alhambra Palace. The starting code refers to code inputted by the researcher. Partial reverse refers to the process of partially flipping the highlighted point sequence (colored boxes).

B0 SMG:

In this group, there are two polylines and no T/QP. Therefore, no path determination is required; however, flipping the sequence of points within each polylines is required. The following steps were utilized to identify the possible codes: 1) the first possible code is the input code (i.e., the initial representational code coded by the researcher); 2) reversing the input code; 3) reversing the first polyline in the input code while keeping the other polyline in the original state; 4) reversing the previous case; 5) reversing the second polyline in the input code while preserving the original sortation of the first 6) reversing the previous case 7) reversing both polylines in the input code while preserving their order (i.e., the first polyline followed by the second polyline); and 8) reversing the previous case (figure 6.4).

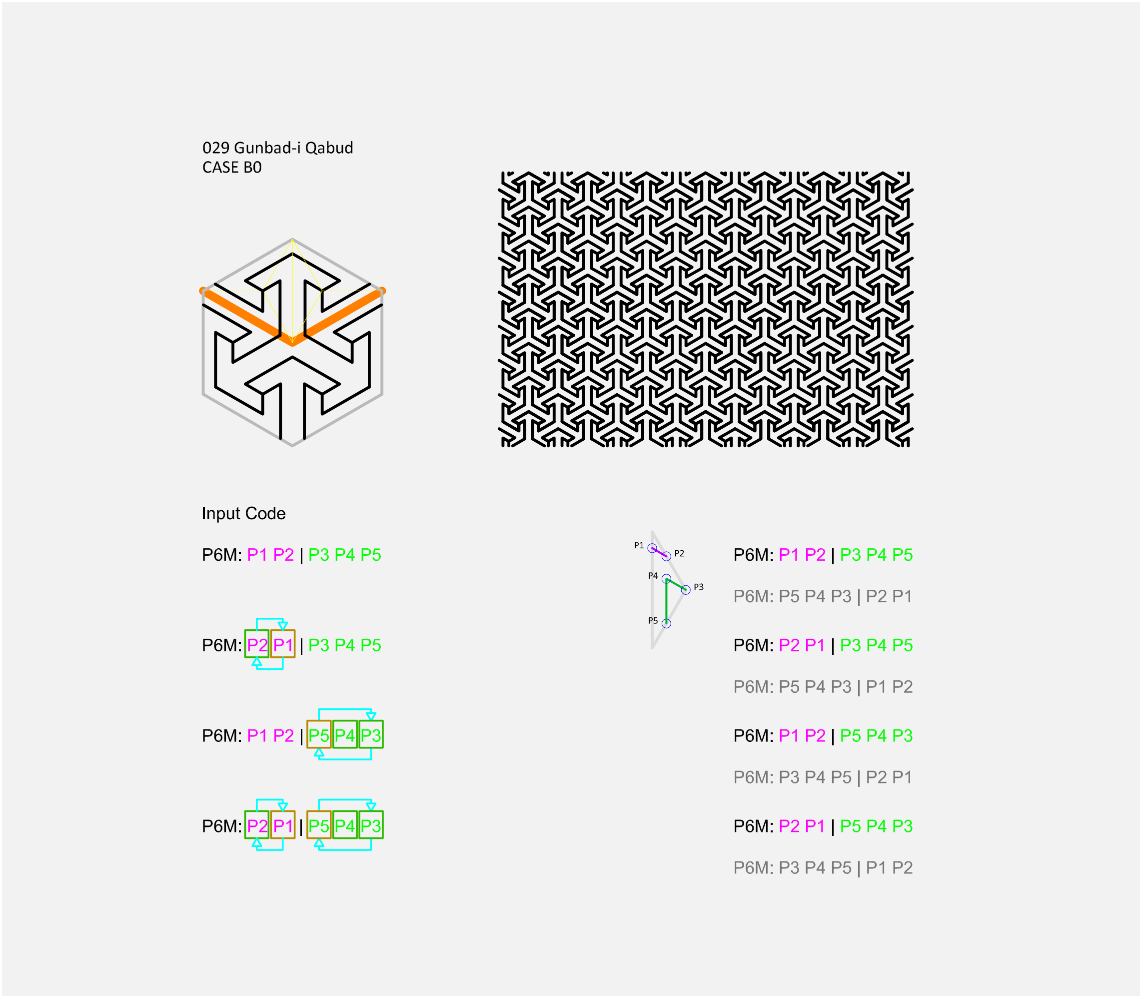


Figure 6.4 Sorting of the B0 SMG. The bottom left shows the flipping process for each polyline, and the bottom right shows the possible description codes for B0 group.

B1 SMG:

In this case, there are two polylines and a single TP or QP. Each polyline can take more than one path; therefore, to identify all possible paths that the two polylines can take, the following steps were followed, starting from an input code (figure 6.5):

- Identify the shared point between the two polylines (the delimiter) and divide each polyline into two parts—the first part, which is located before the delimiter point, and the second part, which is located after the delimiter point.
- Identify the second possible paths by beginning with the input code and switching the first part of the first polyline with the second part of the second polyline and the second part of the first polyline with the first part of the second polyline (see 1 and 2 in figure 6.5).
- Identify the third possible paths by beginning with the input code and switching the first part of the first polyline with the first part of the second polyline and the second part of the first polyline with the second part of the second polyline (see 1 and 3 in figure 6.5).
- After these steps, each code of the paths (including the input code) can be treated as B0 to further derive all possible codes.

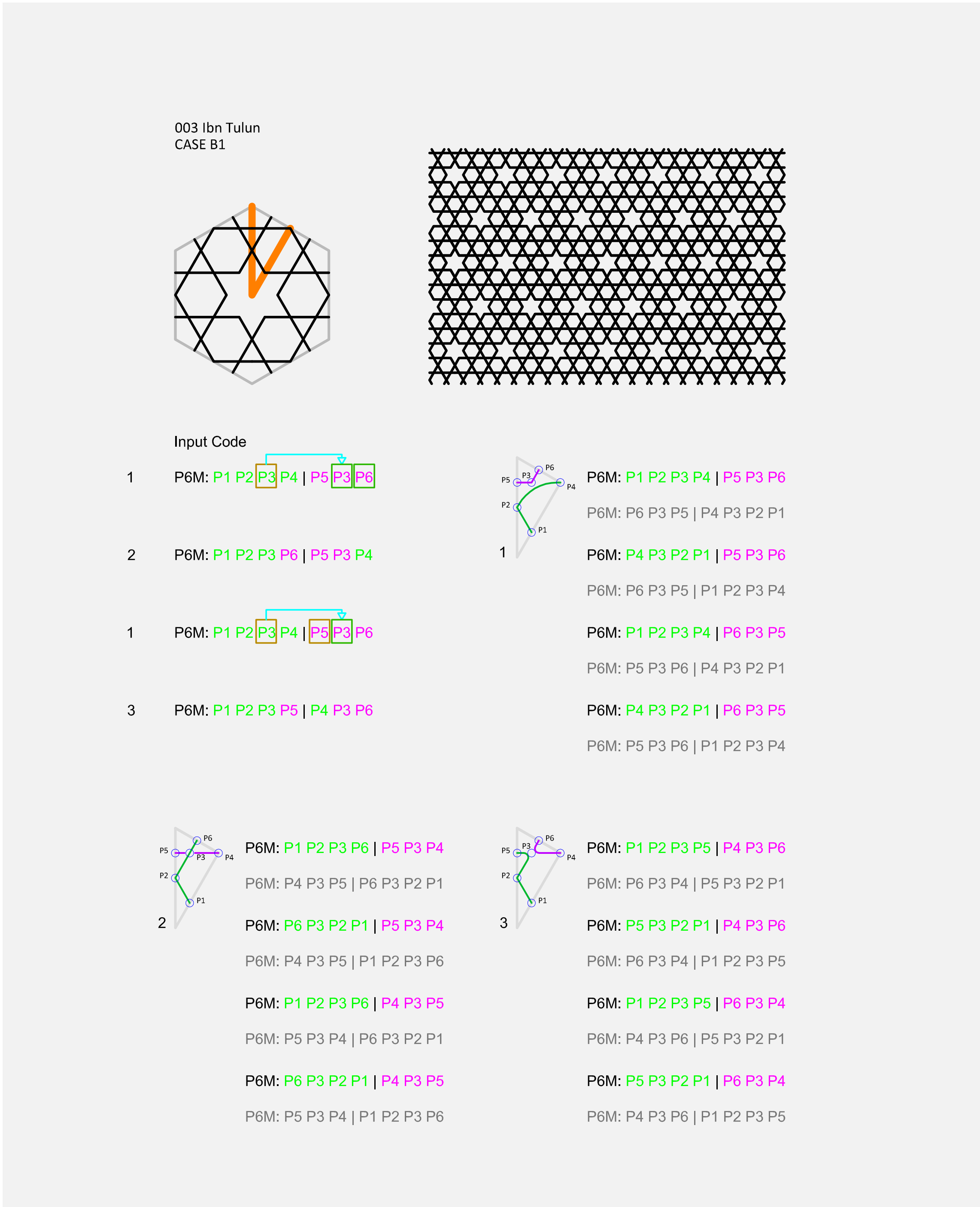


Figure 6.5 Sorting of the B1 SMG. Explanation of the possible paths for designs of B1 specific morphological group.

B2 SMG:

In this case, there are two polylines and two T/QP. The sorting for this group requires the identification of the possible paths of the two polylines. The following steps were followed to identify the main paths that the two polylines can take:

- Identify both T/QP.
- Use the T/QP that exist in both polylines as a delimiter to rearrange the polylines in a fashion similar to B1.
- Use the two resulting representational codes from the last process to rearrange the polylines in a fashion similar to B1, this time using the other T/QP, the second delimiter.
- At any point, if either T/QP listed twice in a single polyline, rearrange the constructional points in that polyline in a fashion similar to the A1 sorting method.

If both T/QP exit on the two polylines in at least one configuration, this sorting methods yields eight possible paths (figure 6.6). Otherwise, the result is six possible paths (only two cases were identified in the later scenario: a design found in *Jami' ibn Tulun* mosque, number 100 in appendix A, and a design found in *Imaret of Ibrahim Bey* of *Konya*, number 234 in appendix A).

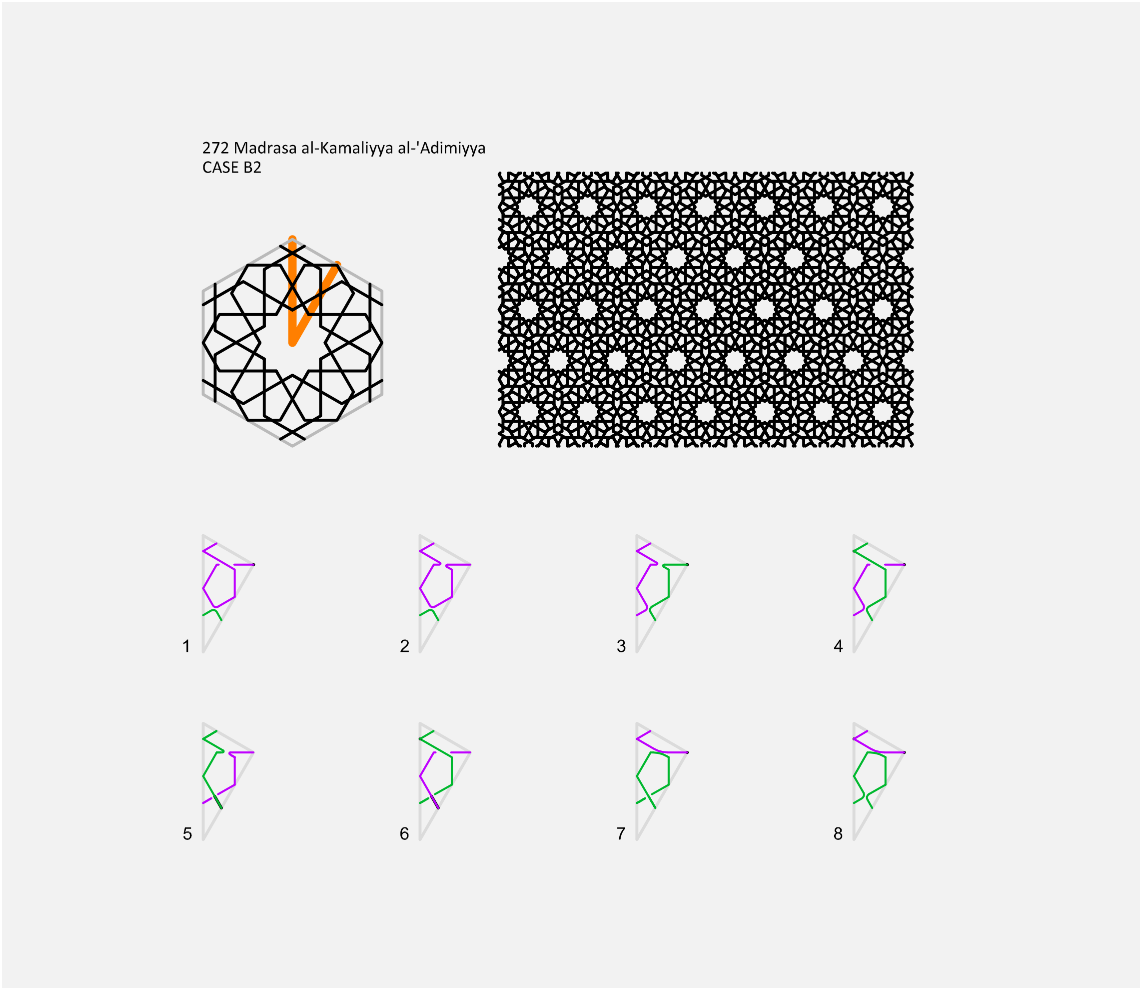


Figure 6.6 Sorting of the B2 SMG. The possible interpretation of the two polyline paths.

6.6 MATCHING ALGORITHMS

Matching algorithms compare two representational codes and return the type of the morphological correlations that exists between the two compared designs. All possible codes from previous section were considered in the comparison. A matching algorithm was developed for each of the content categories. The following sections present the matching algorithms and the result for each category in terms of frequency of occurrence.

6.6.1 IDENTICAL MATCH

This category determines the frequency of occurrence of the replicated hexagonal IGP. The recording unit in this category is the full match of the representational code of the compared designs. The code of each IGP in the collected data is compared with the other 272 designs. If the code matches another design, the two designs are labeled as identical. Figure 6.7 shows the implemented codes.

```
void IdenticalMatch(Geometry g, Geometry[] G){
  // A loop through the array of Geometry object.
  for(int i=0; i<G.length; i++){
    // Compare the representational code of the input IGP with each geometry in the array
    if(g.repCODE.equals(G[i].repCODE) && g.CAD != G[i].CAD && g.SYMMETRY.equals(G[i].SYMMETRY)){
      // Only if a match exist, print the number of the design.
      print(G[i].CAD+",");
    }
  }
}
```

Figure 6.7 The code used for identifying identical designs, implemented using processing programing language.

Based on the examination of 273 designs, 181 were found to share their representational codes with at least one other design while 92 designs were not replicated. To find the percentage of designs that share representational code, the identical designs grouped were together and each group was counted as one design. Therefore, the total number of unique designs becomes 138, and the percentage of replicated designs becomes 33.33% (46 designs) (table 6.3).

	Replicated	Percentage of Replicated	Not Replicated	Percentage Not Replicated	Totals
Grouped	46	33.33%	92	66.67%	138
Ungrouped	181	66.30%	92	33.70%	273

Table 6.3: Unique (grouped) designs vs. ungrouped designs.

The earliest identified copied designs found in Karraqan East Tower from 1067 CE in Iran (figure 6.7 top). The design was copied from an earlier design that existed in 977 at the Ata Arab from in Uzbekistan. This design was found later in three other monuments: Rasd-khaneh-i Ulugh Beg in Uzbekistan (1420 CE), Aramgah-i Shah-i Zindeh in Transoxiana (1434 CE), and Ishrat Khana Tomb in Uzbekistan (1464 CE).

The most frequently copied design, however, is the star design, which originally existed in the West Karraqan Tower (1093 CE) and was then replicated in 23 locations between the 12th and 17th centuries in various regions (figure 6.7 middle). The most frequent design in B1 group is a design that first existed at Masjid-i Jami' Golpayegan in Iran (1105 CE) and was later found in 11 other locations between the 12th and 15th centuries (figure 6.8 bottom).

6.6.2 STRUCTURAL EQUIVALENCY

In this category, the search for morphological correlations departs from the identification of identical forms to the search for matches in the internal arrangements of the constructional components of the compared historical designs.

The matching process is implemented in four levels. The levels are ordinal in nature and span Deleuzian’s actual-virtual extremes. In each level, the search is constrained by specific conditions that make connections to the actualized dimension; in each following level, fewer connections to the actualized dimensions were considered, moving gradually toward the virtual dimension (table 6.1 shows the considered variables in each level). The levels are discussed in the following:

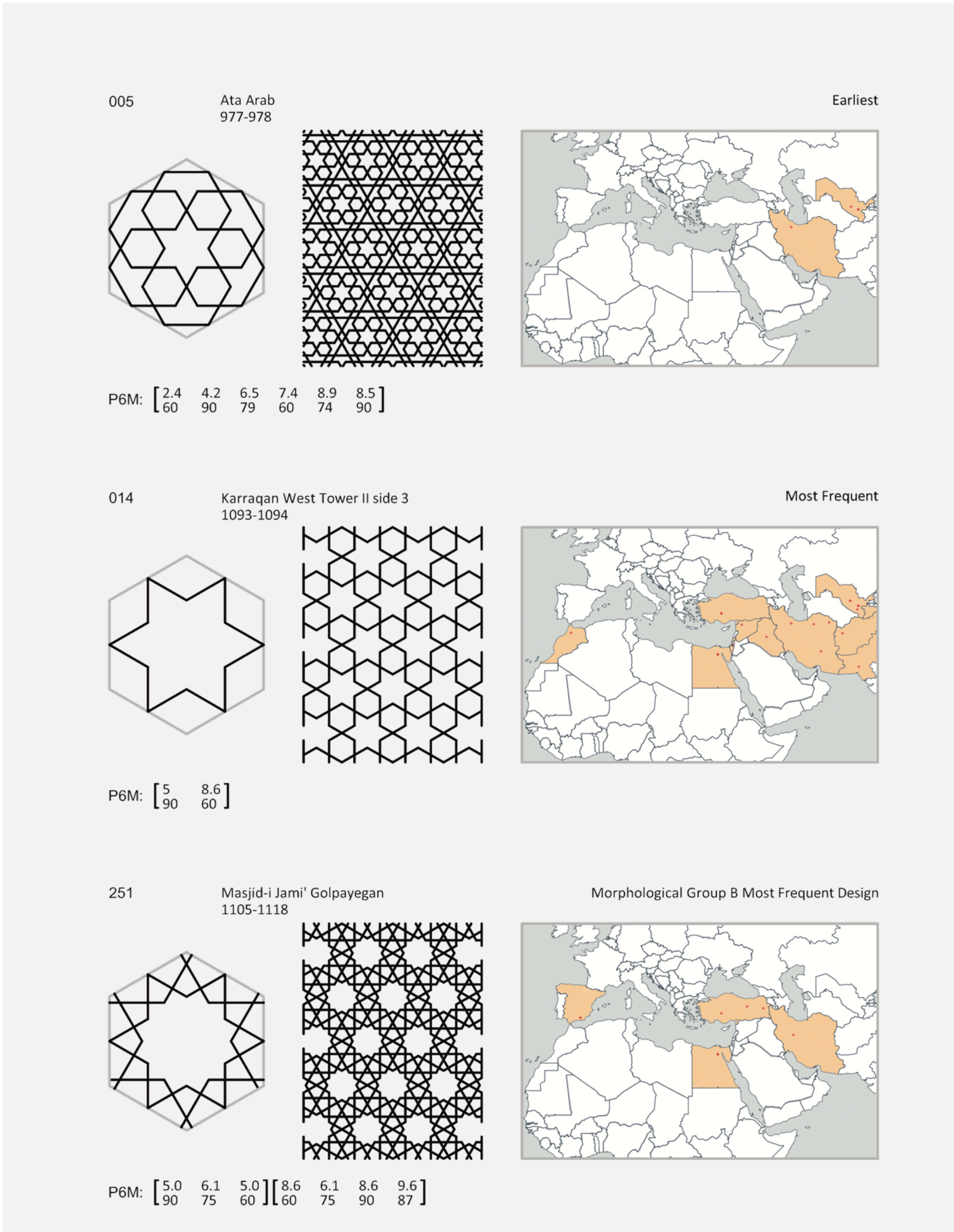


Figure 6.8: The top shows the earliest copied design, the middle the most frequently copied design, and the bottom the most frequently copied design in B1 group.

Level One:

The representational code of the historic designs is examined to compare the value and sequence of the angle parameter while discarding the actualized values of distances (shown in light gray in the below description model):

$$\textbf{Symmetry Group}: \begin{bmatrix} r_{P_1} & r_{P_2} & \cdots & r_{P_i} \\ \emptyset_{P_1} & \emptyset_{P_2} & \cdots & \emptyset_{P_i} \end{bmatrix} \begin{bmatrix} r_{P_{i+1}} & r_{P_{i+2}} & \cdots & r_{P_{i+v}} \\ \emptyset_{P_{i+1}} & \emptyset_{P_{i+2}} & \cdots & \emptyset_{P_{i+v}} \end{bmatrix} \cdots$$

$$\begin{bmatrix} r_{P_{i+v+\cdots+1}} & r_{P_{i+v+\cdots+2}} & \cdots & r_{P_{i+v+\cdots+z}} \\ \emptyset_{P_{i+v+\cdots+1}} & \emptyset_{P_{i+v+\ldots_2}} & \cdots & \emptyset_{P_{i+v+\cdots+z}} \end{bmatrix}$$

In doing so, this level identifies designs that share the same number of segments, the exact flow of polylines, and specific morphological groups, regardless of the actualized measurements of the polyline. Therefore, the recording unit in this level is the entire value and sequence match of angles' parameters. Figure 6.9 shows the implemented codes.

Of the 138 unique designs, 23.19% fall into this level because they share the values and sequence of angles in their representational codes with at least one other design and were identified as structurally equivalent (LV1). The most frequent structure in level one is the following:

$$\textbf{P6M}: \begin{bmatrix} r_{P_1} & r_{P_2} \\ \mathbf{60} & \mathbf{90} \end{bmatrix}$$

The above structure exists in six different arrangements (shown in figure 6.10). The earliest existing design within this structure dates to the 9th century and was discovered by Ernest Hartsfield during the Samara excavations. However, if being extra cautious and considering only designs that are purely geometric (the Samara design contains floral designs), the earliest frequent structure can be dated to the Karraqan East Tower (1067 CE).

```
void LevelZeroSE(Geometry g, Geometry[] G){
  String[] S1 = splitTokens(g.repCODE,"[]"); // Convert First representational code into array
  //Keep only the angle parameter in each array index in the first representational code
  for(int y=0; y<S1.length; y++){
    if(S1[y].equals("/") == false){
      String[] clean = splitTokens(S1[y], "-");
      S1[y] = clean[0];
    }
  }
  String comparison1 = join(S1," ");
  // A loop through the array of Geometry object.
  for(int i=0; i<G.length; i+=1){
    //Fillters to avoid including identical design in this category, design itself, or comparing
    //with different symmetry group
    if(g.ID.equals(G[i].ID) == false && CADno != G[i].CAD && g.SYMMETRY.equals(G[i].SYMMETRY))
    {
      String[] S2 = splitTokens(G[i].repCODE,"[]");
      //Keep only the angle parameter in each array index in the Second representational code
      for(int y=0; y<S2.length; y++){
        if(S2[y].equals("/") == false){
          String[] clean = splitTokens(S2[y], "-");
          S2[y] = clean[0];
        }
      }
      String comparison2 = join(S2," ");
      //Conduct the comparison
      if(comparison1.equals(comparison2)){
        print(G[i].CAD + ",");
      }
    }
  }
}
```

Figure 6.9 The code used for identifying structurally equivalent designs, implemented using processing programing language.

The algorithm used in the search within this level is more conservative in preserving the flow characteristics, as it requires an entire value and sequence match of all angles. However, it is also possible to examine the representational code of the historic designs to compare the value and sequence of only angles that lay at the internal boundaries of the FU –as only these points determine the general flow layout of the designs—while discarding the actualized values of all distances and the values of angles of the constructional points that do not lay at the internal boundaries of the FU (figure 6.11).

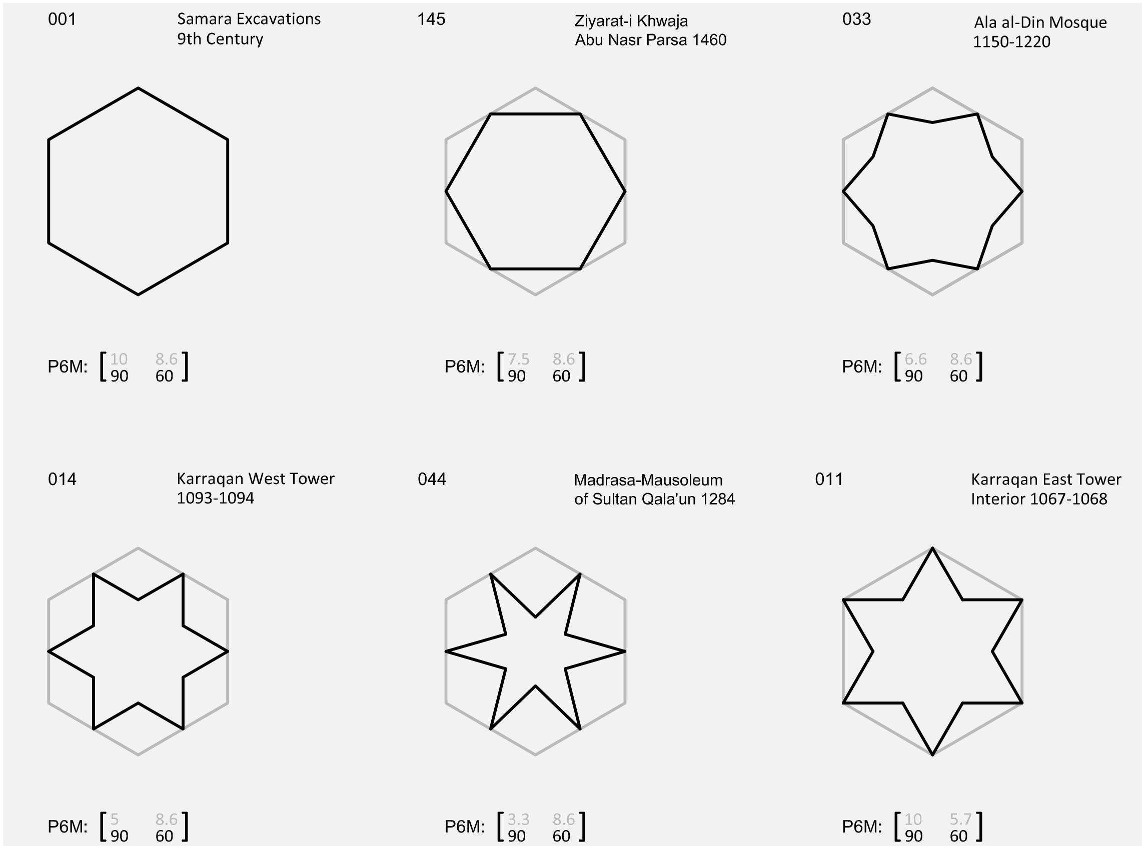


Figure 6.10: The most frequent structures in LV1 structural equivalency.

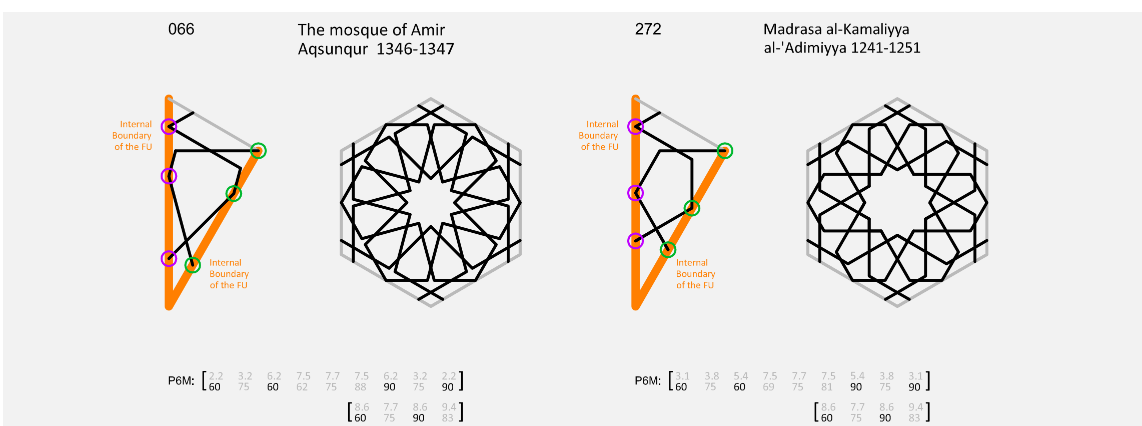


Fig 6.11: The internal boundaries of the FU and the control of the points that lay on these boundaries. The dark black variable in the representational code represents the considered variable in the comparison.

In doing so, this level identifies designs that share a flow in a more flexible manner, regardless of the actualized measurements of the distances of the constructional points. Therefore, the recording unit in this category is the value and sequence match of 30, 60, 90, and 120° angles in the representational codes of the compared designs. It is important to highlight that in this level, the sequence of the discarded angles is still considered. For example, the sequence of 90, 43, 60, and 90° angles matches the sequence of 90, 73, 60, and 90° angles but does not match the sequence of 90, 60, and 90° angles. Figure 6.12 shows the implemented codes.

```
void LevelOneSE(Geometry g, Geometry[] G){
  String[] S1 = splitTokens(g.repCODE,"[]"); // Convert First representational code into array
  //Keep only the angle parameter in each array index in the first representational code
  for(int y=0; y<S1.length; y++){
    if(S1[y].equals("/") == false){
      String[] clean = splitTokens(S1[y], "-");
      S1[y] = clean[0];
      //if the angle is not equal to 30, 60, 90, 120, change the angle to "A"
      if(S1[y].equals("30") == false && S1[y].equals("60") == false &&
      S1[y].equals("90") == false && S1[y].equals("120") == false){
        S1[y] = "A";
      }
    }
  }
  String comparison1 = join(S1," ");
  // A loop through the array of Geometry object.
  for(int i=0; i<G.length; i+=1){
    if(g.ID.equals(G[i].ID) == false && CADno != G[i].CAD && g.Case.equals(G[i].Case) &&
    g.SYMMETRY.equals(G[i].SYMMETRY))
    {
      String[] S2 = splitTokens(G[i].repCODE,"[]");
      //Keep only the angle parameter in each array index in the Second representational code
      for(int y=0; y<S2.length; y++){
        if(S2[y].equals("/") == false){
          String[] clean = splitTokens(S2[y], "-");
          S2[y] = clean[0];
          //if the angle is not equal to 30, 60, 90, 120, change the angle to "A"
          if(S2[y].equals("30") == false && S2[y].equals("60") == false &&
          S2[y].equals("90") == false && S2[y].equals("120") == false){
            S2[y] = "A";
          }
        }
      }
```

Figure 6.12 The code used for identifying structurally equivalent designs of level one for ascertaining the flow in a more flexible manner.

```
      String comparison2 = join(S2," ");
      //Conduct the comparison
      if(comparison1.equals(comparison2) && g.SYMMETRY.equals(G[i].SYMMETRY)){
        print(G[i].CAD + ", ");
      }
    }
  }
}
```

Figure 6.12 Cont.

The results of the new algorithm indicate that 44.20% of the 138 unique designs have at least a structural equivalence with one or more designs (figure 6.13). The most frequent structure is the following:

$$\mathbf{P6M}: \begin{bmatrix} r_{P_1} & r_{P_2} & r_{P_3} \\ \mathbf{60} & \mathbf{90} & \emptyset_{P_3} \end{bmatrix}$$

Level Two:

In this level, the representational code of the historic designs is compared by searching for similar sequences of constructional points while discarding the values of distances and angles (shown in light gray in the description model below):

$$\textbf{Symmetry Group}: \begin{bmatrix} r_{P_1} & r_{P_2} & \cdots & r_{P_i} \\ \emptyset_{P_1} & \emptyset_{P_2} & \cdots & \emptyset_{P_i} \end{bmatrix} \begin{bmatrix} r_{P_{i+1}} & r_{P_{i+2}} & \cdots & r_{P_{i+v}} \\ \emptyset_{P_{i+1}} & \emptyset_{P_{i+2}} & \cdots & \emptyset_{P_{i+v}} \end{bmatrix} \cdots$$

$$\begin{bmatrix} r_{P_{i+v+\cdots+1}} & r_{P_{i+v+\cdots+2}} & \cdots & r_{P_{i+v+\cdots+z}} \\ \emptyset_{P_{i+v+\cdots+1}} & \emptyset_{P_{i+v+\ldots 2}} & \cdots & \emptyset_{P_{i+v+\cdots+z}} \end{bmatrix}$$

In doing so, this level identifies designs that share the same number of segments and specific morphological groups, regardless of the actualized measurements or the flow of the polyline (figure 6.14). Therefore, the recording unit is the match of the sequence of constructional points in polylines between the compared codes. Figure 6.15 shows the implemented codes.

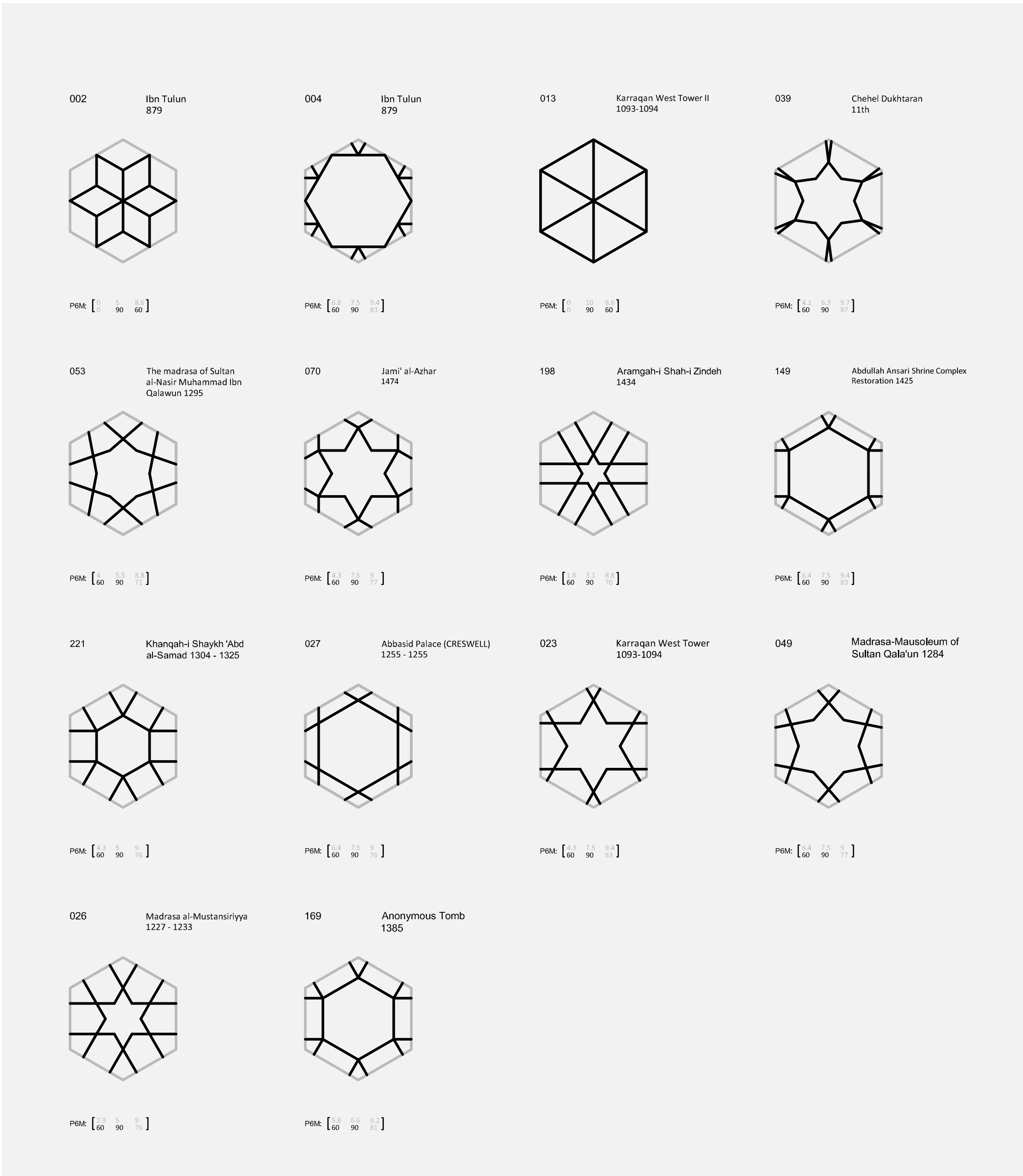


Fig 6.13: The most frequent structure in LV1 structural equivalency when identifying the flow based on points that lay on the internal boundaries of the FU.

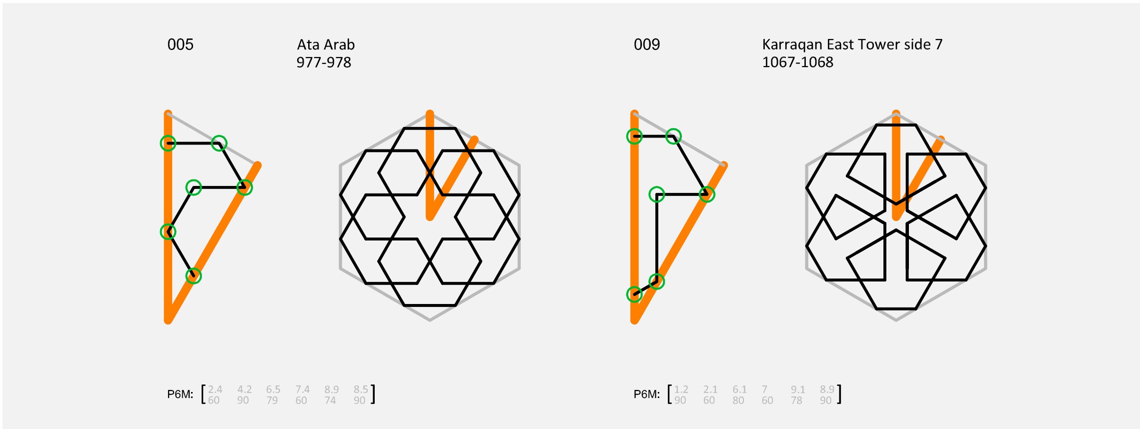


Fig 6.14: Different polyline flows with similar segment counts (five segments each) and similar specific morphological case (A0).

```
void LevelTwoSE(Geometry g, Geometry[] G){
  String[] S1 = splitTokens(g.repCODE,"[]"); // Convert First representational code into array
  // Preserve only point sequence
  for(int y=0; y<S1.length; y++){
    if(S1[y].equals("/") == false){
      String[] clean = splitTokens(S1[y], "-");
      S1[y] = "P";
    }
  }
  String comparison1 = join(S1," ");
  // A loop through the array of Geometry object.
  for(int i=0; i<G.length; i+=1){
    if(g.ID.equals(G[i].ID) == false && CADno != G[i].CAD && g.Seg.equals(G[i].Seg) &&
    g.Case.equals(G[i].Case) && g.SYMMETRY.equals(G[i].SYMMETRY))
    {
      String[] S2 = splitTokens(G[i].repCODE,"[]");// Convert second representational code into array
      // Preserve only point sequence
      for(int y=0; y<S2.length; y++){
        if(S2[y].equals("/") == false){
          String[] clean = splitTokens(S2[y], "-");
          S2[y] = "P";
        }
      }
      String comparison2 = join(S2," ");
      //Conduct the comparison
      if(comparison1.equals(comparison2) && g.SYMMETRY.equals(G[i].SYMMETRY)){
        print(G[i].CAD + ",");
      }
    }
  }
}
```

Figure 6.15 The code used for identifying structurally equivalent designs of level two, implemented using processing programing language.

Of the 138 designs, 76.09% share the same structure with at least one other design. The following structure code represents the most frequent structure (identified design variations are shown in figure 6.16):

$$\mathbf{P6M}: \begin{bmatrix} \mathrm{r}_{\mathbf{P_1}} & \mathrm{r}_{\mathbf{P_2}} & \mathrm{r}_{\mathbf{P_3}} \\ \emptyset_{\mathbf{P_1}} & \emptyset_{\mathbf{P_2}} & \emptyset_{\mathbf{P_3}} \end{bmatrix}$$

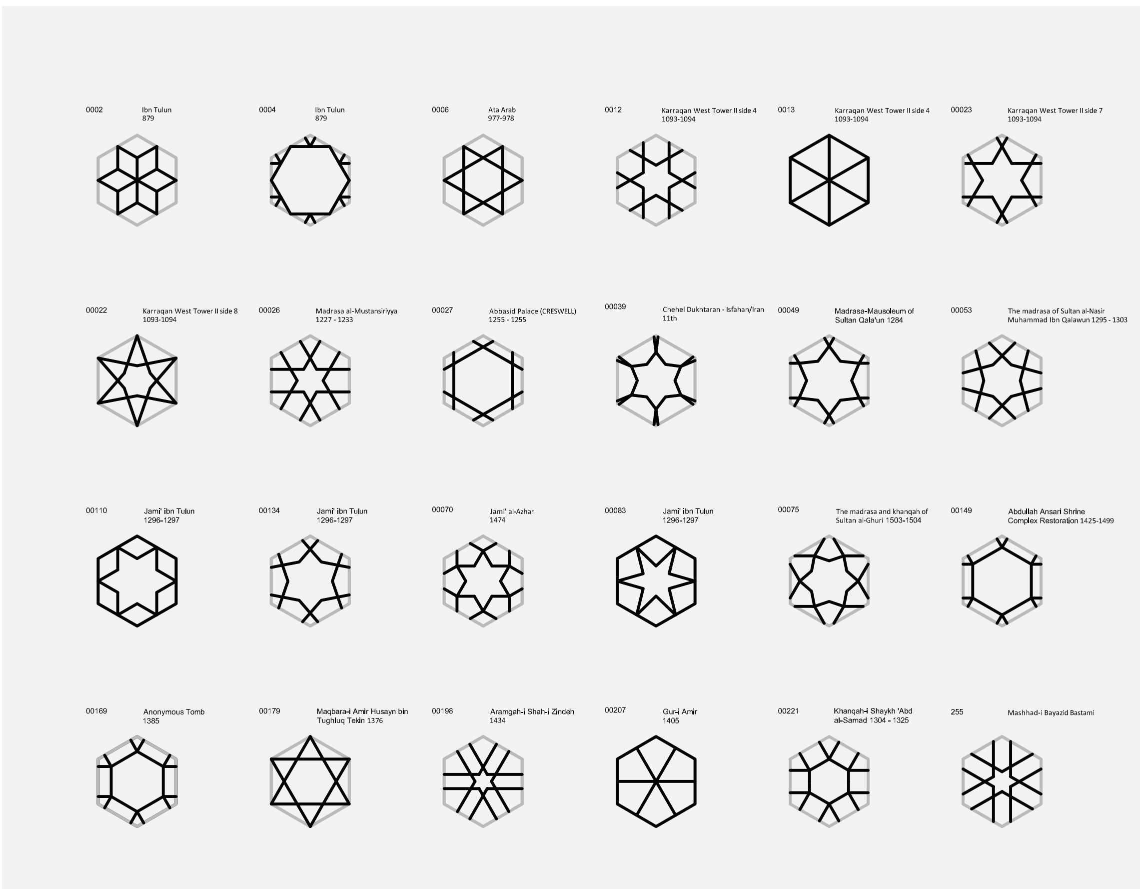


Figure 6.16 The most frequent structures in LV2 structural equivalency

Level Three:

In this level, the representational code of the historic designs is compared by searching for designs that share the same specific morphological groups while discarding the segment counts and the values and sequences of the actualized distances and angles in all of the constructional points (figure 6.17). In doing so, this level identifies designs that share the same number of polylines and T/QP counts regardless of the segment count, flow of polylines, or the actualized measurements of the polylines within the FU. Therefore, the recording unit is the match of the specific morphological groups. Figure 6.18 shows the implemented code.

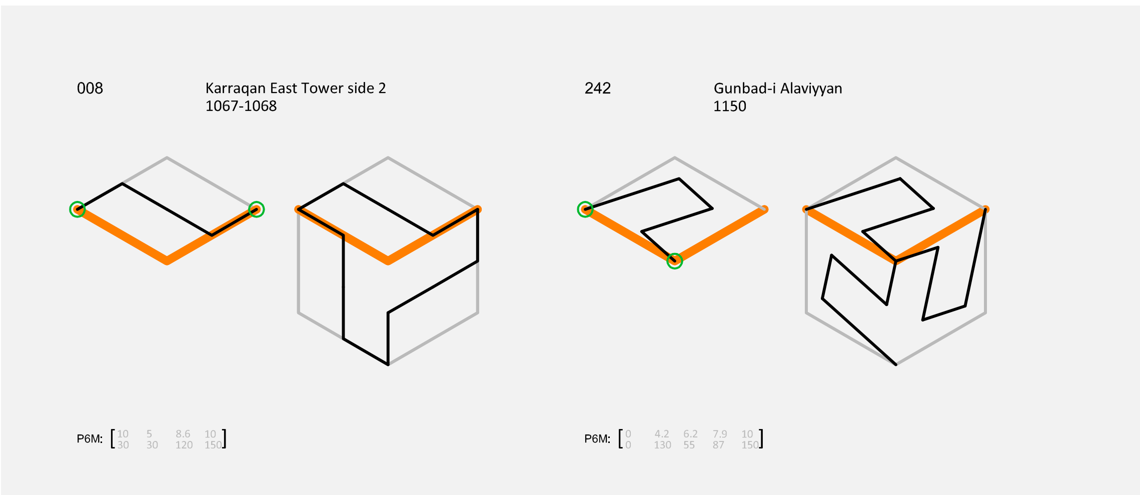


Figure 6.17 Example of LV3 structural equivalency.

```
void LevelThreeSE(Geometry g, Geometry[] G){
  // A loop through the array of Geometry object.
  for(int i=0; i<G.length; i+=1){
    if(g.ID.equals(G[i].ID) == false && CADno != G[i].CAD && g.Case.equals(G[i].Case) &&
    g.SYMMETRY.equals(G[i].SYMMETRY)){
      //If the specific morphological group is the same, print the number of the geometry
      print(G[i].CAD + ",");
    }
  }
}
```

Figure 6.18 The code used for identifying structurally equivalent designs of level three, implemented using processing programing language.

Of the 138 designs, 86.96% share the level of their structures with at least one other design. The most frequent structure is designs composed form a single polyline with no T/QP.

Level Four:

The representational code of the historic designs is compared by searching for similar polyline counts while discarding the existence of T/QP, segment counts, and the values and sequences of the actualized distances and angles of all the constructional points (figure 6.19). Therefore, the recording unit is the match of morphological groups. Figure 6.20 shows the implemented code.

$$\textbf{Symmetry Group}: \begin{bmatrix} r_{P_1} & r_{P_2} & \cdots & r_{P_i} \\ \emptyset_{P_1} & \emptyset_{P_2} & \cdots & \emptyset_{P_i} \end{bmatrix} \begin{bmatrix} r_{P_{i+1}} & r_{P_{i+2}} & \cdots & r_{P_{i+v}} \\ \emptyset_{P_{i+1}} & \emptyset_{P_{i+2}} & \cdots & \emptyset_{P_{i+v}} \end{bmatrix} \ldots$$

$$\begin{bmatrix} r_{P_{i+v+\cdots+1}} & r_{P_{i+v+\cdots+2}} & \cdots & r_{P_{i+v+\cdots+z}} \\ \emptyset_{P_{i+v+\cdots+1}} & \emptyset_{P_{i+v+\ldots 2}} & \cdots & \emptyset_{P_{i+v+\cdots+z}} \end{bmatrix}$$

Of the 138 unique designs, 94.20% share level four structures with at least one other design. The most frequent structure is the single sequence of points (single polyline). Figure 6.20 shows example from the structural equivalency of this level.

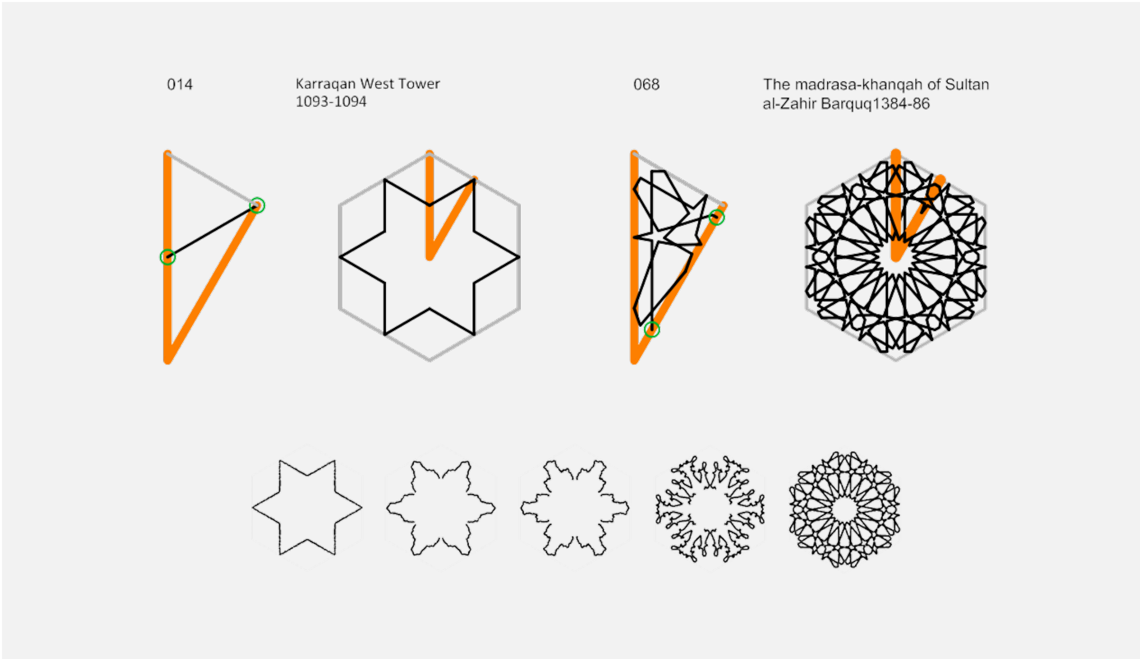


Figure 6.19: Example of LV4 structural equivalency. Simple design vs. complex designs in terms of segment count and specific morphological case. All designs are considered equal in terms of LV4 as they are composed from a single polyline.

```
void LevelFourSE(Geometry g, Geometry[] G){
  // A loop through the array of Geometry object.
  for(int i=0; i<G.length; i+=1){
    if(g.ID.equals(G[i].ID) == false && CADno != G[i].CAD && g.MC.equals(G[i].MC) &&
    g.SYMMETRY.equals(G[i].SYMMETRY)){
      //If the specific morphological group is the same, print the number of the geometry
      print(G[i].CAD + ",");
    }
  }
}
```

Fig 6.20: The code used for identifying structurally equivalent designs of level four, implemented using processing programing language

6.7 THE MORPHOLOGICAL CORRELATIONS

The results from the previous section show that similarities between the hexagon-based designs become more frequent as the virtual dimension is approached. Figure 6.21 shows the identical category and the four levels of structural equivalency categories arranged in an ordinal fashion starting from the actual dimension, which is represented by the frequency of identical designs, followed by the closest level of structural equivalency category to the actual dimension, moving toward the virtual dimension (right of the figure).

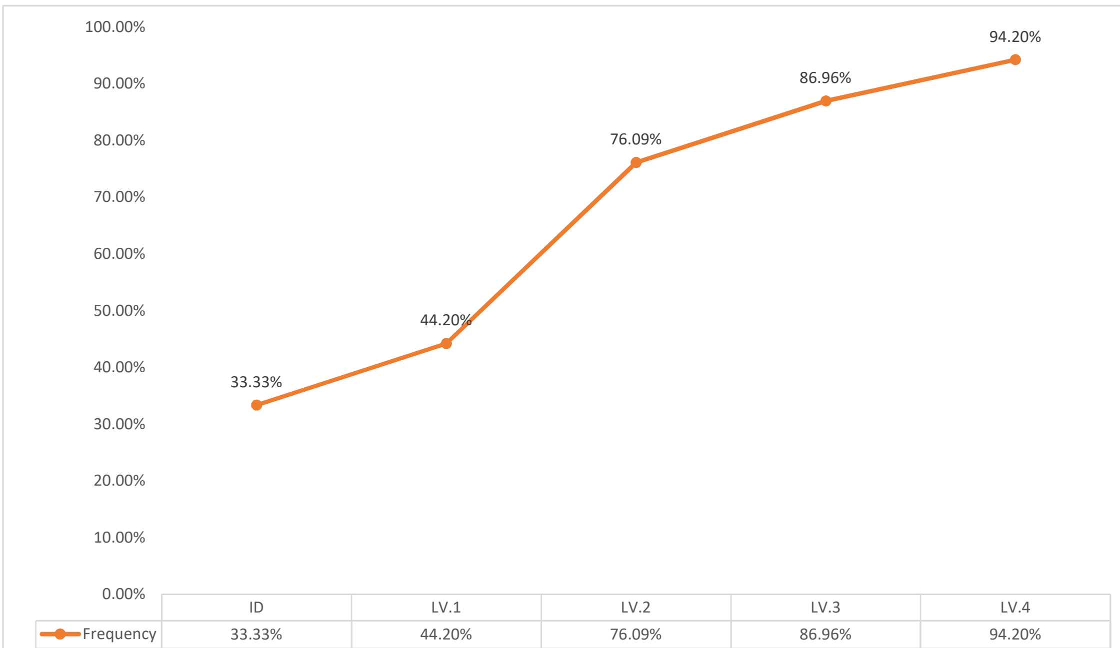


Figure 6.21: Line graph shows the frequency of identical and the four levels of structural equivalency categories.

Figure 6.19 in the previous section shows how a simple design (in terms of segment count and type of connection points) can be structurally equivalent to another more complex design. To identify such morphological correlations in a holistic manner, connections between the content categories and design segment must be established. Figure 6.22 shows the flowchart for each of the five symmetry groups, with the morphological groups on the x-axis broken down by the specific morphological groups.

The y-axis shows the segment count. Each circle in the figure represents a single or group of actualized designs (further explained in the subsequent enlarged views).

This figure shows the existence of a minimum of single polyline (Morphological Group A) in symmetry groups P6M, P6, P3, and P31M and the existence of a maximum of six polylines (Morphological Group F) in symmetry group of P6 only. In addition, the figure shows the existence of a single segment design as well as a design with a maximum of 37 segments (SMG A8).

Figure 6.23 shows an enlarged view of SMG B2 from the P6M symmetry group. This figure shows how the actualized designs relate to each other on multiple structural levels. For instance, if we look at designs that contains 11 segments, we can see four designs each two designs are structurally equivalent at level 1, as indicated by the underlined labeled LV1, since the designs share the same flow of polylines, the same number of segments, and the same specific morphological groups. However, the four designs within 11 segments are structurally equivalent at level 2 as these designs share the same number of segments and the same specific morphological groups. These designs (i.e., all those containing 11 segments) share the same specific morphological group with the entire branch shown in the figure. All the branches include designs with two polylines (indicated in the figure by different colors for each polyline) and two T/QP. However, when comparing this branch to another branch within the B morphological group, the two branches have designs with two polylines but differ in the number of T/QP. Figure 6.23 explains this using morphological group C as an example.

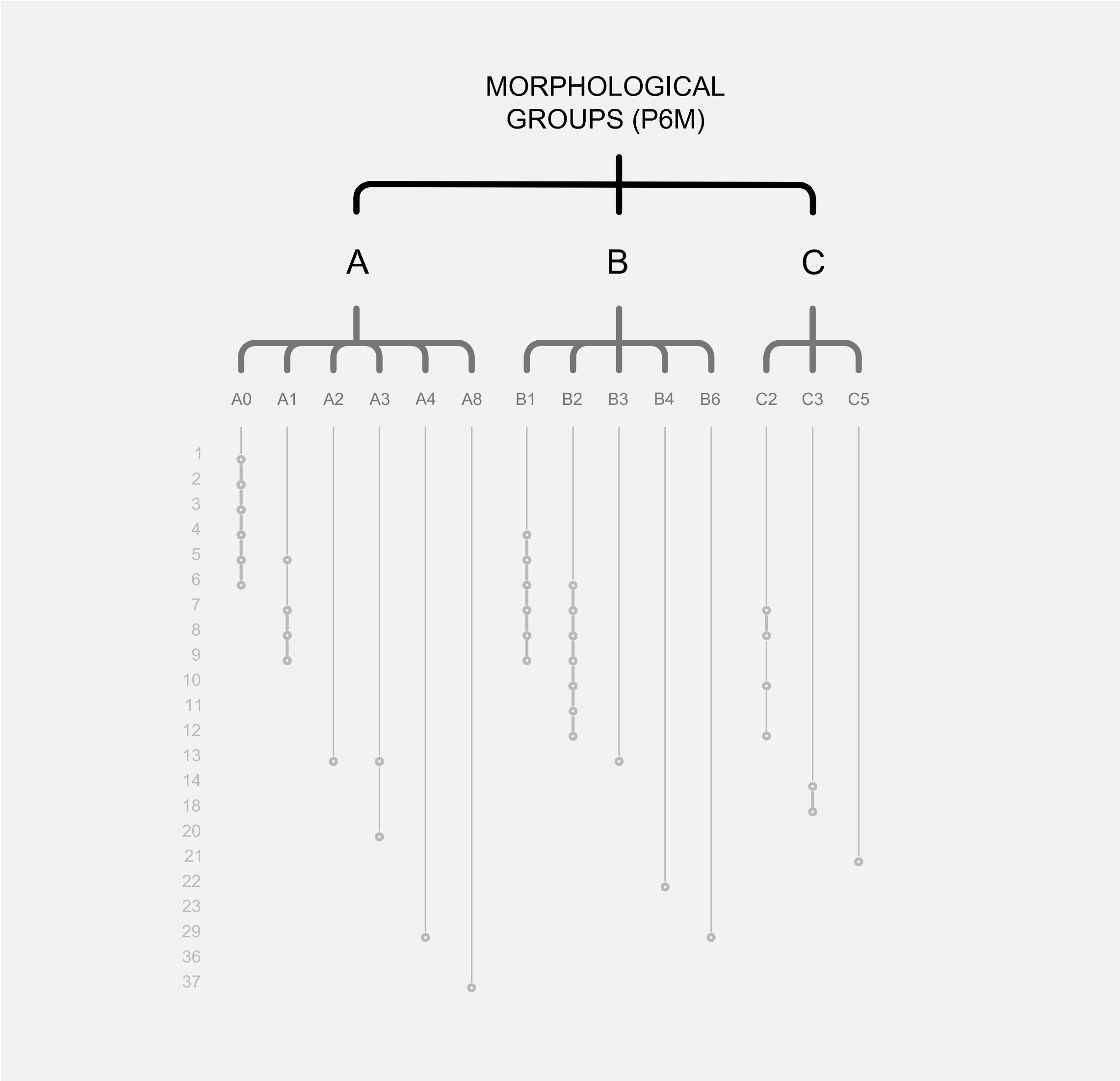


Figure 6.22: Morphological groups that exist within each symmetry type. Each morphological group is broken down by its specific morphological groups on the x-axis, and the segment count on the y-axis.

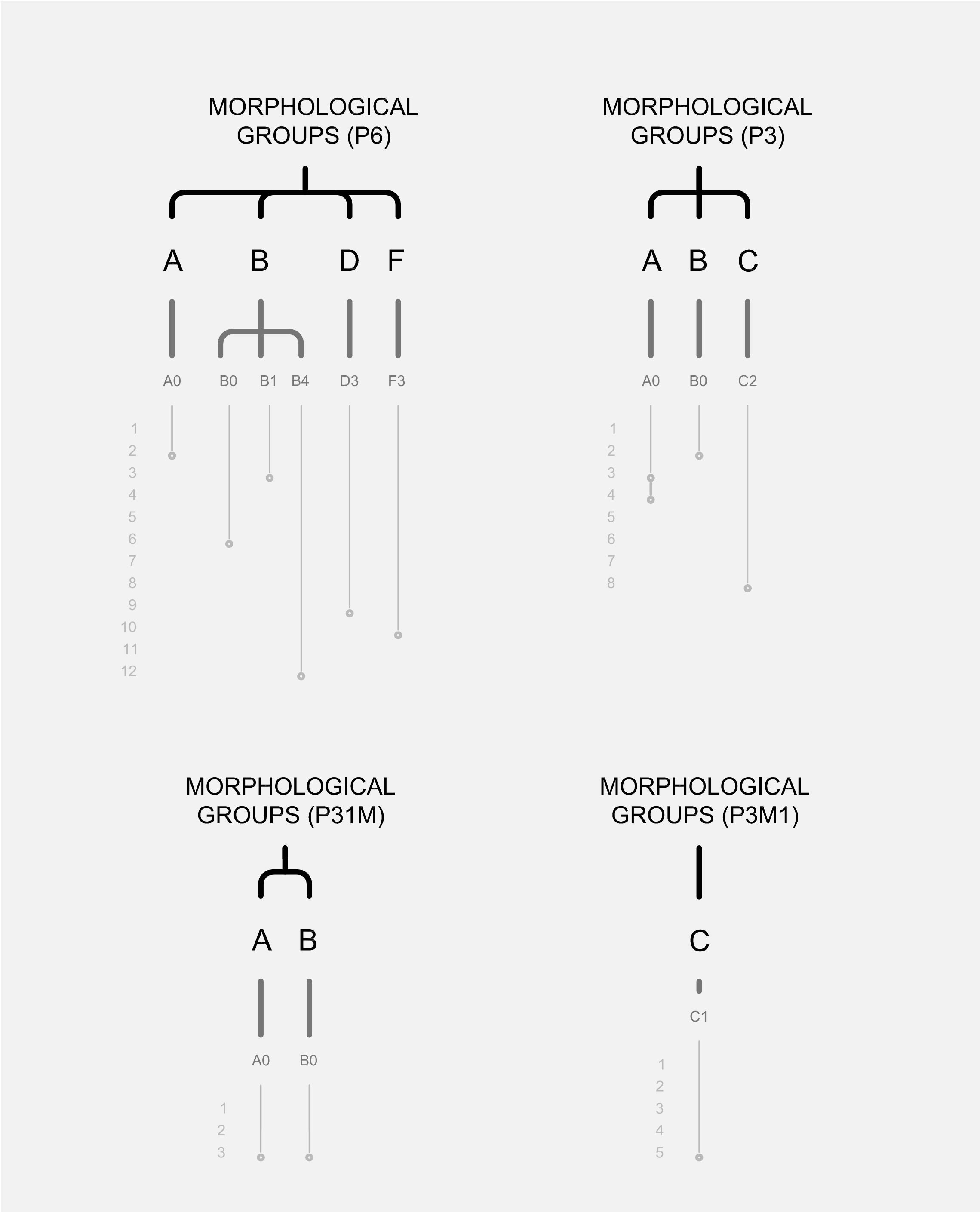


Figure 6.22 cont.

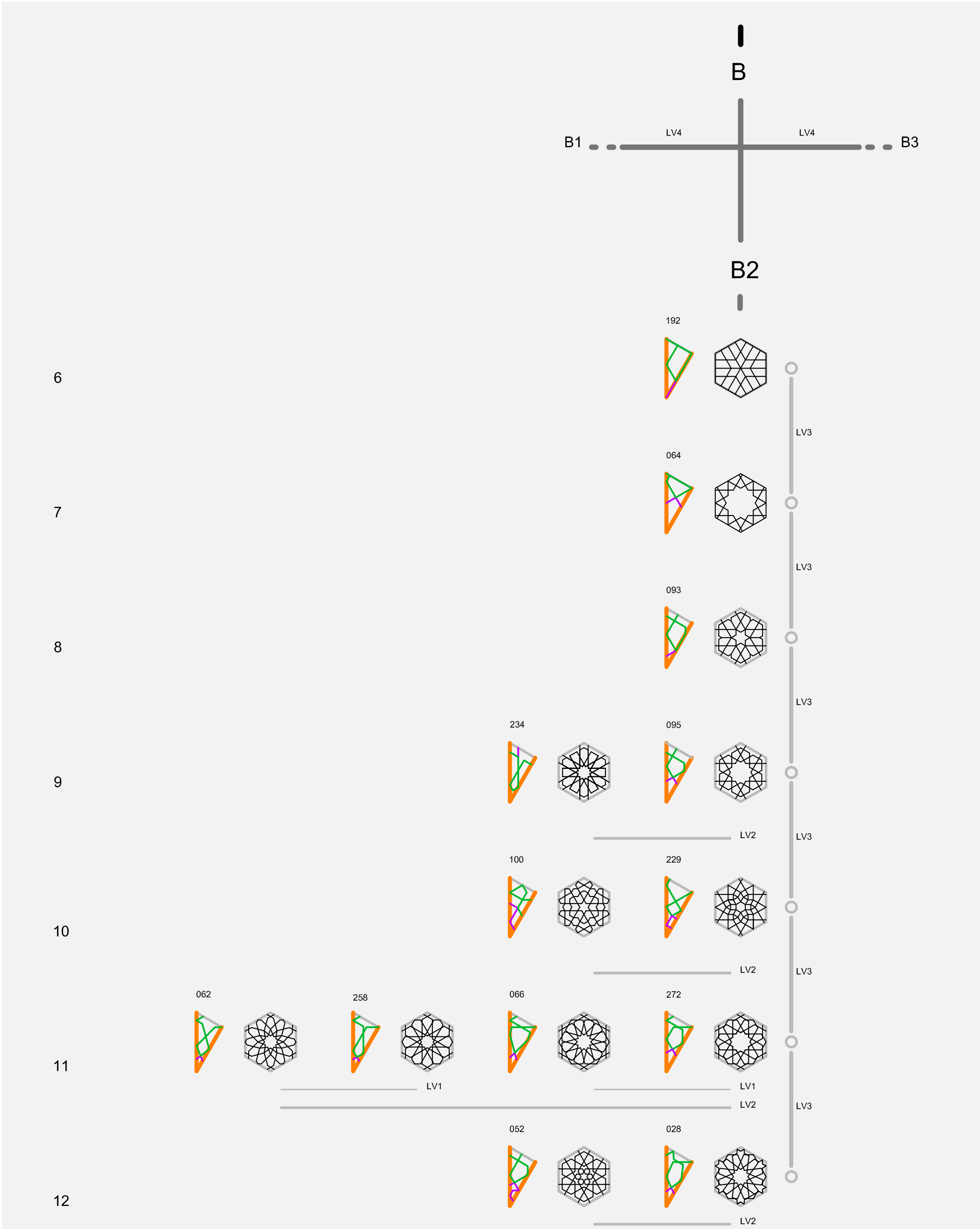


Figure 6.23 Enlarged view of the specific morphological case B2 within the P6M symmetry group.

Figure 2.24 shows how designs that fall within the same morphological group can be related to each other. For instance, the figure explains how designs that fall within SMG C5 correlate to designs that fall within SMG C3; that is, both designs contain three polylines (structurally equivalent at level four).

This grouping system is parametrically expandable and capable to accommodate designs that go beyond the identified historic ones. For instance, if a design contains more than six polylines, or more that 37 segments, such a design will still fit within the same flowchart, and morphological correlations with historical designs can be established. Appendix C shows the flowcharts for all five types of the investigated symmetry groups: P3, P3M1, P31M, P6, and P6M.

The results of the search algorithm utilized when developing the flowcharts. Therefore, The flowcharts help not only to visually understand structural similarities but also assists in validating the results of the search algorithms.

6.8 CHRONOLOGY OF THE MORPHOLOGICAL GROUPS

The morphological groups were revisited to investigate the appearance of categories chronologically. This section examines only the P6M symmetry group as it comprises 93.41% of the data and fewer cases are available for the other four symmetry groups: P6, P3, P3M1, and P31M.

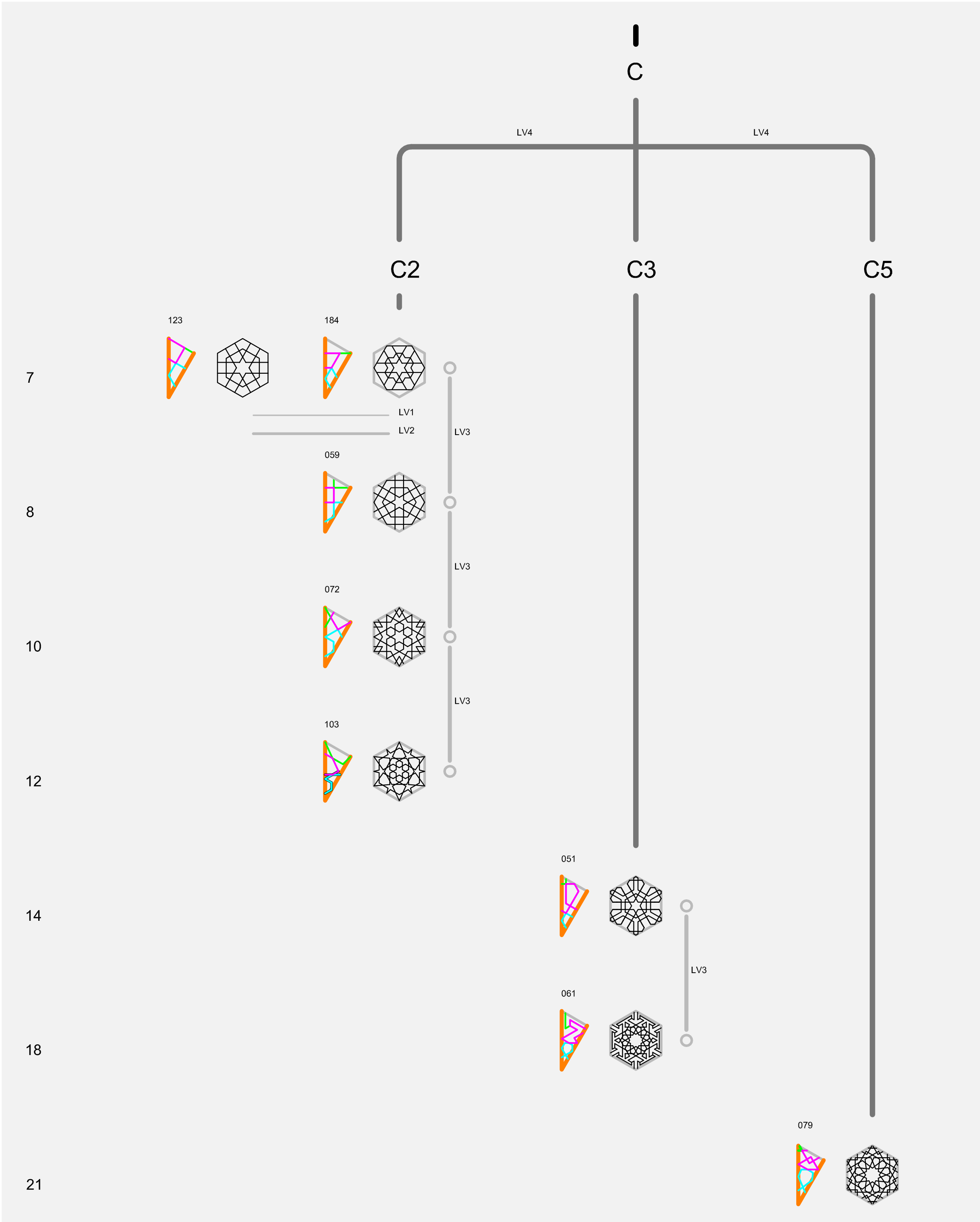


Figure 6.24: Enlarged view from the morphological group C within the P6M symmetry group. Three polylines are indicated in different colors.

The data shows that A0 SMG designs have been the most common since the early period of hexagonal-based IGP and afterwards. The earliest existing design with a single polyline and no T/QP dates to the 9th century in Samara. However, if being extra cautious and considering only designs that are purely geometric (the Samara design contains floral motifs), the earliest A0 SMG design dates to 977 CE at Ata Arab. The highest number of A0 segments was found in 1133 CE with the existence of a design with six segments (figure 6.19).

The rest of the A morphological group (i.e., A1, A2, A3, A4, and A8) occurred later between the 13th and early 15th centuries. The highest number of segments was reached with 37 segments of A8 SMG in 1274 CE (figure 6.25).

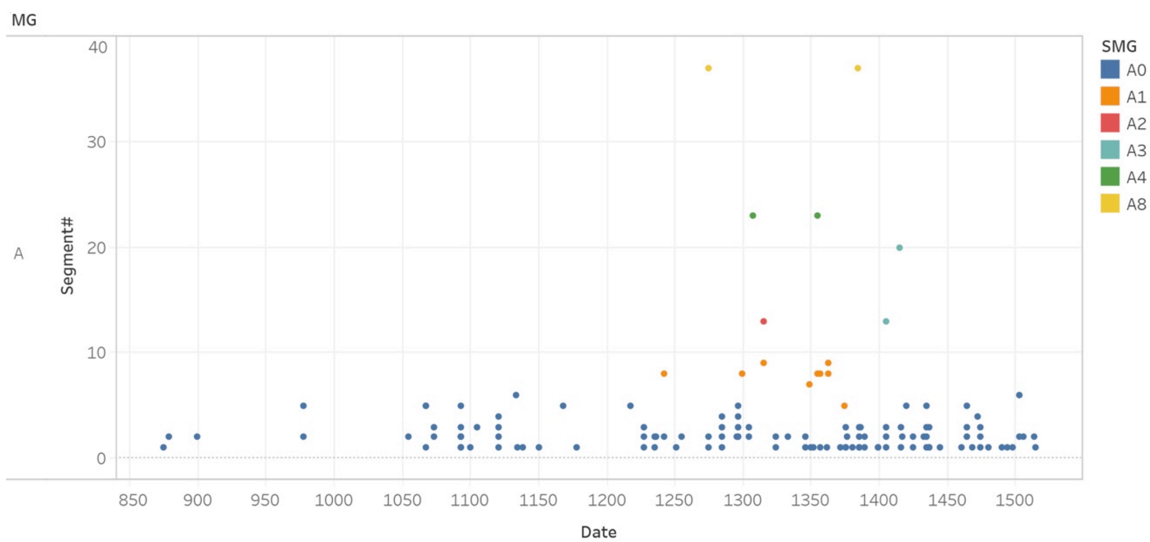


Figure 6.25: Chronological segment count within A MG for symmetry Group P6M. Color indicates the SMG.

Designs of the B MG (two polylines) existed as early as the 9th century with a floral design in the Ibn Tulum mosque dated to this time by K. A. C. Creswell (1919 p.187). This design is a B1. However, B MG designs disappeared after this time and returned in the mid-11th century at the Karraqan East Tower in 1067 CE, reaching the highest number of segments in the late 13th century with nine segments. B2, B3, B4, and B6 existed later between the 13th and early 15th centuries, with the highest number of segments reached in 1323 CE with 29 segments (figure 6.26).

Designs within the C MG started as early as the mid-12th century with C2 designs with 12 segments that later dropped to 10 segments in other designs. C3 designs in the late 13th century had 14 segments, reaching 18 segments in the early 14th century. Moreover, a single case of a C5 design with 21 segments was identified (figure 6.27).

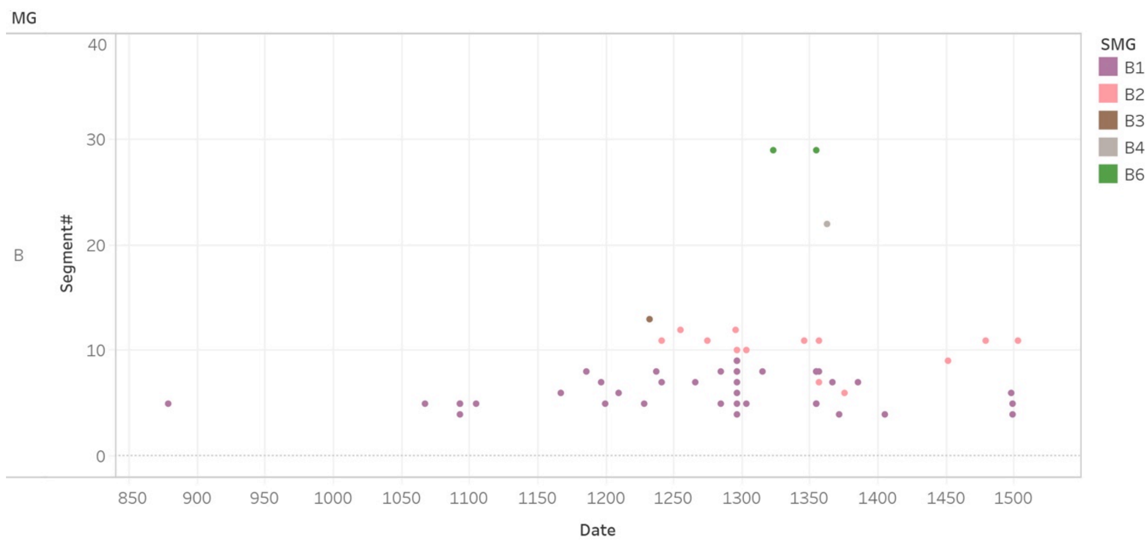


Figure 6.26: Chronological segment count within B MG for symmetry Group P6M. Color indicates the SMG.

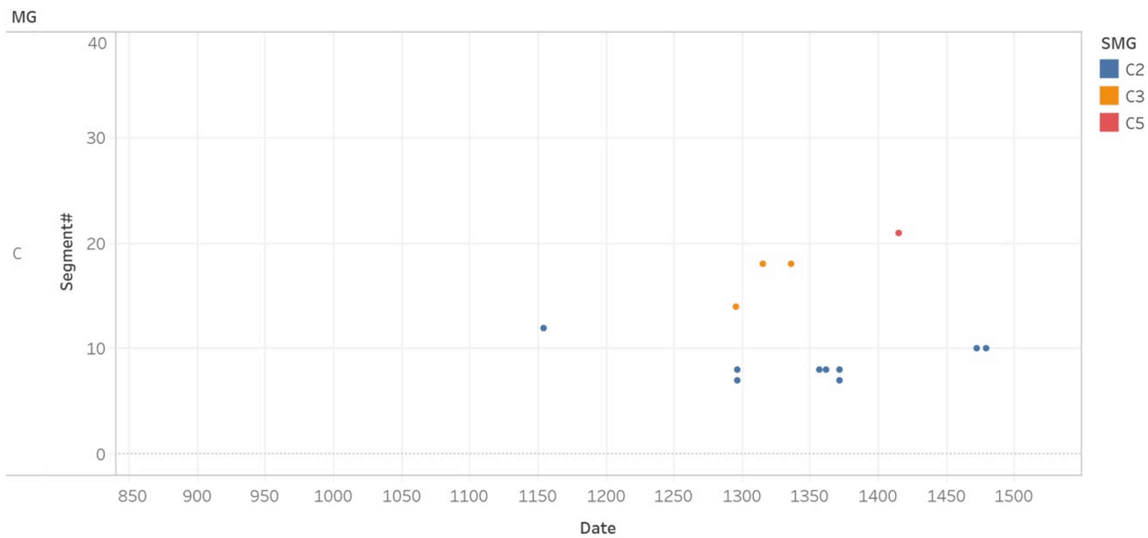


Figure 6.27: Chronological segment count within C MG for symmetry Group P6M. Color indicates the SMG.

In her discussion of the chronological development of Islamic Geometric patterns, Gülru Necipoglu argued that the peak development period falls between the 11th and mid-13th centuries. The findings of this research shows that in the case of the hexagon-based IGP this development is indicated by the introduction of segment intersections and the emergence of cases of designs with multiple polylines. The following SMG were identified before the mid-13th century: A0, A1, B1, B2, B3, and C2. However, the maximum segment count is 13. After the mid-13th century an A8 SMG design with 37 segments existed in Konya, Turkey in 1274 CE.

Necipoglu also argued that the “last creative impulse” for IGP took place between the 14th century and early 16th century. The findings of this research shows that in the case of hexagon-based IGP there are some sophisticated single polyline cases such as A2, A3, and A4 and multiple polyline cases such as the B4, B6, C3, and C5 that began to emerge and be replicated.

CHAPTER SEVEN

TOWARD MORPHOLOGICAL UNDERSTANDING OF HEXAGONAL-BASED ISLAMIC GEOMETRIC PATTERNS

7.1 CONCLUSION

This research addressed the question of how to incorporate mathematics and morphology to describe IGP. It then utilized this description to address the question of what are the morphological corrections between historic design singularities.

Through investigating the historical evidence, the study identified that a hexagonal IGP is the product of infinite replication of a polyline(s) using one of the five hexagon-based wallpaper symmetry groups: P3, P3M1, P31M, P6, and P6M. The fundamental unit of a hexagon-based IGP contains at least a single polyline with at least a single segment and it can be expanded to include multiple polylines with multiple segments that can interact with each other. When put into mathematical terms, this definition captures the reality of historical IGP designs in a parametric, numerical form. Consequently a parametric description model was developed.

The parametric description model was utilized to derive representational codes that store actualized value and structural relations of the historically existing designs. These codes facilitated communication between the historical designs and innovative computational tools and enabled the investegation of similarities between the historical designs. In this sense, this dissertation shares a goal with Abu'l Wafa al-Buzjani, who, in his book *On Geometric Constructions,* aimed to facilitate communication between geometric designs and the scientific language of his age—mathematics.

When the representational codes of historical designs were compared to each other in this research, it was found that hexagon-based IGP correlate to each other in both the actual and virtual dimensions.

The representational codes enabled to identify identical designs that exist in different regions and chronological periods and show how a particular design where replicated. It has been found that 66.3% of the collected 273 designs share their representational codes with at least one other design. This shows that design replication was often practiced and many designs were reproduced later using same, or different embellishment techniques. Furthermore, this study shows that replication is not limited to simple designs in terms of segment count or the design SMG; yet, complex designs also replicated. For instance, consider the design shown in figure 7.1 which show the design exist Madrasa al-'Attarin in Fez, Moroco and its replication in Alhambra palace in Granada, Spain. This design is with up to 29 segments and of B6 SMG. This also supports the transmission of historic designs between regions using some sort of medium such as manuscripts or design scrolls.

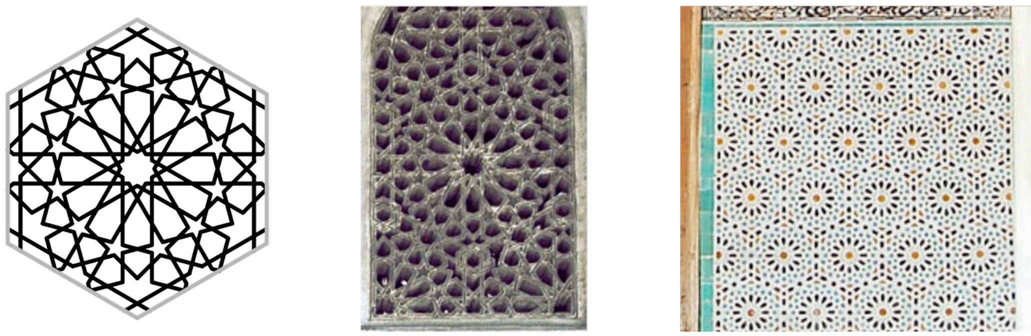


Figure 7.1 Identical designs. From left to right: Madrasa al-'Attarin, Alhambra palace, the ground geometry (Wade 2015).

Beyond the actual dimension, IGP also correlate to each other in the virtual dimension. The representational codes when investigated helped in detecteding links between designs. These structural links are foundational to existing designs and helped to create the enormous diversity of design of hexagonal IGP.

This research determined that a total of 44.2% of designs share with at least one other design the same flow of polylines, number of segments, and specific morphological group. A total of 76.09% of designs share with at least one other design the same number of segments and specific morphological group. A total of 86.96% share with at least one other design the same specific morphological group, and 94.20% share with at least one other design the same morphological group.

The morphological groups are used as a categorization system for patterns that incorporate designs that share basic “morphogenetic” characteristics. Five morphological groups were established: morphological group A, morphological group B, morphological group C, morphological group D, and morphological group F. This system, because it considers both the actual and the virtual dimensions, represent not only what exists but what could exist.

Moreover, this system of categorization does not contradict with previous systems developed by other scientific studies of IGP such as Abas and Salman's symmetry classification, nor is it intended to replace those systems. In contrast, this system further considers the details of each symmetry group to further relate or differentiate the designs within each symmetry group based on the internal relationships of the design components.

Finally, the research investigated the historical development of hexagon-based IGP using morphological categorization. It was found that all three A, B, and C MG were reached prior to the 13th century, with continued use afterwards. However, after the 14th century, the historical designs evolved in regard to segment count and by creating more internal intersections between the polylines.

7.2 LIMITATIONS

Although the results are generalizable for hexagon-based IGP and not for other periodic Islamic geometries, insights can still be gained to create similar procedures for other types of repeat units. The results presented in this research are based on the examination of periodic hexagon-based Islamic geometric designs and thus the results represent those designs.

7.3 FUTURE RESEARCH

Future research will include investigation of other types of periodic structures such as square-based Islamic patterns, with the goal of constructing a database that includes all periodic geometric patterns to advance the research and design practice of periodic IGP. According to Sayed Abas and Amer Salman(1995) symmetry study the hexagonal and square repeat unit constitutes the dominant majority of periodic IGP. Future research will aim to employ the methodology of this research to other periodic patterns to create a unified understanding across different repeat units. For instance, figure 7.2 shows a hexagonal RU next to a square RU. Using the new morphological categorization, both patterns can be identified as B1 as each FU include two polylines and a single QP within the FU.

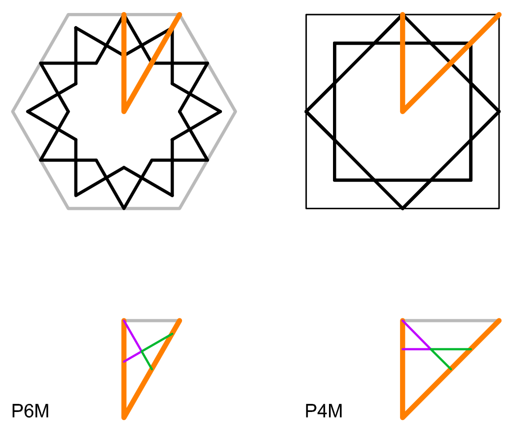


Figure 7.2 employing the morphological categorization across RU.

Furthermore, the developed parametric description establishes a lower level interaction with the methodology that grants designers complete control of the geometric components and their internal structure. Such control of shape is considered the “primary ingredient” for producing architecture that alters shape(Kolarevic, Parlac 2015). To this end, the researcher has taken steps in that direction to build the physical metamorphosis of geometric patterns. Preliminary results of the investigation was presented in poster format at the Conference Proceedings of the Architectural Research Centers Consortium 2017 (figure 7.1).

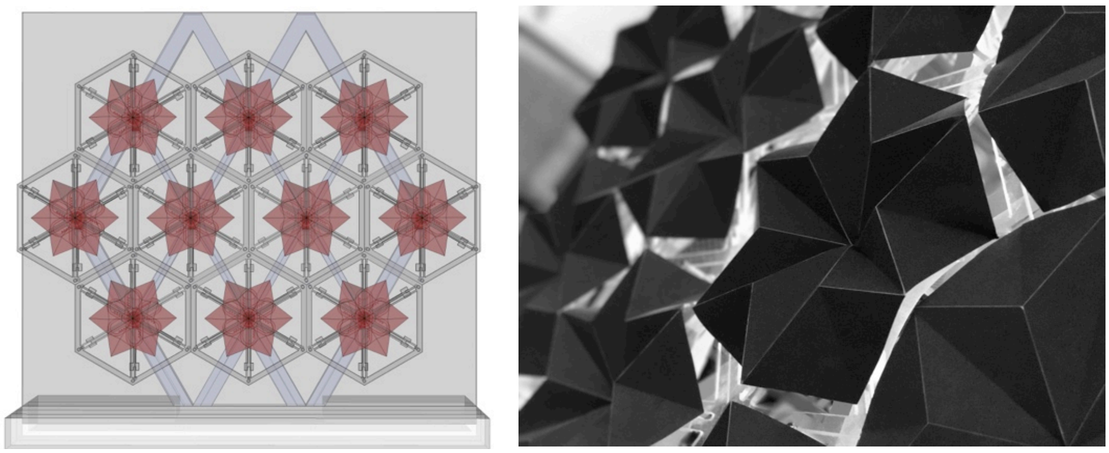


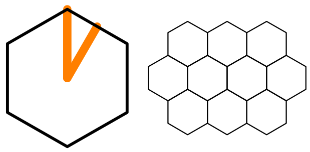
Fig.7.1 Bottom left: digital model with ten hexagonal repeat units. Bottom right: finished prototype

APPENDICES

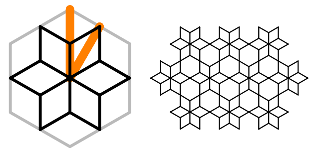
Appendix A

Collected Hexagonal-Based Islamic Geometric Patterns that explains the pattern, single geometry, and the fundamental unit.

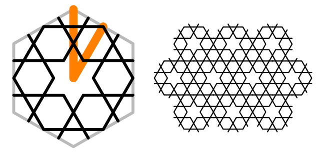
001 Samara Excavations
9th Century



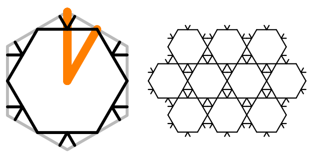
002 Ibn Tulun
879



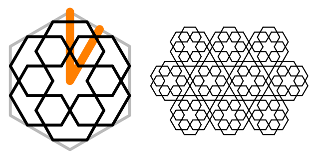
003 Ibn Tulun
879



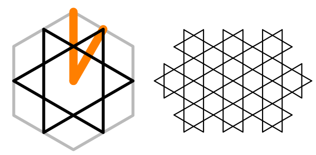
004 Ibn Tulun
879



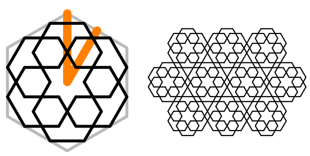
005 Ata Arab
977-978



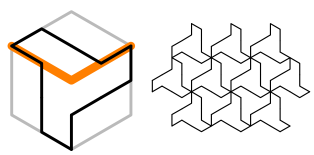
006 Ata Arab
977-978



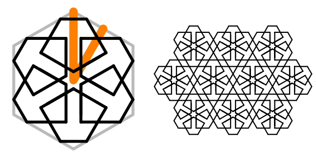
007 Karraqan East Tower side 6
1067-1068



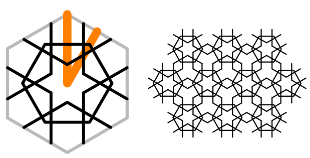
008 Karraqan East Tower side 2
1067-1068



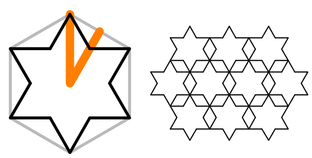
009 Karraqan East Tower side 7
1067-1068



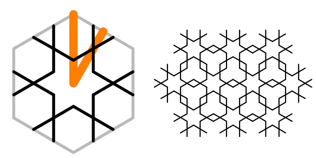
010 Karraqan East Tower side 8
1067-1068



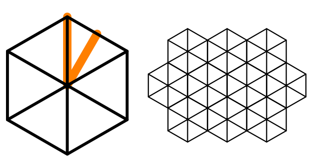
011 Karraqan East Tower Interior
1067-1068



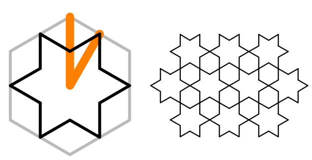
012 Karraqan West Tower II side 4
1093-1094



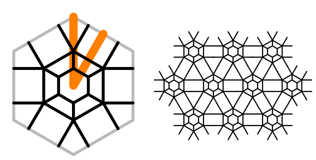
013 Karraqan West Tower II side 4
1093-1094



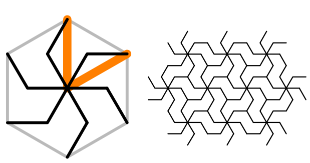
014 Karraqan West Tower II side 3
1093-1094



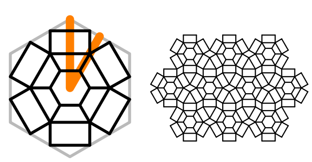
015 Karraqan West Tower II side 3
1093-1094



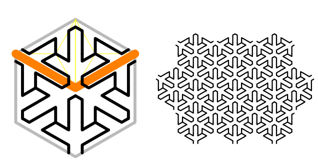
016 Karraqan West Tower II side 3
1093-1094



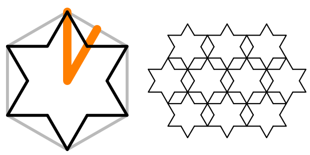
017 Karraqan West Tower II side 6
1093-1094



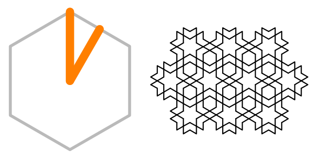
018 Karraqan West Tower II side 6
1093-1094



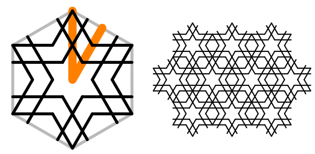
019 Karraqan West Tower II side 6
1093-1094



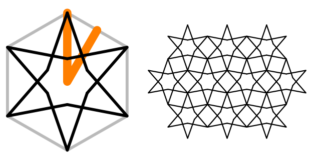
020 Karraqan West Tower II side 5
1093-1094



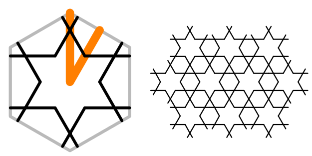
021 Karraqan West Tower II side 8
1093-1094



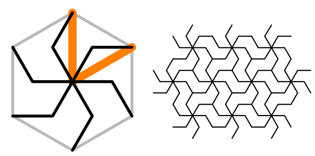
022 Karraqan West Tower II side 8
1093-1094



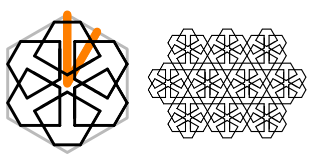
023 Karraqan West Tower II side 7
1093-1094



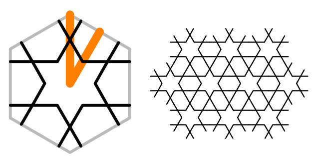
024 Karraqan West Tower II Mihrab
1093-1094



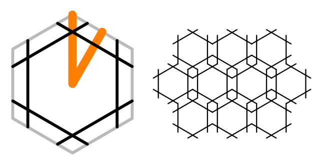
025 Karraqan West Tower II Mihrab
1093-1094



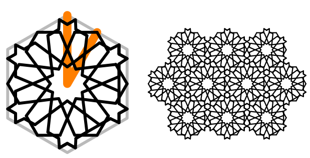
026 Madrasa al-Mustansiriyya (CRESWELL)
1227 - 1233



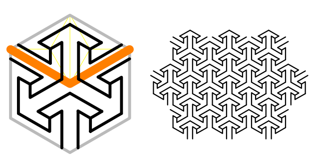
027 Abbasid Palace (CRESWELL)
1255 - 1255



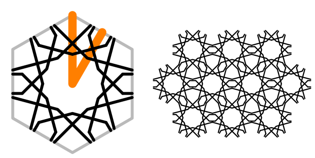
028 Abbasid Palace (CRESWELL)
1255 - 1255



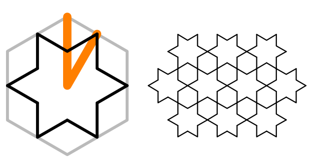
029 Gunbad-i Qabud
1196-1197



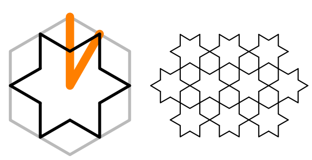
030 Gunbad-i Qabud
1196-1197



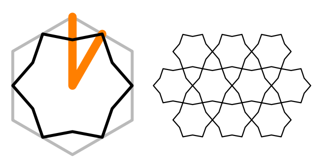
031 Maqbara-i Khwaja Atabak
1100-1150



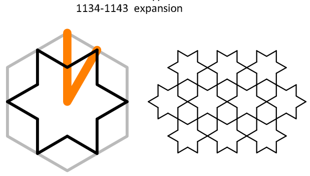
032 Magak-i 'Attari Masjid
1178-1179



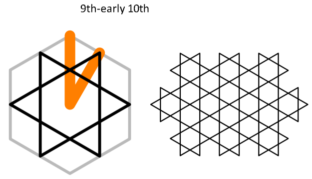
033 Ala al-Din Mosque
1150-1220



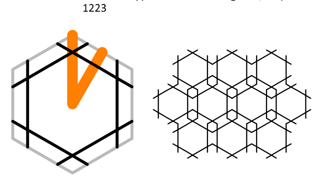
034 Jami' al-Qarawiyyin
1134-1143 expansion



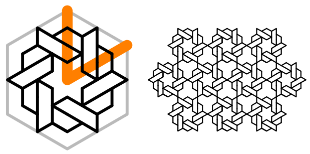
035 Shibam-Kawkaban - Yemen
9th-early 10th



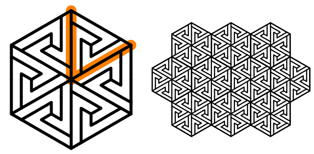
036 Mustansiriyyah Madrasah - Baghdad/Iraq
1223



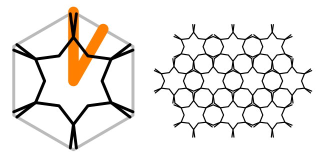
037 Mustansiriyyah Madrasah - Baghdad/Iraq
1223



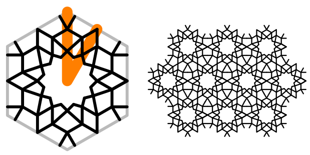
038 Chehel Dukhtaran - Isfahan/Iran
11th



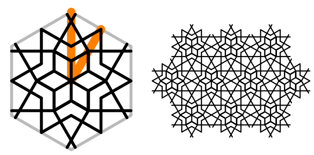
039 Chehel Dukhtaran - Isfahan/Iran
11th



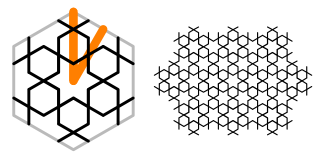
040 The mosque and the khanqah of Amir Shaykhu
1349-1355



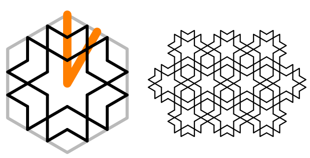
041 Jami' al-Sultan al-Zahir Baybars
1266-1268



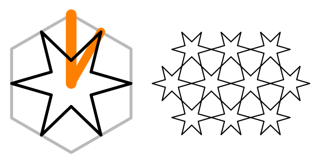
042 Madrasa-Mausoleum of Sultan Qala'un
1284



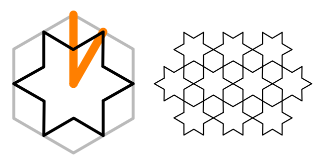
043 Madrasa-Mausoleum of Sultan Qala'un
1284



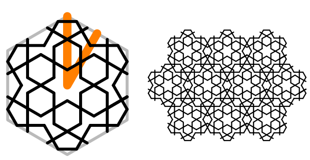
044 Madrasa-Mausoleum of Sultan Qala'un
1284



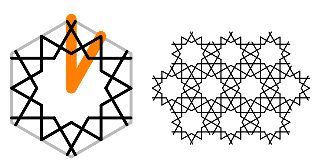
045 Madrasa-Mausoleum of Sultan Qala'un
1284



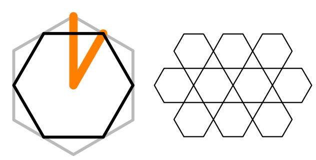
046 Madrasa-Mausoleum of Sultan Qala'un
1284



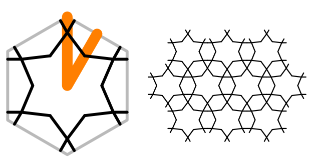
047 Madrasa-Mausoleum of Sultan Qala'un
1284



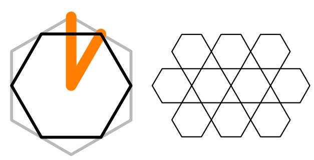
048 Madrasa-Mausoleum of Sultan Qala'un
1284



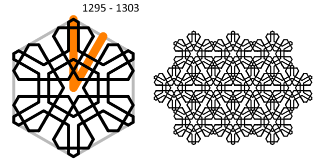
049 Madrasa-Mausoleum of Sultan Qala'un
1284



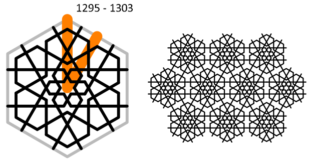
050 Madrasa-Mausoleum of Sultan Qala'un
1284



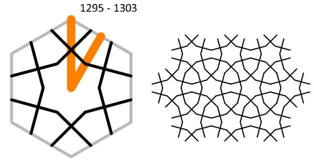
051 The madrasa of Sultan al-Nasir Muhammad
Ibn Qalawun
1295 - 1303



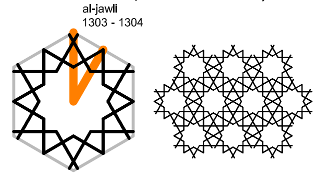
052 The madrasa of Sultan al-Nasir Muhammad
Ibn Qalawun
1295 - 1303



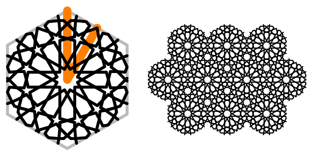
053 The madrasa of Sultan al-Nasir Muhammad
Ibn Qalawun
1295 - 1303



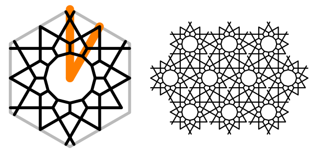
054 The khanqah-madrasa of Amir Sanjar
al-jawli
1303 - 1304



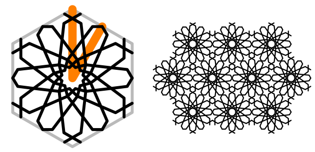
055 Funerary Complex of Baybars al-Jashankir
1307-1310



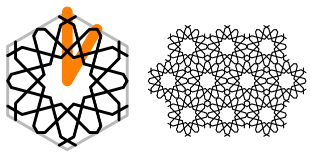
056 The mosque of Sultan Hasan
1356



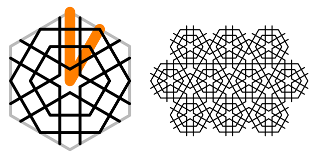
057 Qubba Hasan Sadaqah
1315-1321



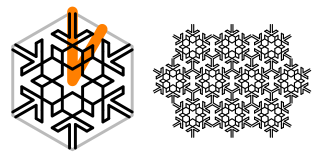
058 Qubba Hasan Sadaqah
1315-1321



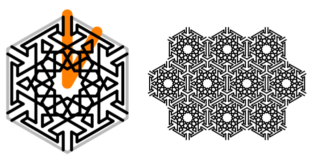
059 The mosque of Sultan Hasan
1356



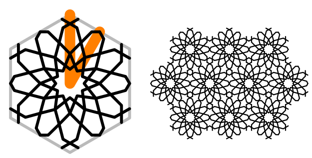
060 Qubba Hasan Sadaqah
1315-1321



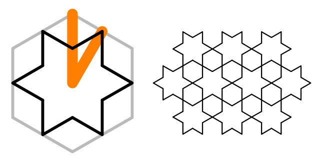
061 Qubba Hasan Sadaqah
1315-1321



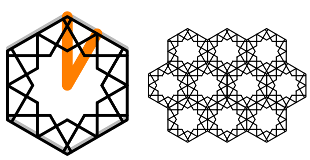
062 The madrasa of Amir Sarghitmish
1356



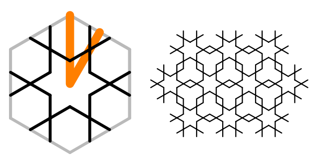
063 The mosque of Sultan Hasan
1356



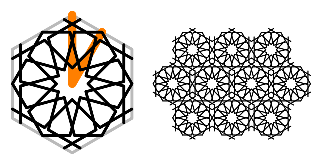
064 The mosque of Sultan Hasan
1356



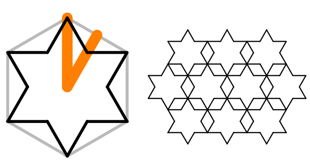
065 The mosque of Amir Aqsunqur
1346-1347



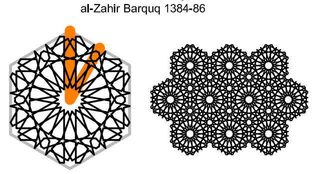
066 The mosque of Amir Aqsunqur
1346-1347



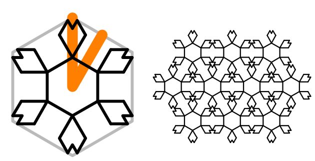
067 The mosque of Amir Aqsunqur
1346-1347



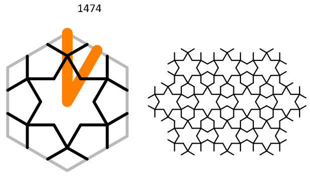
068 The madrasa-khanqah of Sultan
al-Zahir Barquq 1384-86



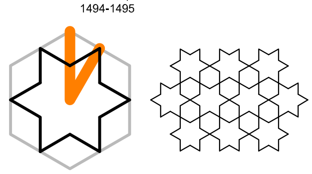
069 Masjid al-Sultan Qaytbay
1472-1474



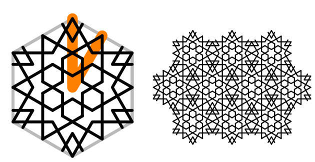
070 Jami' al-Azhar
1474



071 The mosque of Amir Azbak al-Yusufi
1494-1495

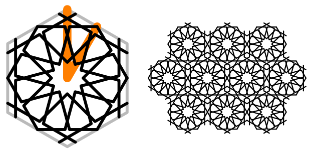


072 The mosque of Amir Qijmas al-Ishaqi
1479-1481



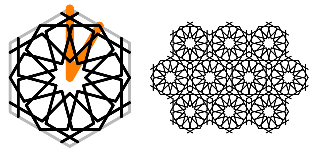
073

The mosque of Amir Qijmas al-Ishaqi
1479-1481



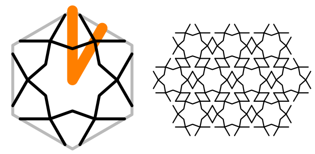
074

The madrasa and khanqah of Sultan al-Ghuri
1503-1504



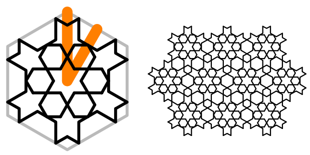
075

The madrasa and khanqah of Sultan al-Ghuri
1503-1504



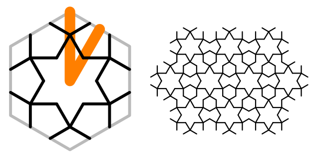
076

The madrasa and khanqah of Sultan al-Ghuri
1503-1504



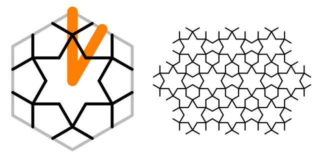
077

The religious-funerary complex of Amir Qurqumas
1503-1504



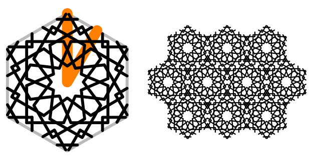
078

The religious-funerary complex of Amir Qurqumas
1503-1504



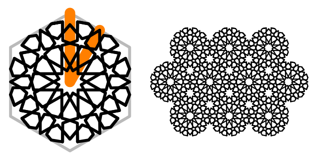
079

Sultan al-Mu'ayyad Shaykh Complex
1414-1421



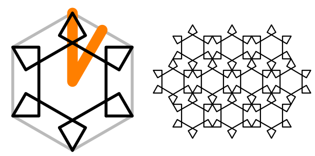
080

Sultan al-Mu'ayyad Shaykh Complex
1414-1421



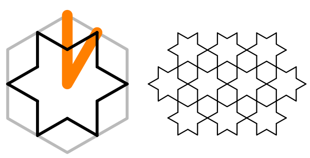
081

Jami' al-Azhar
1474



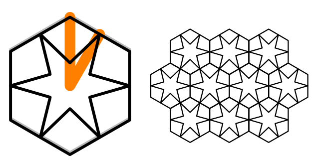
082

Jami' al-Azhar
1501-1516



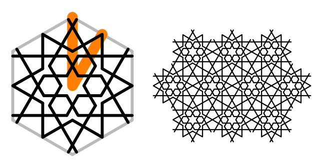
083

Jami' ibn Tulun
1296-1297



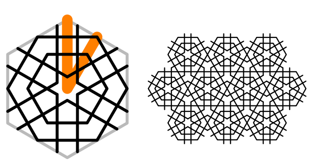
084

Jami' ibn Tulun
1296-1297



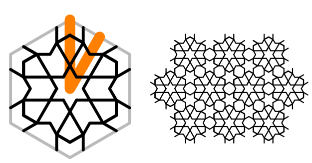
085

Jami' ibn Tulun
1296-1297



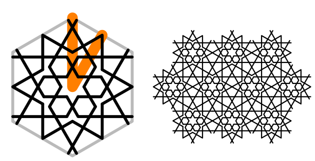
086

Jami' ibn Tulun
1296-1297



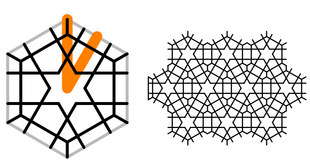
087

Jami' ibn Tulun
1296-1297



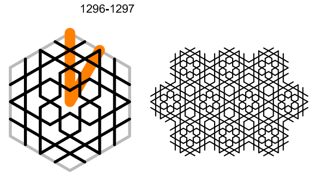
088

Jami' ibn Tulun
1296-1297



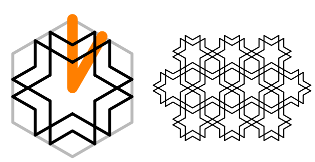
089

Jami' ibn Tulun
1296-1297

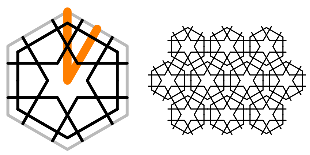


090

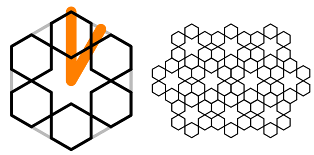
Jami' ibn Tulun
1296-1297



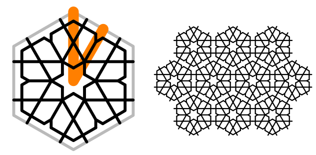
091 Jami' ibn Tulun
1296-1297



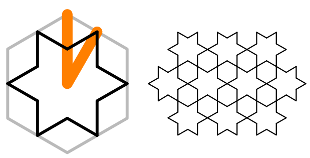
092 Jami' ibn Tulun
1296-1297



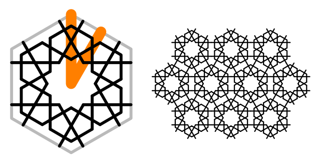
093 Jami' ibn Tulun
1296-1297



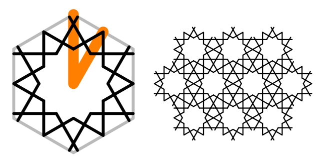
094 Madrasa al-Kamaliyya al-'Adimiyya



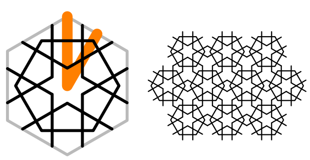
095 Jami' ibn Tulun
1296-1297



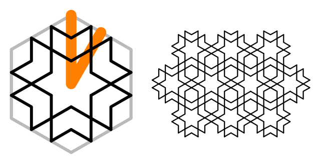
096 Jami' ibn Tulun
1296-1297



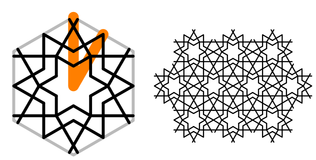
097 Jami' ibn Tulun
1296-1297



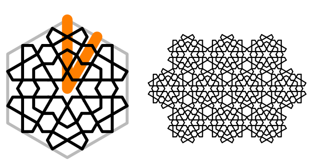
098 Jami' ibn Tulun
1296-1297



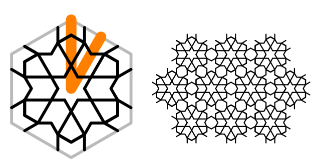
099 Jami' ibn Tulun
1296-1297



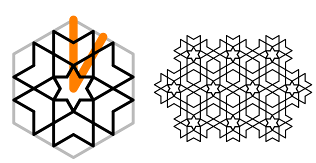
100 Jami' ibn Tulun
1296-1297



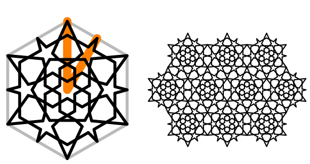
101 Jami' ibn Tulun
1296-1297



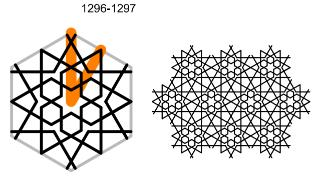
102 Jami' ibn Tulun
1296-1297



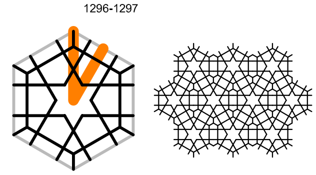
103 Madrasa al-Kamaliyya al-'Adimiyya



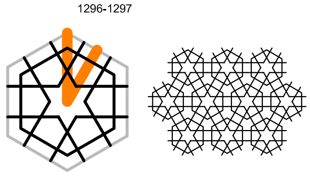
104 Jami' ibn Tulun
1296-1297



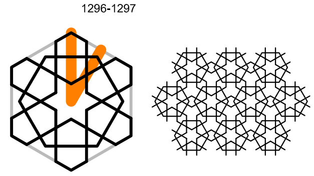
105 Jami' ibn Tulun
1296-1297



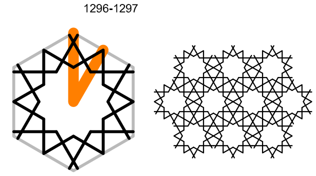
106 Jami' ibn Tulun
1296-1297



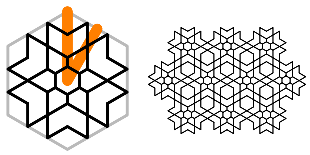
107 Jami' ibn Tulun
1296-1297



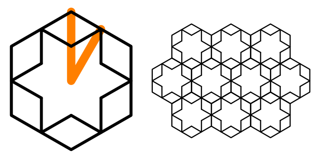
108 Jami' ibn Tulun
1296-1297



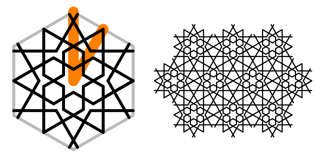
109 Jami' ibn Tulun
1296-1297



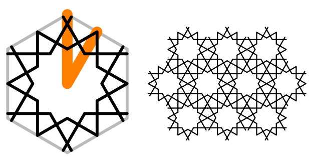
110 Jami' ibn Tulun
1296-1297



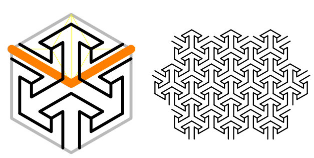
111 Jami' ibn Tulun
1296-1297



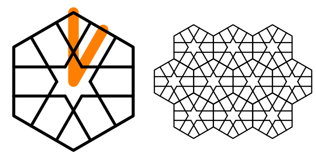
112 Jami' ibn Tulun
1296-1297



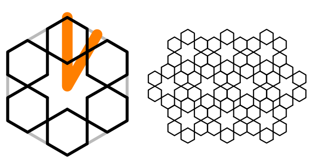
113 Madrasa al-Zahiriyya
1277



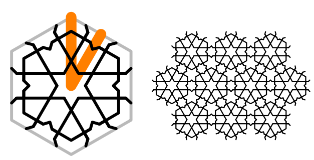
114 Jami' ibn Tulun
1296-1297



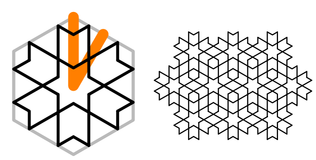
115 Jami' ibn Tulun
1296-1297



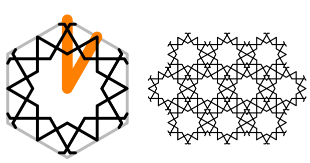
116 Jami' ibn Tulun
1296-1297



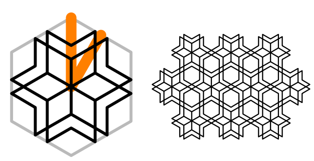
117 Jami' ibn Tulun
1296-1297



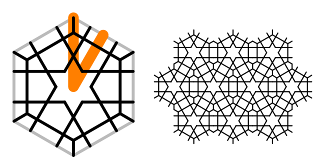
118 Jami' ibn Tulun
1296-1297



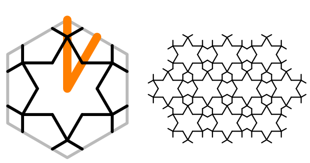
119 Jami' ibn Tulun
1296-1297



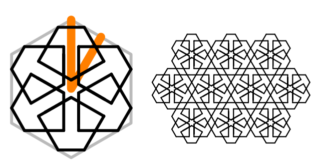
120 Jami' ibn Tulun
1296-1297



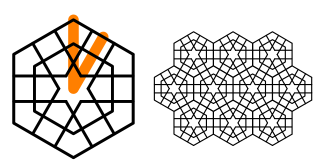
00121 Madrasa al-Firdaws



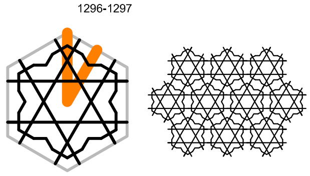
00122 Masjid al-Saghir
1168



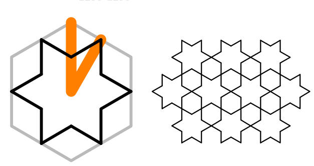
00123 Jami' ibn Tulun
1296-1297



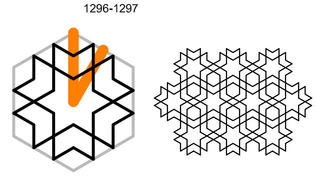
00124 Jami' ibn Tulun
1296-1297



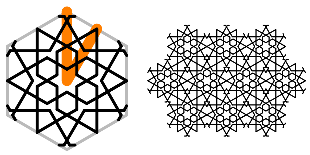
00125 Madrasa al-Firdaws
1235-1236



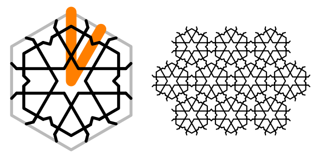
00126 Jami' ibn Tulun
1296-1297



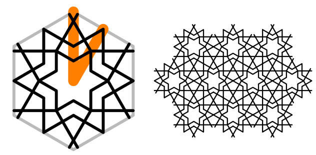
127 Jami' ibn Tulun
1296-1297



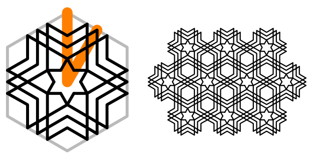
128 Jami' ibn Tulun
1296-1297



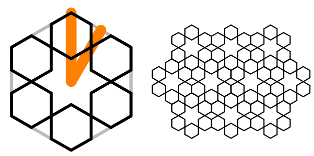
129 Jami' ibn Tulun
1296-1297



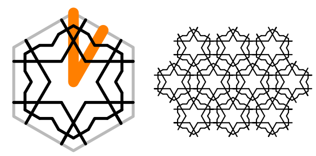
130 Jami' ibn Tulun
1296-1297



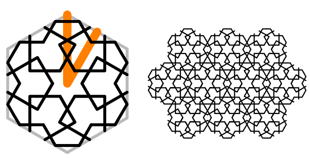
131 Jami' ibn Tulun
1296-1297



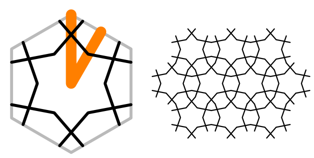
132 Jami' ibn Tulun
1296-1297



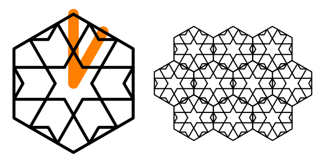
133 Khanqah al-Farafra



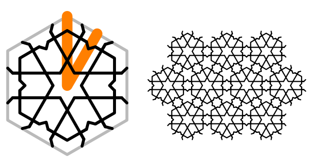
134 Jami' ibn Tulun
1296-1297



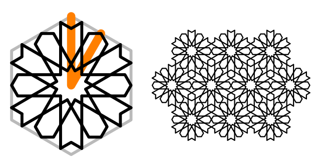
135 Jami' ibn Tulun
1296-1297



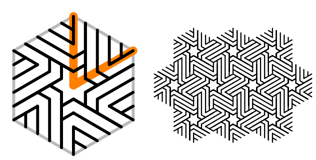
136 Jami' ibn Tulun
1296-1297



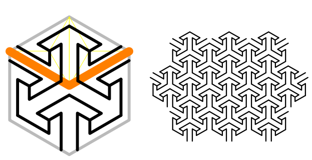
137 Alhambra
14th century



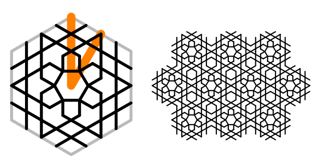
138 Alhambra
14th century



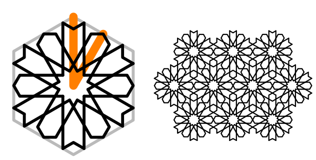
139 Alhambra
14th century



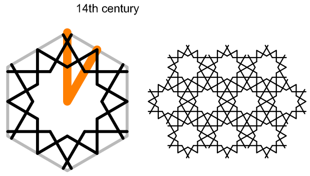
140 Alhambra
14th century



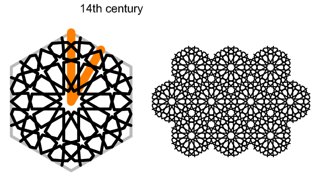
141 Alhambra
14th century



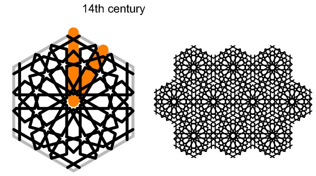
142 Alhambra
14th century



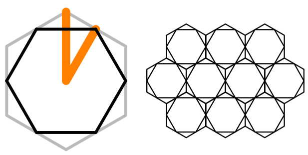
143 Alhambra
14th century



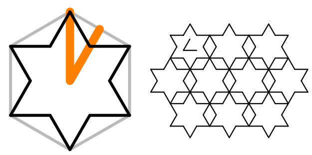
144 Alhambra
14th century



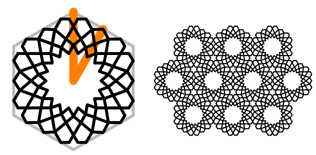
145 Ziyarat-i Khwaja Abu Nasr Parsa
1460



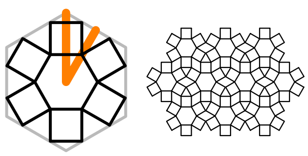
146 Ziyarat-i 'Abd Allah Ansari
1405-1429



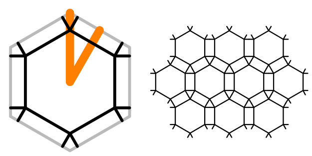
147 Ziyarat-i 'Abd Allah Ansari
1405-1429



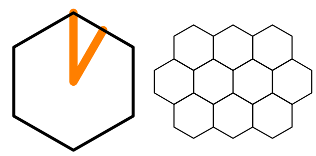
148 Ziyarat-i 'Abd Allah Ansari
1405-1429



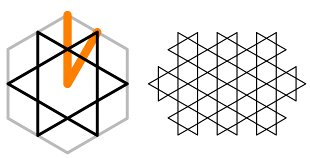
149 Abdullah Ansari Shrine Complex Restoration
1425-1499



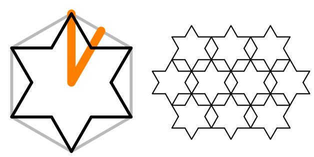
150 Abdullah Ansari Shrine Complex Restoration
1425-1499



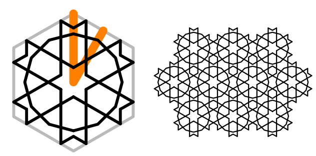
151 Info: Madrasa-i Gawhar Shad
1425-1499



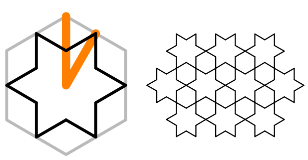
152 Info: Madrasa-i Gawhar Shad
1425-1499



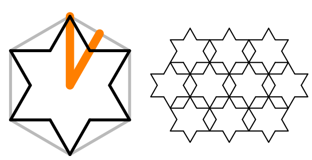
153 Masjid-i Jami'
1498-1500



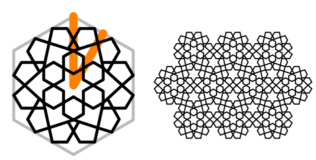
154 Masjid-i Jami'
1498-1500



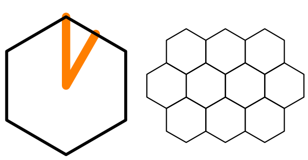
155 Masjid-i Jami'
1498-1500



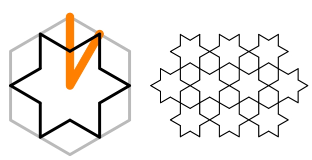
156 Alhambra
14th century



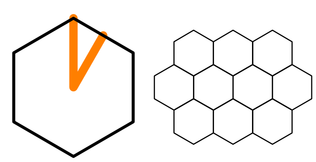
157 Rowze-i Sharif
1480-1481



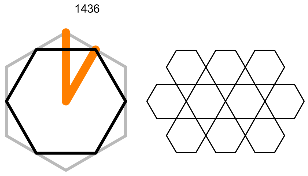
158 Rowze-i Sharif
1480-1481



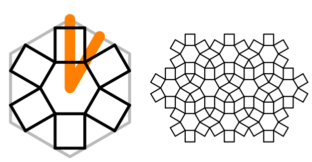
159 Nur al-Din Ni'matullah Vali Shrine
1436



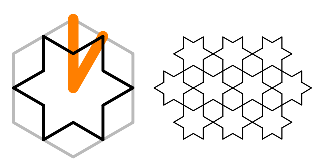
160 Nur al-Din Ni'matullah Vali Shrine
1436



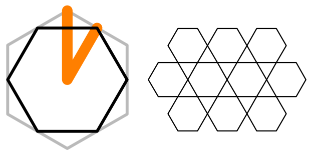
161 Masjid-i Gawhar Shad
1416-1418



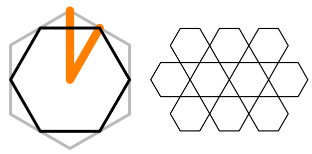
162 Masjid-i Gawhar Shad
1416-1418



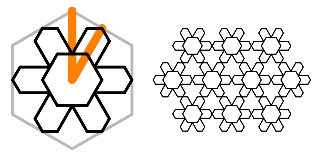
163 Masjid-i Gawhar Shad
1416-1418



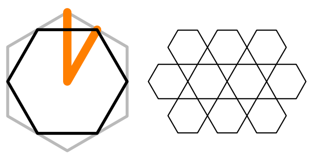
164 Shaykh Zayn Al-Din Mausoleum
1444



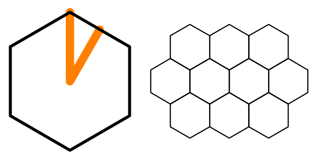
165 Masjid-i Mir Chaqmaq
1436-1437



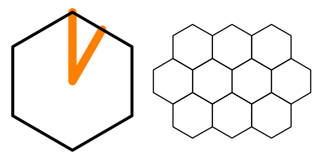
166 Masjid-i Mir Chaqmaq
1436-1437



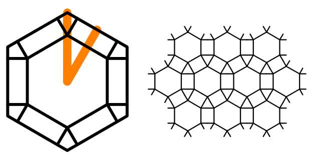
167 Masjid-i Mir Chaqmaq
1436-1437



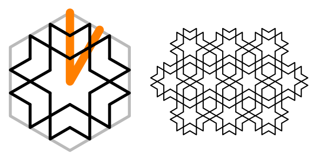
168 Mazar-i Khwaja Ahmad Yasavi
1389



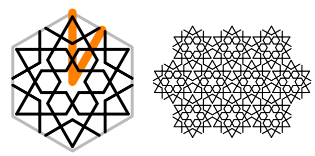
169 Anonymous Tomb
1385



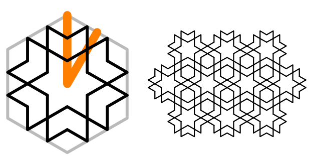
170 Anonymous Tomb
1385



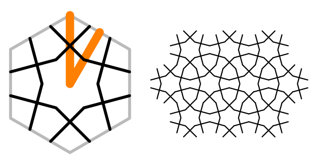
171 Anonymous Tomb
1385



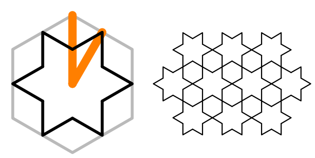
172 Anonymous Tomb
1385



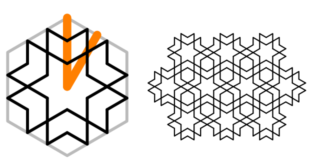
173 Anonymous Tomb
1385



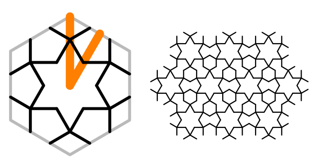
174 Anonymous Tomb
1385



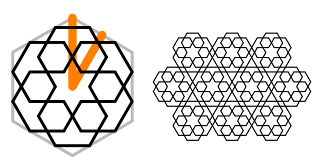
00175 Ishrat Khana Tomb
1385



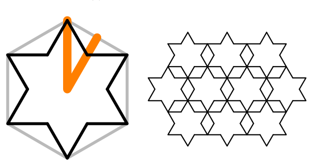
176 Ishrat Khana Tomb
1385



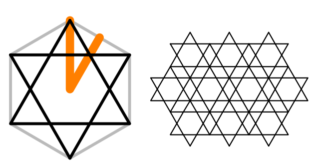
177 Ishrat Khana Tomb
1385



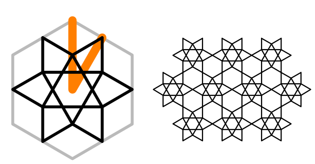
178 Khwaja Akhrar Ensemble
1490



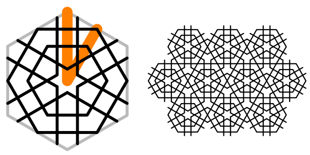
179 Maqbara-i Amir Husayn bin Tughluq Tekin
1376



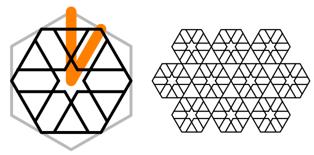
180 Maqbara-i Amir Husayn bin Tughluq Tekin
1376



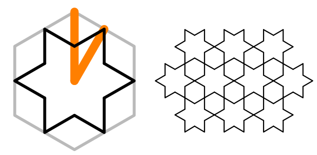
181 Maqbara-i Shad-i Mulk Aqa
1371



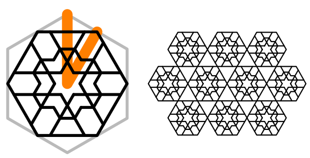
182 Maqbara-i Shad-i Mulk Aqa
1371



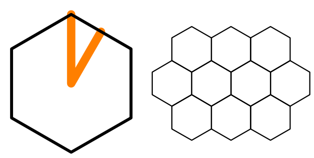
183 Maqbara-i Shad-i Mulk Aqa
1371



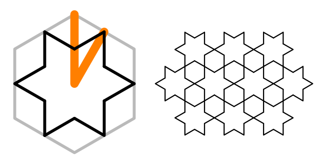
184 Maqbara-i Shad-i Mulk Aqa
1371



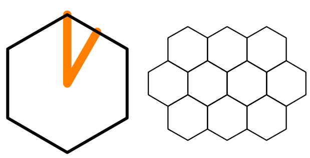
185 Maqbara-i Shirin Bika Aqa
1385



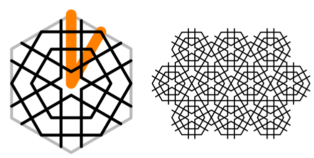
186 Masjid-i Bibi Khanum
1399



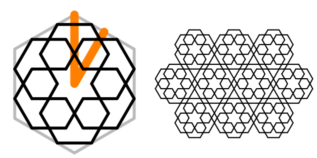
187 Mausoleum of 1361



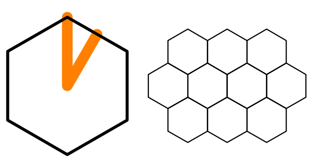
188 Mausoleum of 1361



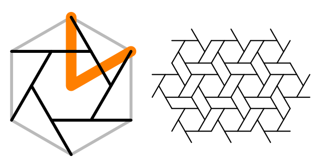
189 Rasd-khaneh-i Ulugh Beg
1420



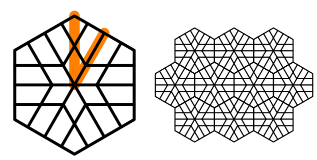
190 Ruhabad Mausoleum
1380



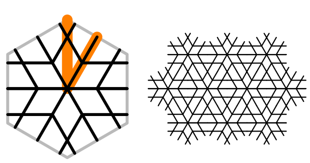
191 Aq Saray Palace
1379



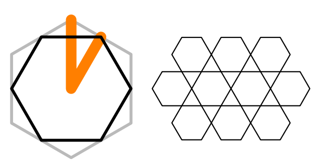
192 Dar al-Siyadah Complex
1375



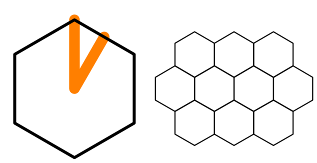
193 Dar al-Siyadah Complex
1375



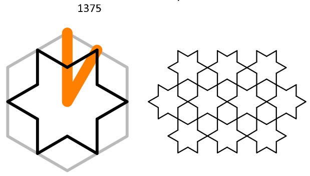
194 Dar al-Siyadah Complex
1375



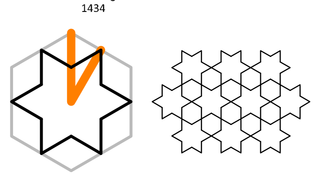
195 Dar al-Siyadah Complex
1375



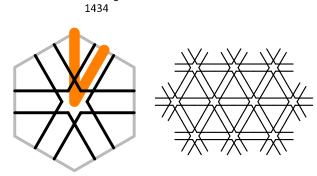
196 Dar al-Tilavah Complex Restoration
1375



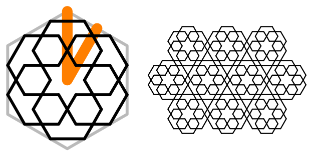
197 Aramgah-i Shah-i Zindeh
1434



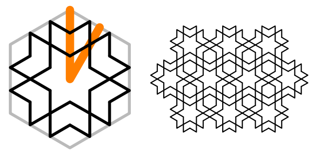
198 Aramgah-i Shah-i Zindeh
1434



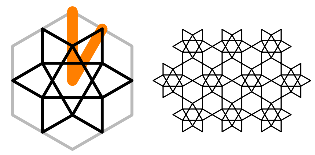
199 Aramgah-i Shah-i Zindeh
1434



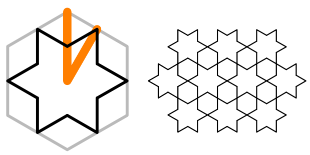
200 Aramgah-i Shah-i Zindeh
1434



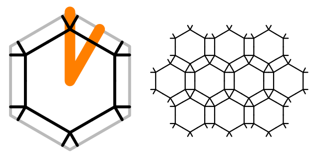
201 Aramgah-i Shah-i Zindeh
1434



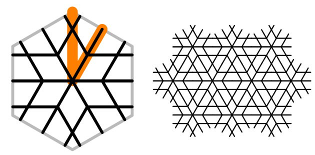
202 Kalyan Mosque
14th



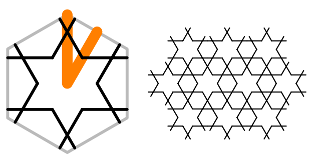
203 Gur-i Amir
1405



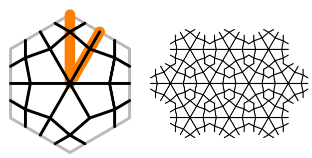
204 Gur-i Amir
1405



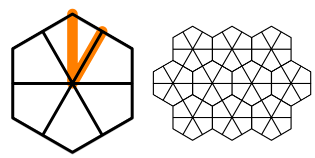
205 Gur-i Amir
1405



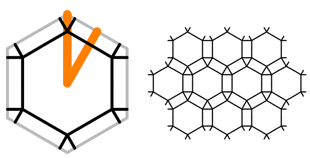
206 Gur-i Amir
1405



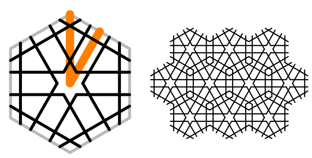
207 Gur-i Amir
1405



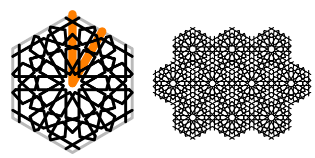
208 Ulugh Beg Madrasa
1417



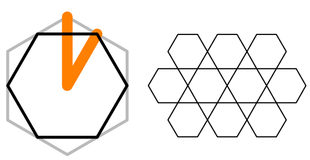
209 Gombad-i Chisht-e Sharif
1167



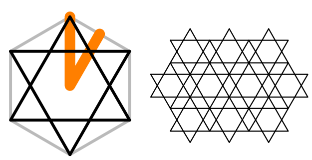
210 Madrasa al-'Attarin
1323



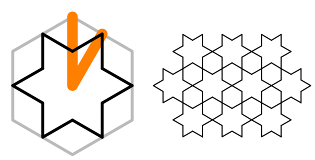
211 Amir Taz Palace Conservation
1352



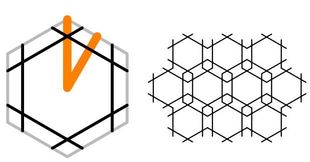
212 Mazar-i Khwaja Ahmad Yasavi
1389



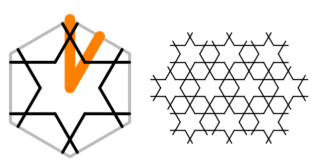
213 Jami' al-Aqsa
14th century



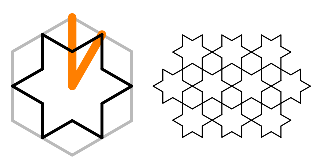
214 Madrasa al-Mustansiriyya
1227-1234



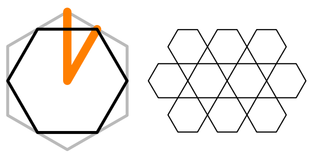
215 Mezquita de Córdoba
1236



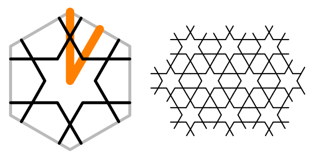
216 Jami' al-Azhar
1474



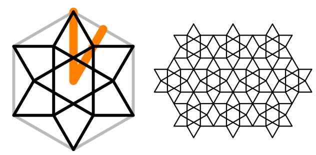
217 Jami' al-Aqsa
14th century



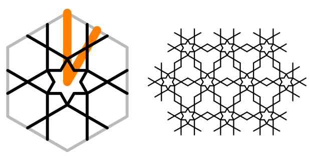
218 Madrasa al-Sahrij
1333



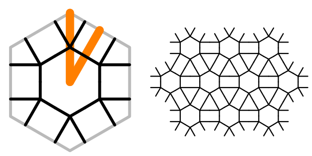
219 Khanqah-i Shaykh 'Abd al-Samad
1304 - 1325



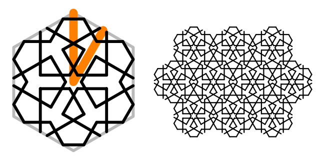
220 Khanqah-i Shaykh 'Abd al-Samad
1304 - 1325



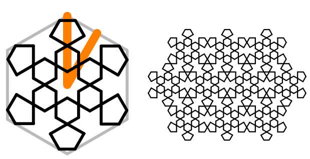
221 Khanqah-i Shaykh 'Abd al-Samad
1304 - 1325



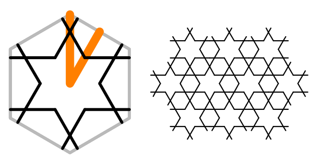
222 Mömina Xatun Türbasi
1186-1187



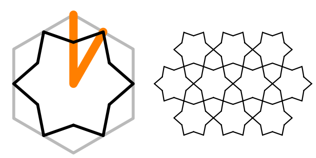
223 Mashhad al-Sayyida Ruqayya
1133



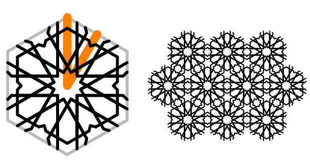
224 Friday Mosque of Yazd
1324



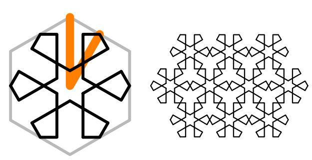
225 Friday Mosque of Yazd
1324



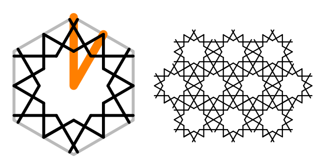
226 Sultan Han
1232-1236



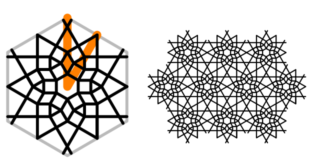
227 Qubba al-Barudiyyin
1120



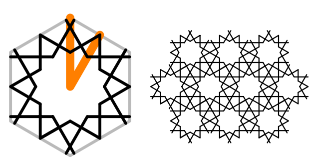
228 Fragment from Konya
12th



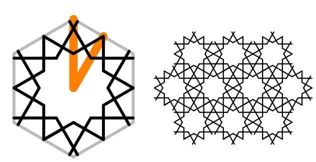
229 Madrasa wa Khanqah Salar wa Sanjar al-Jawli
1303



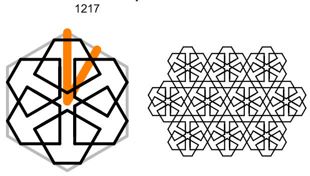
230 Alhambra palace
14th century



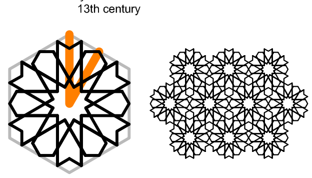
231 Ulu Camii ve Darüssifa
1228-1229



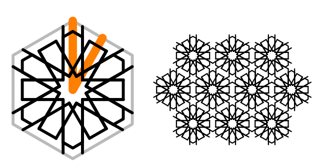
232 Izzeddin Keykavus Darüssifasi
1217



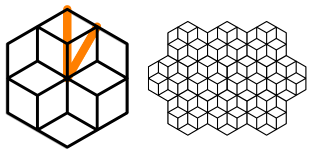
233 Çifte Minareli Medrese
13th century



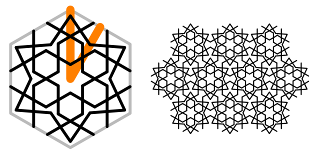
234 imaret of ibrahim bey of konya
1451



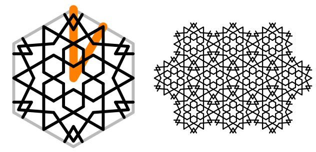
235 Madrasa al-Mustansiriyya
1227-1234



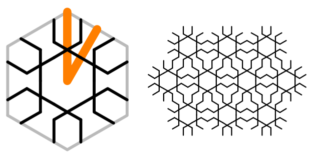
236 Qasr al-Amir Tashtimur
1366-1377



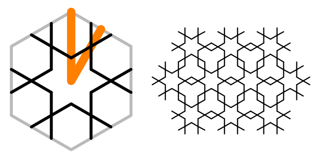
237 Masjid al-Sultan Qaytbay
1472-1474



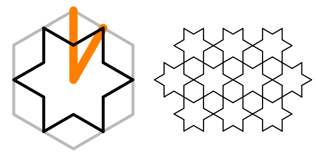
238 Madrasa al-Mustansiriyya
1227-1234



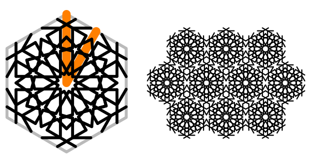
239 Madrasa al-Mustansiriyya
1227-1234



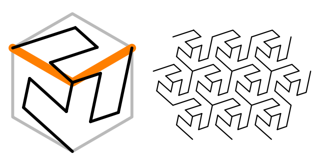
240 Madrasa al-Mustansiriyya
1227-1234



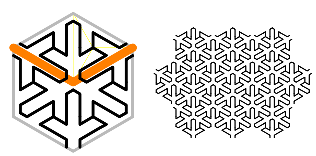
241 Alhambra
14th century



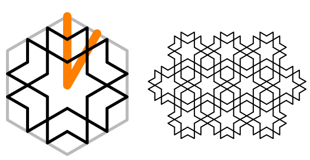
242 Gunbad-i Alaviyyan
1150



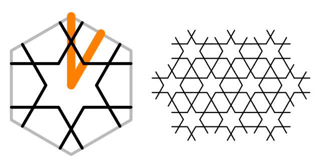
243 Gunbad-i Alaviyyan
1150



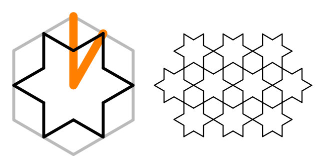
244 Masjid-i Jami'-i Abyaneh
1073-1074



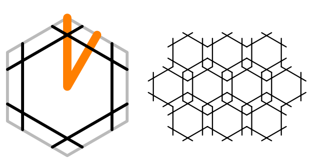
245 Masjid-i Jami'-i Abyaneh
1073-1074



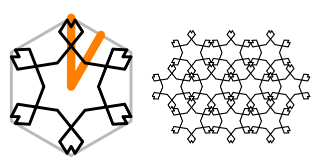
246 Mashhad-i Bayazid Bastami



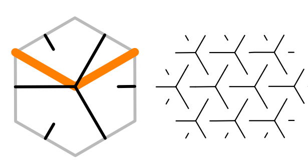
247 Mashhad-i Bayazid Bastami



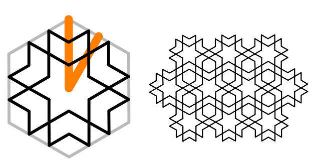
248 Mashhad-i Bayazid Bastami



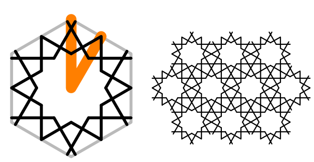
249 Masjid-i Jami' Golpayegan
1105-1118



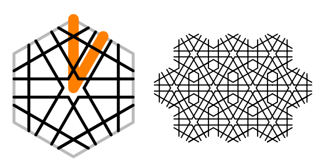
250 Masjid-i Jami' Golpayegan
1105-1118



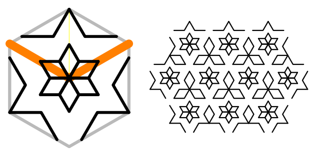
251 Masjid-i Jami' Golpayegan
1105-1118



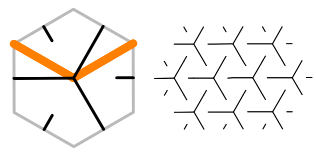
252 Masjid-i Jami'-i Abyaneh
1073-1074



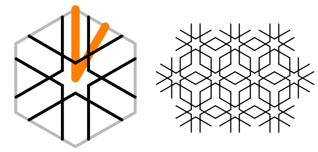
253 Masjid-i Jami'-i Sangan-i Pa'in
12th century



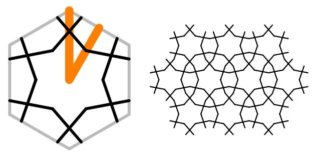
254 Pir-i Takestan Mausoleum



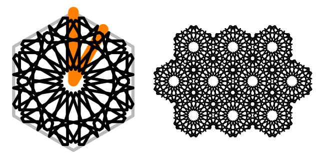
255 Mashhad-i Bayazid Bastami



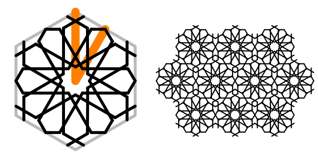
256 Esrefoglu Süleyman Bey Camii



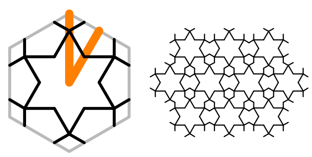
257 Mevlana Celaleddin Rumi Türbesi ve Dergahi



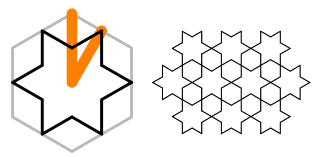
258 Mevlana Celaleddin Rumi Türbesi ve Dergahi



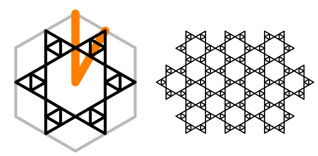
259 Mevlana Celaleddin Rumi Türbesi ve Dergahi



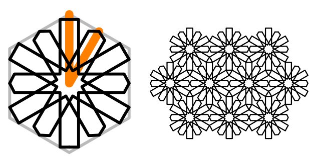
260 Mevlana Celaleddin Rumi Türbesi ve Dergahi



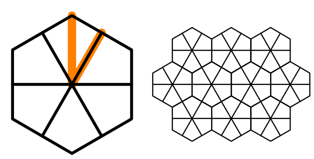
261 Isa Bey Camii



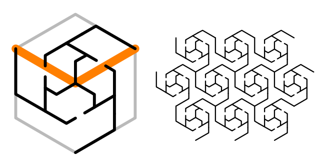
262 Sirçali Medrese



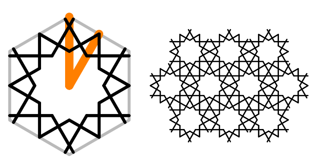
263 Sirçali Medrese



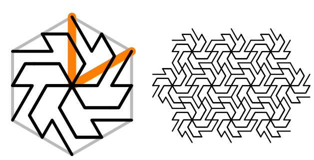
264 Ulu Camii



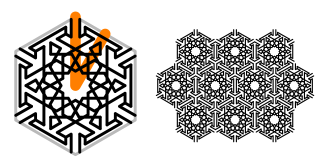
265 Van Ulu Camii



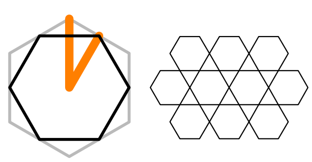
266 Van Ulu Camii



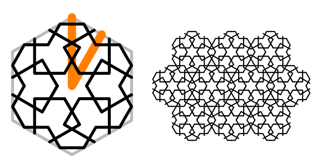
267 Jami' Taynal
1336



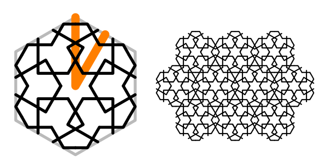
268 Jami' al-'Attar
1350



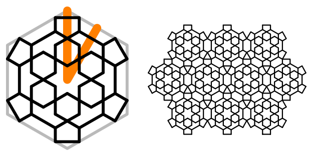
269 Bimaristan al-Arghuni
1354



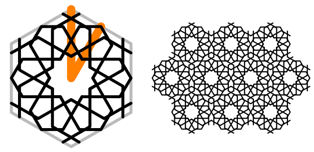
270 Jami' al-Hariri



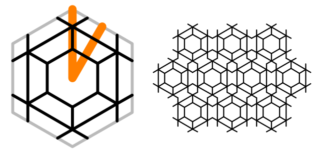
271 Madrasa al-Kamaliyya al-'Adimiyya



272 Madrasa al-Kamaliyya al-'Adimiyya



273 Qal'a Halab



Appendix B

IGP Explorer (the Simulation Program)

```
//Program Name: IGP EXPLORER
//Program description: SIMULATION PROGRAM FOR VISUALIZATION AND MORPHING HEXAGONAL BASED
ISLAMIC GEOMETRIC PATTERNS
//PROGRAMING LANGUAGE: PROCESSING
//MAY 15TH, 2017
//PROGRAM'S AUTHOR: MOSTAFA ALANI

//                                                                          IMPORT LIBRARY
//                                                                          IMPORT LIBRARY
//                                                                          IMPORT LIBRARY

import processing.dxf.*;

//
DECLARATION
//
DECLARATION
//
DECLARATION

boolean record;
int codeLength;
String[] Code, CodeDrawingTemp, CodeSpliting, Str, MorphCodeDis;
float[] CodeDrawing, CodeDrawingMorphed, CodeMorph, CodeToMorph;

//                                                             DECLARATION & INITIALIZATION
//                                                             DECLARATION & INITIALIZATION
//                                                             DECLARATION & INITIALIZATION

color[] theme = {#FFFFFF, #D7DADB, #FC4349, #2C3E50, #FC4349};
String myCode = "p6m:[3.3-90][5.8-60][8.8-71][6.7-90]";
String myCodeMorphed = "SAO"; // SameAsOrigional
int executeLock = 0, morphLock = 0, x1 = 0, y1 = 635;
float morphAddition = 0, morphSubstraction = 0;

//                                                                                 SETUP
//                                                                                 SETUP
//                                                                                 SETUP

void setup(){
  size(1280, 695, P3D);
    if (frame != null)
    {
      surface.setResizable(true);
    }
  background(theme[0]);
  smooth();
}

//                                                                                 DRAW
//                                                                                 DRAW
//                                                                                 DRAW

void draw(){
  background(theme[0]); //Background refresh
  viewPort(); //Viewports Display
  hovering(); //Buttons
  Code = split(myCode, ':'); //Code conversion
  codeLength = Code.length;
  execute(); // execute Button
  textSize(13);

  if(executeLock != 0){
    drawCodeR(1160, 120);
    fill(theme[2]);
    text("Symmetry Type= " + Code[0], 820, 270);
    CaseIdentifier(Code[1]);
  }
```

```
  if(morphLock == 1)
  {
    morph(920, 120);
  }
  noCursor(); //Curser Location
  fill(theme[1]);
  ellipse(mouseX,mouseY,1,1);
  fill(theme[4]);
  text((mouseX + " " + mouseY), mouseX, mouseY+20, width, height);

  patternGenerator();

  if(record) {
    beginRaw(DXF, "output.dxf");
  }
  if(record) {
    endRaw();
    record = false;
  }
}

//                                                                                VIEWPORT
//                                                                                VIEWPORT
//                                                                                VIEWPORT

void viewPort(){
  pushMatrix();

  noFill(); //Big viewport
  stroke(theme[3],150);
  rect(0,0,800,y1);
  fill(theme[3]);
  text("Pattern View", 0,0, width, height);

  fill(theme[1]);
  noStroke();
  rect(800+1,0,displayWidth,displayHeight);

  translate(800,0);

  fill(theme[0]); //Small viewports
  stroke(theme[3],150);
  rect(0, 0, 240, 240);
  fill(theme[3]);
  text("Morphed view", 0,0, width, height);

  fill(theme[0]);
  stroke(theme[3],150);
  rect(240, 0, 240, 240);
  fill(theme[3]);
  text("Origional view", 240,0, width, height);

  popMatrix();
  pushMatrix();
  translate(10,370);
  fill(theme[3],150);
  popMatrix();
}

//                                                                                HOVERING
//                                                                                HOVERING
//                                                                                HOVERING

float hovering(){
  fill(255);
  rect(x1+53,y1,width,30);
  rect(x1+53,y1+30,width,30);
  fill(0, 150);
  text(myCode, x1+59,y1, width, height);
```

```
  if(morphLock == 1){
    MorphCodeDis = new String[CodeMorph.length];
    MorphCodeDis[0] = "[";
    for(int i=0; i<CodeMorph.length-2; i+=3){
      if(i!=0){MorphCodeDis[i] = "][";}
      if(i!=0){
        if(CodeDrawingTemp[i].equals("/"))
        {
          MorphCodeDis[i] = "]/[";
        }
      }
      MorphCodeDis[i+1] = str((CodeMorph[i+1]));
      MorphCodeDis[i+1] = MorphCodeDis[i+1] + "-";
      MorphCodeDis[i+2] = str(int(CodeMorph[i+2]));
      MorphCodeDis[i+3] = "]";
    }
    String newCode = join(MorphCodeDis,"");
    text("P6M:" + newCode, x1+59,y1+30, width, height);
    print(" NEW CODE :: " + newCode + "\n");
  }
  if(mouseX>x1+53){
    if(mouseX<width){
      if(mouseY>y1){
        if(mouseY<y1+30)
        {
          fill(255);
          rect(x1+53,y1,width,30);
          fill(0);
          text(myCode, x1+59,y1, width, height);
        }
      }
    }
  }
  return(float(myCode));
}

//                                                                          EXECUTION
//                                                                          EXECUTION
//                                                                          EXECUTION

void execute(){
  fill(0);
  rect(0,y1,50,15);
  rect(0,y1+30,50,15);
  fill(255,150);
  text("Execute",0,y1,width,height);
  text("Animate ",0,y1+30,width,height);
  if(mouseX>0){
    if(mouseX<500){
      if(mouseY>y1){
        if(mouseY<y1+20){
          if(mousePressed)
          {
            fill(255);
            rect(0,y1,50,15);
            fill(0,150);
            text("Execute",0,y1,width,height);

            String f = Code[1].replace("]", ",");
            f = f.replace("[", ",");
            f = f.replace("-", ",");
            CodeDrawingTemp = split(f, ',');

            executeLock = 1;
            morphLock = 1;
            CodeToMorph = float(split(f, ','));
            CodeMorph = float(CodeDrawingTemp);

            firstP1 = 1;
```

```
            firstP2= 0;

            secondP1 = 1;
            secondP2= 0;

            thirdP1 = 1;
            thirdP2= 0;
          }
        }
      }
    }
  }

  if(mouseX>0){
    if(mouseX<500){
      if(mouseY>y1+30){
        if(mouseY<y1+45){
          if(mousePressed)
          {
            fill(theme[4]);
            rect(0,y1+30,50,15);
            fill(0);
            text("Animate ",0,y1+30,width,height);
            animate();
          }
        }
      }
    }
  }
  if(morphLock ==1)
  {
    for(int y = 0; y < CodeMorph.length; y++)
    {
      if(y == CodeMorph.length-1)
        print(" END \n");
    }
  }
}

//                                                                              DRAW CODE
//                                                                              DRAW CODE
//                                                                              DRAW CODE

void drawCodeR(int x, int y){
  if(Code[0].equals("P6M") == true || Code[0].equals("p6m") == true ||
Code[0].equals("p6") == true || Code[0].equals("P6") == true || Code[0].equals("P31M") ||
Code[0].equals("p31m") || Code[0].equals("P3M1") || Code[0].equals("p3m1"))
  {
    pushMatrix();
    translate(x,y);

    CodeDrawing = float(CodeDrawingTemp);
    for(int z=0; z<CodeDrawingTemp.length-2; z+=3)
    {
      float tempAng = float(CodeDrawingTemp[z+2]);
      float tempDis = float(CodeDrawingTemp[z+1]);
      CodeDrawing[z+2] = sin(radians(tempAng)) * tempDis; // X coordinate
      CodeDrawing[z+1] = cos(radians(tempAng)) * tempDis; // Y coordinate
    }

    for(int c=0; c<6; c++)
    {
      stroke(theme[3]);
      strokeWeight(2.2);
      pushMatrix();
      if(CodeDrawing.length<6)
        text("Incorrect CODE" + "\n", 400,400,width,height);
      int iii = 0;
      for(int ii=0; ii+iii<CodeDrawing.length-6; ii+=3)
```

```
      {
        line(10*CodeDrawing[ii+1+iii], -10*CodeDrawing[ii+2+iii],
10*CodeDrawing[ii+4+iii], -10*CodeDrawing[ii+5+iii]);
        if(Code[0].equals("p6m") || Code[0].equals("P6M") || Code[0].equals("P3M1") ||
Code[0].equals("p3m1") || Code[0].equals("p31m") || Code[0].equals("P31M") ||
Code[0].equals("P31m") || Code[0].equals("p31M")){
          line(-10*CodeDrawing[ii+1+iii], -10*CodeDrawing[ii+2+iii], -
10*CodeDrawing[ii+4+iii], -10*CodeDrawing[ii+5+iii]);
        }
        if(Code[0].equals("p31m") || Code[0].equals("P31M") || Code[0].equals("P31m") ||
Code[0].equals("p31M")){
          pushMatrix();
          rotate(radians(120));
          translate(0,100);
          line(10*CodeDrawing[ii+1+iii], -10*CodeDrawing[ii+2+iii],
10*CodeDrawing[ii+4+iii], -10*CodeDrawing[ii+5+iii]);
          line(-10*CodeDrawing[ii+1+iii], -10*CodeDrawing[ii+2+iii], -
10*CodeDrawing[ii+4+iii], -10*CodeDrawing[ii+5+iii]);
          rotate(radians(120));
          translate(0,100);
          line(-10*CodeDrawing[ii+1+iii], -10*CodeDrawing[ii+2+iii], -
10*CodeDrawing[ii+4+iii], -10*CodeDrawing[ii+5+iii]);
          rotate(radians(-120));
          line(10*CodeDrawing[ii+1+iii], -10*CodeDrawing[ii+2+iii],
10*CodeDrawing[ii+4+iii], -10*CodeDrawing[ii+5+iii]);
          popMatrix();
        }

        if(CodeDrawingTemp[ii+6+iii].equals("/"))
        {
          iii+=3;
        }
        fill(theme[4]);
      }
      popMatrix();
      if(Code[0].equals("p6m") || Code[0].equals("P6M") || Code[0].equals("P6") ||
Code[0].equals("p6")){
        rotate(radians(60));
      }
      if(Code[0].equals("p3") || Code[0].equals("P3") || Code[0].equals("P3M1") ||
Code[0].equals("p3m1") || Code[0].equals("P31M") || Code[0].equals("p31m")){
        rotate(radians(120));
      }
    }
    popMatrix();
  }
}

//                                                                      ANALYSIS
//                                                                      ANALYSIS
//                                                                      ANALYSIS

String TQp;
String MG;

void CaseIdentifier(String g){
  int f =70;
  text("Characteristics of the FU: ", 820, 310);

  //Counting Segment
  int Count;
  String[] S = splitTokens(g, "[]");
  Count = S.length-1;
  for(int i=0; i<S.length; i++){
    if(S[i].equals("/")){
      Count-=2;
    }
  }
  text("Count of Segments = " + Count, 820, 260+f);
```

```
  println(Count);

  //Q/TP
  println(g);
  String[] Sarray = splitTokens(g,"[/]");

  IntDict Search = new IntDict();
  for(int i=0; i<Sarray.length; i++){
    Search.increment(Sarray[i]);
  }
  Search.sortValuesReverse();
  int[] counts = Search.valueArray();
  String[] SEARCH = Search.keyArray();
  println(Search);

  if(counts[0] == 1){
    println("QV = " + 0);
    TQp = "0";
  }
  if(counts[0] > 1 && counts[1] == 1){
    println("QV = " + 1);
    TQp = "1";
  }
  if(counts[0] > 1 && counts[1] > 1 && counts[2] == 1){
    println("QV = " + 2);
    TQp = "2";
  }
  if(counts[0] > 1 && counts[1] > 1 && counts[2] > 1 && counts[3] == 1){
    println("QV = " + 3);
    TQp = "3";
  }
  if(counts[0] > 1 && counts[1] > 1 && counts[2] > 1 && counts[3] > 1 && counts[4] == 1){
    println("QV = " + 4);
    TQp = "4";
  }
  if(counts[0] > 1 && counts[1] > 1 && counts[2] > 1 && counts[3] > 1 && counts[4] > 1 &&
counts[5] == 1){
    println("QV = " + 5);
    TQp = "5";
  }
  if(counts[0] > 1 && counts[1] > 1 && counts[2] > 1 && counts[3] > 1 && counts[4] > 1
&& counts[5] > 1 && counts[6] == 1){
    println("QV = " + 6);
    TQp = "6";
  }
  if(counts[0] > 1 && counts[1] > 1 && counts[2] > 1 && counts[3] > 1 && counts[4] > 1
&& counts[5] > 1 && counts[6] > 1 && counts[7] == 1){
    println("QV = " + 7);
    TQp = "7";
  }
  if(counts[0] > 1 && counts[1] > 1 && counts[2] > 1 && counts[3] > 1 && counts[4] > 1
&& counts[5] > 1 && counts[6] > 1 && counts[7] > 1 && counts[8] == 1){
    println("QV = " + 8);
    TQp = "8";
  }
  if(counts[0] > 1 && counts[1] > 1 && counts[2] > 1 && counts[3] > 1 && counts[4] > 1
&& counts[5] > 1 && counts[6] > 1 && counts[7] > 1 && counts[8] > 1 && counts[9] == 1){
    println("QV = " + 9);
    TQp = "9";
  }

  text("Count of Points = " + counts.length, 820, 280+f);
  text("Count of Triple/Quadrable Connection Point = " + TQp, 820, 300+f);

  //Polyline
  int countPolylines = 1;
  String[] S1 = splitTokens(g, "[]");
  for(int i=0; i<S1.length; i++){
    if(S1[i].equals("/")){
```

```
      countPolylines++;
    }
  }
  print("Polyline count " + (countPolylines) + " ", 820, 320+f );
  text("Count of Polylines = " + countPolylines, 820, 340+f);

  //Group
  if(countPolylines==1){
    MG="A";
  }
  if(countPolylines==2){
    MG="B";
  }
  if(countPolylines==3){
    MG="C";
  }
  if(countPolylines==4){
    MG="D";
  }
  if(countPolylines==5){
    MG="E";
  }
  if(countPolylines==6){
    MG="F";
  }
  text("Morphological Group = " + MG, 820, 380+f);
  text("Specific Morphological Group = " + MG+TQp, 820, 400+f);
}

//                                                                         MORPH
//                                                                         MORPH
//                                                                         MORPH

void morph(int x, int y){
  pushMatrix();
  translate(x,y);
  for(int z=0; z<CodeMorph.length-2; z+=3)
  {
    float tempAng = (CodeMorph[z+2]);
    float tempDis = (CodeMorph[z+1]);

    CodeToMorph[z+2] = sin(radians(tempAng)) * tempDis; // X coordinate
    CodeToMorph[z+1] = cos(radians(tempAng)) * tempDis; // Y coordinate
  }

  for(int c=0; c<6; c++)
  {
    pushMatrix();
    if(CodeToMorph.length<6){
      text("Incorrect CODE" + "\n", 400,400,width,height);
    }
    int iii = 0;
    for(int ii=0; ii+iii<CodeToMorph.length-6; ii+=3)
    {
        line(10*CodeToMorph[ii+1+iii], -10*CodeToMorph[ii+2+iii],
10*CodeToMorph[ii+4+iii], -10*CodeToMorph[ii+5+iii]);
        if(Code[0].equals("p6m") || Code[0].equals("P6M") || Code[0].equals("P3M1") ||
Code[0].equals("p3m1") || Code[0].equals("p31m") || Code[0].equals("P31M") ||
Code[0].equals("P31m") || Code[0].equals("p31M")){
          line(-10*CodeToMorph[ii+1+iii], -10*CodeToMorph[ii+2+iii], -
10*CodeToMorph[ii+4+iii], -10*CodeToMorph[ii+5+iii]);
        }
        if(Code[0].equals("p31m") || Code[0].equals("P31M") || Code[0].equals("P31m") ||
Code[0].equals("p31M")){
          pushMatrix();
          rotate(radians(120));
          translate(0,100);
          line(10*CodeToMorph[ii+1+iii], -10*CodeToMorph[ii+2+iii],
10*CodeToMorph[ii+4+iii], -10*CodeToMorph[ii+5+iii]);
```

```
          line(-10*CodeToMorph[ii+1+iii], -10*CodeToMorph[ii+2+iii], -
10*CodeToMorph[ii+4+iii], -10*CodeToMorph[ii+5+iii]);
          rotate(radians(120));
          translate(0,100);
          line(-10*CodeToMorph[ii+1+iii], -10*CodeToMorph[ii+2+iii], -
10*CodeToMorph[ii+4+iii], -10*CodeToMorph[ii+5+iii]);
          rotate(radians(-120));
          line(10*CodeToMorph[ii+1+iii], -10*CodeToMorph[ii+2+iii],
10*CodeToMorph[ii+4+iii], -10*CodeToMorph[ii+5+iii]);
          popMatrix();
        }
      if(CodeDrawingTemp[ii+6+iii].equals("/"))
      {
        iii+=3;
      }
      fill(theme[4]);
    }
    popMatrix();
    if(Code[0].equals("p6m") || Code[0].equals("P6M") || Code[0].equals("P6") ||
Code[0].equals("p6")){
      rotate(radians(60));
    }
    if(Code[0].equals("p3") || Code[0].equals("P3") || Code[0].equals("P3M1") ||
Code[0].equals("p3m1") || Code[0].equals("P31M") || Code[0].equals("p31m")){
      rotate(radians(120));
    }
  }
  popMatrix();
}

//                                                                  PATTERN GENERATOR
//                                                                  PATTERN GENERATOR
//                                                                  PATTERN GENERATOR

void patternGenerator(){
  pushMatrix();
  translate(920,120);
  hexa();
  translate(240,0);
  hexa();
  popMatrix();

  pushMatrix();
  translate(100,100);
  scale(.5);
  for(int z=0; z<7; z++)
  {
    for(int i =0; i<8; i++)
    {
      if(executeLock == 0){
        hexa();
      }
      if(executeLock != 0){
        stroke(theme[3]);
        morph(0, 0);
      }
      translate(173.206,0);
    }
    if(z == 0 || z == 2 || z == 4 || z == 6 || z == 8)
      translate((-9*173.206)+(173.206/2),150);
    else
      translate((-8*173.206)+(173.206/2),150);
  }
  popMatrix();
}

//                                                                         HEXAGONAL
//                                                                         HEXAGONAL
//                                                                         HEXAGONAL
```

```
void hexa(){
  stroke(theme[3],50);
  line(0,-100,86.603,-50);
  rotate(PI/3);
  line(0,-100,86.603,-50);
  rotate(PI/3);
  line(0,-100,86.603,-50);
  rotate(PI/3);
  line(0,-100,86.603,-50);
  rotate(PI/3);
  line(0,-100,86.603,-50);
  rotate(PI/3);
  line(0,-100,86.603,-50);
  rotate(PI/3);
}

//                                                                                  ANIMATE
//                                                                                  ANIMATE
//                                                                                  ANIMATE

int reverseDirection = -1;

int firstP1 = 1;
int firstP2= 0;
float aimationSpeed = 0.01;

int secondP1 = 1;
int secondP2= 0;
float aimationSpeed3 = 0.01;

int thirdP1 = 1;
int thirdP2= 0;
float aimationSpeed4 = 0.01;

int fourthP1 = 1;
int fourthP2= 0;
float aimationSpeed5 = 0.01;

void animate()
{
  if(CodeMorph[1] <= limits( CodeMorph[2], CodeMorph[1]) && firstP1 == firstP2)
  {
    CodeMorph[1] += 1;  // CHANGE TO 0.01 TO RESTORE ACTUAL SPEED
    firstP1 +=1;

    if(CodeMorph[1] == limits( CodeMorph[2], CodeMorph[1]))
    {
      CodeMorph[1] = limits( CodeMorph[2], CodeMorph[1]);
    }

    if(CodeMorph[1] >10)
      CodeMorph[1] =10;
  }

  if(CodeMorph.length <=7 || secondP1 == secondP2) // Two or point points senerio
  {
    if(CodeMorph[4] < limits( CodeMorph[5], CodeMorph[4])+.01 || CodeMorph[4] == limits(
CodeMorph[5], CodeMorph[4]))
    {
      if(CodeMorph[4] == limits( CodeMorph[5], CodeMorph[4]) || CodeMorph[4] <= 0)
      {
        aimationSpeed = aimationSpeed * -1;
        firstP2 += 1;
      }
      CodeMorph[4] += aimationSpeed*10; // CHANGE TO aimationSpeed TO RESTORE ACTUAL
SPEED
      secondP1+=1;
    }
```

```
    if(CodeMorph[4] > limits( CodeMorph[5], CodeMorph[4])+.01) // to optimize and prevent
outlyers, i.e. points that are higher than LIMITS
      CodeMorph[4] = limits( CodeMorph[5], CodeMorph[4]);
  }

  if(CodeMorph.length ==10 || thirdP1 == thirdP2) // 10 means u have 3 points
  {
    if(CodeMorph[7] < limits( CodeMorph[8], CodeMorph[7])+.01 || CodeMorph[7] == limits(
CodeMorph[8], CodeMorph[7]))
    {
      if(CodeMorph[7] == limits( CodeMorph[8], CodeMorph[7]) || CodeMorph[7] <= 0)
      {
        aimationSpeed3 = aimationSpeed3 * -1;
        secondP2 += 1;
      }
      CodeMorph[7] += aimationSpeed3 * 10; // CHANGE TO aimationSpeed TO RESTORE ACTUAL
SPEED
      thirdP1 += 1;
    }
    if(CodeMorph[7] > limits( CodeMorph[8], CodeMorph[7])+.01) // to optimize and prevent
outlyers, i.e. points that are higher than LIMITS
      CodeMorph[7] = limits( CodeMorph[8], CodeMorph[7]);
  }

  if(CodeMorph.length ==13) // 13 means u have 4 points
  {
    if(CodeMorph[10] < limits( CodeMorph[11], CodeMorph[10])+.01 || CodeMorph[10] ==
limits( CodeMorph[11], CodeMorph[10]))
    {
      if(CodeMorph[10] == limits( CodeMorph[11], CodeMorph[10]) || CodeMorph[10] <= 0)
      {
        aimationSpeed4 = aimationSpeed4 * -1;
        thirdP2 += 1;
      }
      CodeMorph[10] += aimationSpeed4 * 5; // CHANGE TO aimationSpeed TO RESTORE ACTUAL
SPEED
    }
    if(CodeMorph[10] > limits( CodeMorph[11], CodeMorph[10])+.01) // to optimize and
prevent outlyers, i.e. points that are higher than LIMITS
      CodeMorph[10] = limits( CodeMorph[11], CodeMorph[10]);
  }

  if(CodeMorph.length ==16) // 16 means u have 5 points
  {
    if(CodeMorph[13] < limits( CodeMorph[14], CodeMorph[13])+.01 || CodeMorph[13] ==
limits( CodeMorph[14], CodeMorph[13]))
    {
      if(CodeMorph[13] == limits( CodeMorph[14], CodeMorph[13]) || CodeMorph[13] <= 0)
      {
        aimationSpeed5 = aimationSpeed5 * -1;
        fourthP2 += 1;
      }
      CodeMorph[13] += aimationSpeed5 * 5; // CHANGE TO aimationSpeed TO RESTORE ACTUAL
SPEED
    }
    if(CodeMorph[13] > limits( CodeMorph[14], CodeMorph[13])+.01) // to optimize and
prevent outlyers, i.e. points that are higher than LIMITS
      CodeMorph[13] = limits( CodeMorph[14], CodeMorph[13]);
  }
}

float result;
float limits( float angle, float distance)
{
  if(angle <=90)
  {
    float reverseAngle = 90 - angle;
    float missingAngle = 180 - 60 - reverseAngle;
    result = (10 * sin(radians(60)) / sin(radians(missingAngle)));
```

```
  }
  return result;
}

//                                                                              KEYPRESSED
//                                                                              KEYPRESSED
//                                                                              KEYPRESSED

void keyPressed() {
  if(mouseX>x1+53)
  if(mouseX<width)
    if(mouseY>y1)
      if(mouseY<y1+30)
      {
        if (keyCode == BACKSPACE) {
          if (myCode.length() > 0) {
            myCode = myCode.substring(0, myCode.length()-1);
          }
        } else if (keyCode == DELETE) {
          myCode = "";
        } else if (keyCode != SHIFT && keyCode != CONTROL && keyCode != ALT) {
          myCode = myCode + key;
        }
      }
  if (key == 'r')
    record = true;
}
```

Appendix C

Representational Code Analyzer (the Search Program)

```
//PROGRAM NAME: repCode analyzer
//PROGRAM DESCRIPTION: READ THE REPRESENTATIONAL CODES AND DETECT MORPHOLOGICAL
SIMILARITIES ON MULTIPLE LEVELS
//PROGRAMING LANGUAGE: PROCESSING
//MAY 15TH, 2017
//PROGRAM'S AUTHOR: MOSTAFA ALANI

color[] theme = {#E8E6EB, #84B1D9, #075473, #A62D12, #D94E41};// Color theme of the
visualization
Table t; // CSV(Excel) sheet
Geometry g; // Geometry has all information about particular historical design
Geometry[] G; // Object array for the above geometries
int readDataOnlyOnce=0; //So the code read the data only once
int textInhowverCounter = 0;
float growth =810.0;
String Original; int CADno;

//                                                                                SETUP
//                                                                                SETUP
//                                                                                SETUP

void setup(){
  size(500, 500);
  background(theme[0]);
  smooth();
  processData();
}

//                                                                                DRAW
//                                                                                DRAW
//                                                                                DRAW

void draw(){
  background(theme[0]);
  //below assign each value from CSV to temprory variable to prepare the transfer to the
object and put it in a particualr array index
  if(readDataOnlyOnce == 0)// So the file read data only once
  {
    for(int i =0; i<t.getRowCount(); i++)
    {
      TableRow tr = t.getRow(i);
      int x = tr.getInt("CAD#");   int d = tr.getInt("DATE");   int td = tr.getInt("TO-
DATE");   int c = tr.getInt("CENTURY"); String m = tr.getString("MONUMENT");   String r =
tr.getString("REGION");   String tow = tr.getString("TOWN");   String dy =
tr.getString("DYNASTY");   String mat = tr.getString("MATERIAL");   String fun =
tr.getString("FUNCTION");   String SY = tr.getString("SYMMETRY");   String SC =
tr.getString("SHAPE-CODE");   float SLR = tr.getFloat("SCALER"); float XL =
tr.getFloat("xLocation"); float YL = tr.getFloat("yLocation"); String C =
tr.getString("Case"); String seg = tr.getString("Seg"); String id = tr.getString("ID");
String mc = tr.getString("MC"); String N = tr.getString("Nominal"); String L =
tr.getString("List");
      G[i] = new Geometry(x, m, d, td, c, r, tow, dy, mat, fun, SY, SC, SLR, XL, YL, C,
seg, id, mc, N, L); //transfere to the object through constructor
      GroupIdentifier(G[i]);
    }
  }

  if(readDataOnlyOnce == 0)
  {
    //for(int i =36; i<37; i++)//
    for(int i =0; i<t.getRowCount(); i++)
    {
      Original = G[i].repCODE; CADno = G[i].CAD;
      //Analysis control keys
      print(G[i].CAD+ " ");
      //ID
      int identity =0;
      //LV1 con.
      int LV0 =0;
```

```
        //LV1
        int LV1 =0;
        //LV2
        int LV2 = 0;
        //LV3
        int LV3=0;
        //LV4
        int LV4=0;

        if(LV3==1){LevelThreeSE(G[i], G);}
        if(LV4==1){LevelFourSE(G[i], G);}

////////////        FLIPPING & SEARCHING ALL SMG EXCEPT A0, A1, B0, B1, & B2      //////

        if(G[i].Case.equals("A0")==false && G[i].Case.equals("A1")==false &&
G[i].Case.equals("B0")==false && G[i].Case.equals("B1")==false &&
G[i].Case.equals("B2")==false){
          if(identity==1){IdenticalMatch(G[i], G);} if(LV0==1){LevelZeroSE(G[i], G);}
          if(LV1==1){LevelOneSE(G[i], G);} if(LV2==1){LevelTwoSE(G[i], G);}
          flipCode(G[i].repCODE); G[i].repCODE=flipResults;
          if(identity==1){IdenticalMatch(G[i], G);} if(LV0==1){LevelZeroSE(G[i], G);}
          if(LV1==1){LevelOneSE(G[i], G);} if(LV2==1){LevelTwoSE(G[i], G);}
        }

////////////        FLIPPING & SEARCHING SMG A0      ////////////

        if(G[i].Case.equals("A0")){
          if(identity==1){IdenticalMatch(G[i], G);} if(LV0==1){LevelZeroSE(G[i], G);}
          if(LV1==1){LevelOneSE(G[i], G);} if(LV2==1){LevelTwoSE(G[i], G);}
          flipCode(G[i].repCODE); G[i].repCODE=flipResults;
          if(identity==1){IdenticalMatch(G[i], G);} if(LV0==1){LevelZeroSE(G[i], G);}
          if(LV1==1){LevelOneSE(G[i], G);} if(LV2==1){LevelTwoSE(G[i], G);}
          G[i].repCODE = Original;
        }

////////////        FLIPPING & SEARCHING SMG A1      ////////////

        if(G[i].Case.equals("A1")){
          if(identity==1){IdenticalMatch(G[i], G);} if(LV0==1){LevelZeroSE(G[i], G);}
          if(LV1==1){LevelOneSE(G[i], G);} if(LV2==1){LevelTwoSE(G[i], G);}
          flipCode(G[i].repCODE); G[i].repCODE=flipResults;
          if(identity==1){IdenticalMatch(G[i], G);} if(LV0==1){LevelZeroSE(G[i], G);}
          if(LV1==1){LevelOneSE(G[i], G);} if(LV2==1){LevelTwoSE(G[i], G);}
          G[i].repCODE = Original;
          partialA1Flip(G[i].repCODE, G[i]); G[i].repCODE=partialA1flipResults;
          if(identity==1){IdenticalMatch(G[i], G);} if(LV0==1){LevelZeroSE(G[i], G);}
          if(LV1==1){LevelOneSE(G[i], G);} if(LV2==1){LevelTwoSE(G[i], G);}
          flipCode(G[i].repCODE); G[i].repCODE=flipResults;
          if(identity==1){IdenticalMatch(G[i], G);} if(LV0==1){LevelZeroSE(G[i], G);}
          if(LV1==1){LevelOneSE(G[i], G);} if(LV2==1){LevelTwoSE(G[i], G);}
          G[i].repCODE = Original;
        }

////////////        FLIPPING & SEARCHING SMG B0      ////////////

        if(G[i].Case.equals("B0")){
          flipCaseB(G[i].repCODE, G[i], G, identity, LV0, LV1, LV2);
        }
//////////////      FLIPPING & SEARCHING SMG B1      ////////////

        if(G[i].Case.equals("B1")){
          caseB1Flip(G[i].repCODE, G[i], "");
          flipCaseB(G[i].repCODE, G[i], G, identity, LV0, LV1, LV2);
          flipCaseB(caseB1possibleCodeA, G[i], G, identity, LV0, LV1, LV2);
          flipCaseB(caseB1possibleCodeB, G[i], G, identity, LV0, LV1, LV2);
        }

//////////////      FLIPPING & SEARCHING SMG B2      ////////////
```

```
      if(G[i].Case.equals("B2")){
        caseB2Flip(G[i].repCODE, G[i], "");
        flipCaseB(G[i].repCODE, G[i], G, identity, LV0, LV1, LV2);
        flipCaseB(newB2code1, G[i], G, identity, LV0, LV1,LV2);
        flipCaseB(caseB2possibleCodeA, G[i], G, identity, LV0, LV1, LV2);
        flipCaseB(caseB2possibleCodeB, G[i], G, identity, LV0, LV1, LV2);
        flipCaseB(caseB2possibleCodeC, G[i], G, identity, LV0, LV1, LV2);
        flipCaseB(caseB2possibleCodeD, G[i], G, identity, LV0, LV1, LV2);
        flipCaseB(caseB2possibleCodeE, G[i], G, identity, LV0, LV1, LV2);
        flipCaseB(caseB2possibleCodeF, G[i], G, identity, LV0, LV1, LV2);
        flipCaseB(newB2code2, G[i], G, identity, LV0, LV1, LV2);
      }
      //special B2 case
      if(G[i].CAD == 234){
        if(identity==1){IdenticalMatch(G[i], G);} if(LV0==1){LevelZeroSE(G[i], G);}
        if(LV1==1){LevelOneSE(G[i], G);} if(LV2==1){LevelTwoSE(G[i], G);}
        flipCode(G[i].repCODE); G[i].repCODE=flipResults;
        if(identity==1){IdenticalMatch(G[i], G);} if(LV0==1){LevelZeroSE(G[i], G);}
        if(LV1==1){LevelOneSE(G[i], G);} if(LV2==1){LevelTwoSE(G[i], G);}
      }
      println();
      G[i].repCODE = Original;
    }
  }
  readDataOnlyOnce =1;
}

//                                                                       DATA FROM TABLES
//                                                                       DATA FROM TABLES
//                                                                       DATA FROM TABLES

void processData(){
  t = loadTable("data10.csv", "header");//CSV file name
  G = new Geometry[t.getRowCount()]; // array intialiatiazion and allocation
}

//                                                                         CLASS GEOMETRY
//                                                                         CLASS GEOMETRY
//                                                                         CLASS GEOMETRY

class Geometry{
  int  CAD, DATE, TODATE, CENTURY;
  float SCALER, XL, YL;
  String MONUMENT,REGION,TOWN,DYNASTY,MATERIAL,FUNCTION, SYMMETRY, repCODE, Case, Seg,
ID, MC, Nomi, List;
  IntDict conc = new IntDict();//Dictionar used in countPoints function belowreads
chuncks of angle and distance, for instanc "90-10" as one string to count correctlly
  IntDict countPoints = new IntDict();
  float Xlocation, ylocation;

//Constructor
  Geometry(int cad, String monument,int date, int todate, int century, String region,
String town, String dynasty, String material, String function, String Symmetry, String
repCode, Float scaler , Float xl, Float yl, String cases, String seg, String id, String
mc, String N, String L){
    CAD = cad;
    MONUMENT = monument;
    DATE = date;
    TODATE = todate;
    CENTURY = century;
    REGION = region;
    TOWN = town;
    DYNASTY = dynasty;
    MATERIAL = material;
    FUNCTION = function;
    SYMMETRY = Symmetry;
    repCODE = repCode;
    SCALER = scaler;
    XL = xl;
```

```
    YL = yl;
    Case = cases;
    Seg=seg;
    ID=id;
    MC = mc;
    Nomi=N;
    List=L;
  }
}

//                                                                    GROUP IDENTIFICATION
//                                                                    GROUP IDENTIFICATION
//                                                                    GROUP IDENTIFICATION

String TQp;
String MG;

void GroupIdentifier(Geometry G){
  String g = G.repCODE;
  int f =70;

  //COUNTING SEGMENT
  int Count;
  String[] S = splitTokens(g, "[]");
  Count = S.length-1;
  for(int i=0; i<S.length; i++){
    if(S[i].equals("/")){
      Count-=2;
    }
  }
  print(G.CAD + " #Segments = " + Count + " ");

  //Q/TP
  String[] Sarray = splitTokens(g,"[/]");
  IntDict Search = new IntDict();
  for(int i=0; i<Sarray.length; i++){
    Search.increment(Sarray[i]);
  }
  Search.sortValuesReverse();
  int[] counts = Search.valueArray();
  String[] SEARCH = Search.keyArray();

  if(counts[0] == 1){
    TQp = "0";
  }
  if(counts[0] > 1 && counts[1] == 1){
    TQp = "1";
  }
  if(counts[0] > 1 && counts[1] > 1 && counts[2] == 1){
    TQp = "2";
  }
  if(counts[0] > 1 && counts[1] > 1 && counts[2] > 1 && counts[3] == 1){
    TQp = "3";
  }
  if(counts[0] > 1 && counts[1] > 1 && counts[2] > 1 && counts[3] > 1 && counts[4] == 1){
    TQp = "4";
  }
  if(counts[0] > 1 && counts[1] > 1 && counts[2] > 1 && counts[3] > 1 && counts[4] > 1 &&
counts[5] == 1){
    TQp = "5";
  }
  if(counts[0] > 1 && counts[1] > 1 && counts[2] > 1 && counts[3] > 1 && counts[4] > 1
&& counts[5] > 1 && counts[6] == 1){
    TQp = "6";
  }
  if(counts[0] > 1 && counts[1] > 1 && counts[2] > 1 && counts[3] > 1 && counts[4] > 1
&& counts[5] > 1 && counts[6] > 1 && counts[7] == 1){
    TQp = "7";
  }
```

```
  if(counts[0] > 1 && counts[1] > 1 && counts[2] > 1 && counts[3] > 1 && counts[4] > 1
&& counts[5] > 1 && counts[6] > 1 && counts[7] > 1 && counts[8] == 1){
    TQp = "8";
  }
  if(counts[0] > 1 && counts[1] > 1 && counts[2] > 1 && counts[3] > 1 && counts[4] > 1
&& counts[5] > 1 && counts[6] > 1 && counts[7] > 1 && counts[8] > 1 && counts[9] == 1){
    TQp = "9";
  }
  print(" | #Points = " + counts.length);
  print(" | #T/QP = " + TQp);

  //POLYLINES
  int countPolylines = 1;
  String[] S1 = splitTokens(g, "[]");
  for(int i=0; i<S1.length; i++){
    if(S1[i].equals("/")){
      countPolylines++;
    }
  }
  print(" | #Polyline= " + (countPolylines));

  //GROUPS
  if(countPolylines==1){MG="A";}
  if(countPolylines==2){MG="B";}
  if(countPolylines==3){MG="C";}
  if(countPolylines==4){MG="D";}
  if(countPolylines==5){MG="E";}
  if(countPolylines==6){MG="F";}
  println(" | SMG: " + MG+TQp + " | MG:"+MG);
}

//                                                                    IDENTITY MATCH
//                                                                    IDENTITY MATCH
//                                                                    IDENTITY MATCH

// Identity Match function, takes two inputs:
// 1) An IGP to be examined;
// 2) Array of Geometry object (stores the historical Data).
void IdenticalMatch(Geometry g, Geometry[] G){
  // A loop through the array of Geometry object.
  for(int i=0; i<G.length; i++){
    // Compare the representational code of the input IGP with each geometry in the array
    if(g.repCODE.equals(G[i].repCODE) && g.CAD != G[i].CAD &&
g.SYMMETRY.equals(G[i].SYMMETRY)){
      // Only if a match exist, print the number of the design.
      print(G[i].CAD+",");
    }
  }
}

//                                                  LV0 STRUCTURAL EQUIVALENCY MATCH
//                                                  LV0 STRUCTURAL EQUIVALENCY MATCH
//                                                  LV0 STRUCTURAL EQUIVALENCY MATCH

// LV0 structural equivalency function, takes two inputs:
// 1) An IGP to be examined;
// 2) Array of Geometry object (stores the historical Data).
void LevelZeroSE(Geometry g, Geometry[] G){
  String[] S1 = splitTokens(g.repCODE,"[]"); // Convert First representational code into
array
  //Keep only the angle parameter in each array index in the first representational code
  for(int y=0; y<S1.length; y++){
    if(S1[y].equals("/") == false){
      String[] clean = splitTokens(S1[y], "-");
      S1[y] = clean[0];
    }
  }
  String comparison1 = join(S1," ");
  // A loop through the array of Geometry object.
```

```
  for(int i=0; i<G.length; i+=1){
    //Fillters to avoid including identical design in this category, design itself, or
comparing with different symmetry group
    if(g.ID.equals(G[i].ID) == false && CADno != G[i].CAD &&
g.SYMMETRY.equals(G[i].SYMMETRY))
    {
      String[] S2 = splitTokens(G[i].repCODE,"[]");
      //Keep only the angle parameter in each array index in the Second representational
code
      for(int y=0; y<S2.length; y++){
        if(S2[y].equals("/") == false){
          String[] clean = splitTokens(S2[y], "-");
          S2[y] = clean[0];
        }
      }
      String comparison2 = join(S2," ");
      //Conduct the comparison
      if(comparison1.equals(comparison2)){
        print(G[i].CAD + ",");
      }
    }
  }
}

//                                                        LV1 STRUCTURAL EQUIVALENCY MATCH
//                                                        LV1 STRUCTURAL EQUIVALENCY MATCH
//                                                        LV1 STRUCTURAL EQUIVALENCY MATCH

// LV1 structural equivalency function, takes two inputs:
// 1) An IGP to be examined;
// 2) Array of Geometry object (stores the historical Data).
void LevelOneSE(Geometry g, Geometry[] G){
  String[] S1 = splitTokens(g.repCODE,"[]"); // Convert First representational code into
array
  //Keep only the angle parameter in each array index in the first representational code
  for(int y=0; y<S1.length; y++){
    if(S1[y].equals("/") == false){
      String[] clean = splitTokens(S1[y], "-");
      S1[y] = clean[0];
      //if the angle is not equal to 30, 60, 90, 120, change the angle to "A"
      if(S1[y].equals("30") == false && S1[y].equals("60") == false && S1[y].equals("90")
== false && S1[y].equals("120") == false){
        S1[y] = "A";
      }
    }
  }
  String comparison1 = join(S1," ");
  // A loop through the array of Geometry object.
  for(int i=0; i<G.length; i+=1){
    if(g.ID.equals(G[i].ID) == false && CADno != G[i].CAD && g.Case.equals(G[i].Case) &&
g.SYMMETRY.equals(G[i].SYMMETRY))
    {
      String[] S2 = splitTokens(G[i].repCODE,"[]");
      //Keep only the angle parameter in each array index in the Second representational
code
      for(int y=0; y<S2.length; y++){
        if(S2[y].equals("/") == false){
          String[] clean = splitTokens(S2[y], "-");
          S2[y] = clean[0];
          //if the angle is not equal to 30, 60, 90, 120, change the angle to "A"
          if(S2[y].equals("30") == false && S2[y].equals("60") == false &&
S2[y].equals("90") == false && S2[y].equals("120") == false){
            S2[y] = "A";
          }
        }
      }
      String comparison2 = join(S2," ");
      //Conduct the comparison
      if(comparison1.equals(comparison2) && g.SYMMETRY.equals(G[i].SYMMETRY)){
```

```
        print(G[i].CAD + ", ");
      }
    }
  }
}

//                                                        LV2 STRUCTURAL EQUIVALENCY MATCH
//                                                        LV2 STRUCTURAL EQUIVALENCY MATCH
//                                                        LV2 STRUCTURAL EQUIVALENCY MATCH

// LV2 structural equivalency function, takes two inputs:
// 1) An IGP to be examined;
// 2) Array of Geometry object (stores the historical Data).
void LevelTwoSE(Geometry g, Geometry[] G){
  String[] S1 = splitTokens(g.repCODE,"[]"); // Convert First representational code into
array
  // Preserve only point sequence
  for(int y=0; y<S1.length; y++){
    if(S1[y].equals("/") == false){
      String[] clean = splitTokens(S1[y], "-");
      S1[y] = "P";
    }
  }
  String comparison1 = join(S1," ");
  // A loop through the array of Geometry object.
  for(int i=0; i<G.length; i+=1){
    if(g.ID.equals(G[i].ID) == false && CADno != G[i].CAD && g.Seg.equals(G[i].Seg) &&
g.Case.equals(G[i].Case) && g.SYMMETRY.equals(G[i].SYMMETRY))
    {
      String[] S2 = splitTokens(G[i].repCODE,"[]");// Convert second representational
code into array
      // Preserve only point sequence
      for(int y=0; y<S2.length; y++){
        if(S2[y].equals("/") == false){
          String[] clean = splitTokens(S2[y], "-");
          S2[y] = "P";
        }
      }
      String comparison2 = join(S2," ");
      //Conduct the comparison
      if(comparison1.equals(comparison2) && g.SYMMETRY.equals(G[i].SYMMETRY)){
        print(G[i].CAD + ",");
      }
    }
  }
}

//                                                        LV3 STRUCTURAL EQUIVALENCY MATCH
//                                                        LV3 STRUCTURAL EQUIVALENCY MATCH
//                                                        LV3 STRUCTURAL EQUIVALENCY MATCH

// LV3 structural equivalency function, takes two inputs:
// 1) An IGP to be examined;
// 2) Array of Geometry object (stores the historical Data).
void LevelThreeSE(Geometry g, Geometry[] G){
  // A loop through the array of Geometry object.
  for(int i=0; i<G.length; i+=1){
    if(g.ID.equals(G[i].ID) == false && CADno != G[i].CAD && g.Case.equals(G[i].Case) &&
g.SYMMETRY.equals(G[i].SYMMETRY)){
      //If the specific morphological group is the same, print the number of the geometry
      print(G[i].CAD + ",");
    }
  }
}

//                                                        LV4 STRUCTURAL EQUIVALENCY MATCH
//                                                        LV4 STRUCTURAL EQUIVALENCY MATCH
//                                                        LV4 STRUCTURAL EQUIVALENCY MATCH
```

```
// LV4 structural equivalency function, takes two inputs:
// 1) An IGP to be examined;
// 2) Array of Geometry object (stores the historical Data).
void LevelFourSE(Geometry g, Geometry[] G){
  // A loop through the array of Geometry object.
  for(int i=0; i<G.length; i+=1){
    if(g.ID.equals(G[i].ID) == false && CADno != G[i].CAD && g.MC.equals(G[i].MC) &&
g.SYMMETRY.equals(G[i].SYMMETRY)){
      //If the specific morphological group is the same, print the number of the geometry
      print(G[i].CAD + ",");
    }
  }
}

//                                                                                              A0 SORTING
//                                                                                              A0 SORTING
//                                                                                              A0 SORTING

String flipResults;

void flipCode(String S){
  String[] Sarray = splitTokens(S, "[]");
  String[] SarrayTemp = new String[Sarray.length];
  for(int i=0; i<Sarray.length; i++){
    if(Sarray[Sarray.length-1-i].equals("/") ==false){
      SarrayTemp[i] = "["+Sarray[Sarray.length-1-i]+"]";
    }
    if(Sarray[Sarray.length-1-i].equals("/") ==true){
      SarrayTemp[i] = Sarray[Sarray.length-1-i];
    }
  }
  flipResults = join(SarrayTemp,"");
}

//                                                                                              A1 SORTING
//                                                                                              A1 SORTING
//                                                                                              A1 SORTING

String partialA1flipResults;

void partialA1Flip(String S, Geometry g){
  String[] lookUpSharedPoint = splitTokens(S, "[]/");
  IntDict Search = new IntDict();
  for(int i=0; i<lookUpSharedPoint.length; i++){
    Search.increment(lookUpSharedPoint[i]);
  }
  Search.sortValuesReverse();
  String[] SEARCH = Search.keyArray();
  String delimiter = SEARCH[0];
  String[] divideCode = split(S, delimiter);
  flipCode(divideCode[1]);

partialA1flipResults=divideCode[0]+delimiter+"]"+flipResults+"["+delimiter+divideCode[2];
}

//                                                                                              B0 SORTING
//                                                                                              B0 SORTING
//                                                                                              B0 SORTING

void flipCaseB(String S, Geometry g, Geometry[] G, int identity, int LV0, int LV1, int
LV2){
  g.repCODE = S;
  if(identity==1){IdenticalMatch(g, G);} if(LV0==1){LevelZeroSE(g,
G);}if(LV1==1){LevelOneSE(g, G);} if(LV2==1){LevelTwoSE(g, G);}
  //FLIP ALL
  flipCode(S); g.repCODE=flipResults;
  if(identity==1){IdenticalMatch(g, G);} if(LV0==1){LevelZeroSE(g,
G);}if(LV1==1){LevelOneSE(g, G);} if(LV2==1){LevelTwoSE(g, G);}
  g.repCODE=Original;
```

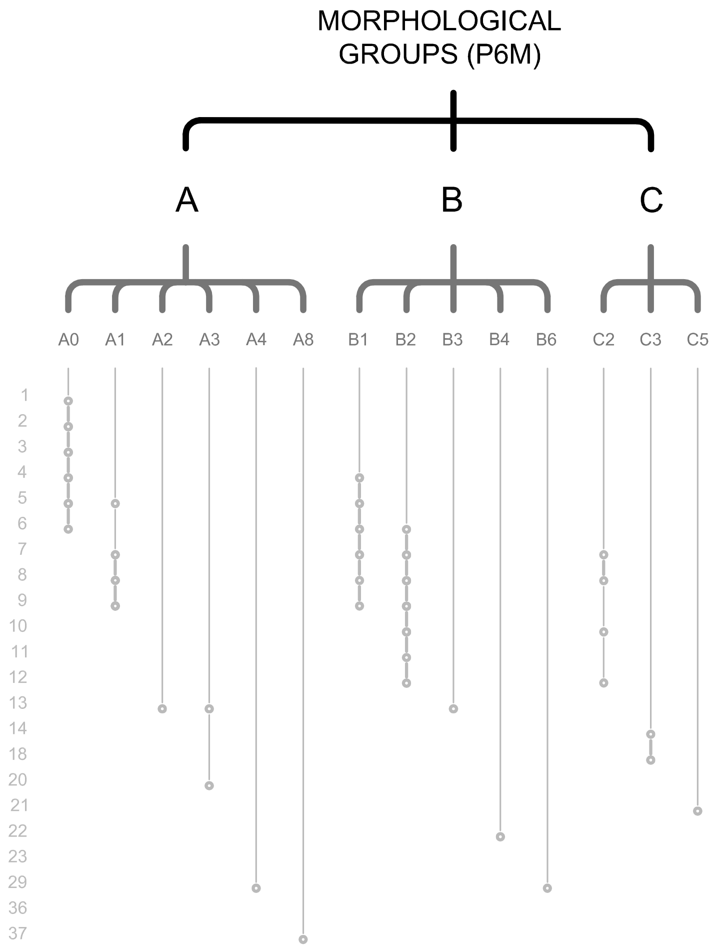
```
  //FLIP-KEEP
  flipFirst(S);g.repCODE=flipFirstResult;
  if(identity==1){IdenticalMatch(g, G);} if(LV0==1){LevelZeroSE(g,
G);}if(LV1==1){LevelOneSE(g, G);} if(LV2==1){LevelTwoSE(g, G);}

  //FLIP-KEEP REVERSED
  flipCode(g.repCODE); g.repCODE=flipResults;
  if(identity==1){IdenticalMatch(g, G);} if(LV0==1){LevelZeroSE(g,
G);}if(LV1==1){LevelOneSE(g, G);} if(LV2==1){LevelTwoSE(g, G);}
  g.repCODE=Original;

  //KEEP-FLIP REVERSED
  flipSecond(S);g.repCODE=flipSecondResult;
  if(identity==1){IdenticalMatch(g, G);} if(LV0==1){LevelZeroSE(g,
G);}if(LV1==1){LevelOneSE(g, G);} if(LV2==1){LevelTwoSE(g, G);}

  //KEEP-FLIP REVERSED
  flipCode(g.repCODE); g.repCODE=flipResults;
  if(identity==1){IdenticalMatch(g, G);} if(LV0==1){LevelZeroSE(g,
G);}if(LV1==1){LevelOneSE(g, G);} if(LV2==1){LevelTwoSE(g, G);}
  g.repCODE=Original;

  //FLIP-FLIP REVERSED
  flipboth(S);g.repCODE=flipbothResult;
  if(identity==1){IdenticalMatch(g, G);} if(LV0==1){LevelZeroSE(g,
G);}if(LV1==1){LevelOneSE(g, G);} if(LV2==1){LevelTwoSE(g, G);}


  //FLIP-FLIP REVERSED
  flipCode(g.repCODE); g.repCODE=flipResults;
  if(identity==1){IdenticalMatch(g, G);} if(LV0==1){LevelZeroSE(g,
G);}if(LV1==1){LevelOneSE(g, G);} if(LV2==1){LevelTwoSE(g, G);}
  g.repCODE=Original;
}

String flipFirstResult;
void flipFirst(String S){
  String[] Sarray = split(S, "/");
  String[] SarrayfirstPart = splitTokens(Sarray[0], "[]");
  String[] tempSarrayfirstPart = new String[SarrayfirstPart.length];
  for(int i=0; i<SarrayfirstPart.length; i++){
    tempSarrayfirstPart[tempSarrayfirstPart.length-1-i] = "[" + SarrayfirstPart[i] + "]";
  }
  flipFirstResult = join(tempSarrayfirstPart,"");
  flipFirstResult = flipFirstResult+"/"+Sarray[1];
}

String flipSecondResult;
void flipSecond(String S){
  String[] Sarray = split(S, "/");
  String[] SarraySecondPart = splitTokens(Sarray[1], "[]");
  String[] tempSarraySecondPart = new String[SarraySecondPart.length];
  for(int i=0; i<SarraySecondPart.length; i++){
    tempSarraySecondPart[tempSarraySecondPart.length-1-i] = "[" + SarraySecondPart[i] +
"]";
  }
  flipSecondResult = join(tempSarraySecondPart,"");
  flipSecondResult = Sarray[0] + "/" + flipSecondResult;
}

String flipbothResult;
void flipboth(String S){
  String[] Sarray1 = split(S, "/");
  String[] SarrayfirstPart = splitTokens(Sarray1[0], "[]");
  String[] tempSarrayfirstPart = new String[SarrayfirstPart.length];
  for(int i=0; i<SarrayfirstPart.length; i++){
    tempSarrayfirstPart[tempSarrayfirstPart.length-1-i] = "[" + SarrayfirstPart[i] + "]";
  }
```

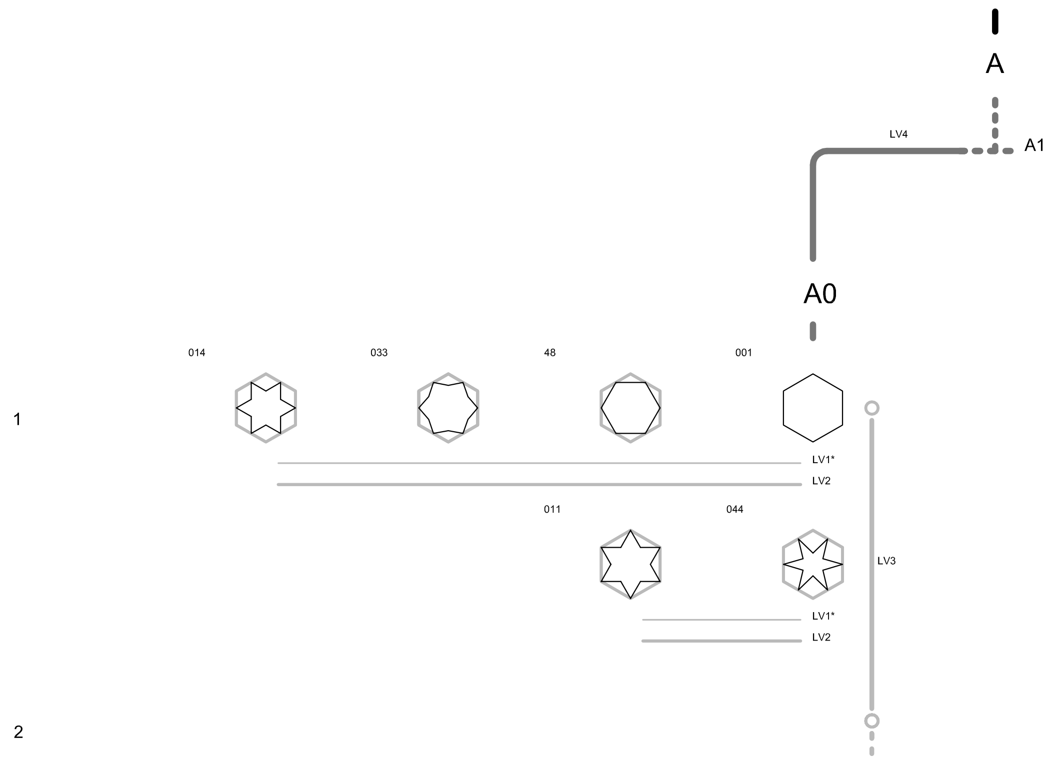
```
  flipFirstResult = join(tempSarrayfirstPart,"");
  String[] Sarray2 = split(S, "/");
  String[] SarraySecondPart = splitTokens(Sarray2[1], "[]");
  String[] tempSarraySecondPart = new String[SarraySecondPart.length];
  for(int i=0; i<SarraySecondPart.length; i++){
    tempSarraySecondPart[tempSarraySecondPart.length-1-i] = "[" + SarraySecondPart[i] +
"]";
  }
  flipSecondResult = join(tempSarraySecondPart,"");
  flipbothResult = flipFirstResult + "/" + flipSecondResult;
}

//                                                                                  B1 SORTING
//                                                                                  B1 SORTING
//                                                                                  B1 SORTING

String caseB1possibleCodeA;
String caseB1possibleCodeB;

void caseB1Flip(String S, Geometry g, String delimiter){
  String[] lookUpSharedPoint = splitTokens(S, "[]/");
  IntDict Search = new IntDict();
  for(int i=0; i<lookUpSharedPoint.length; i++){
    Search.increment(lookUpSharedPoint[i]);
  }
  Search.sortValuesReverse();
  int[] counts = Search.valueArray();
  String[] SEARCH = Search.keyArray();
  String[] S1 = splitTokens(S,"/");
  if(delimiter==""){
    delimiter = SEARCH[0];
  }
  String[] S1A = split(S1[0], delimiter);
  String[] S2A = split(S1[1], delimiter);
  caseB1possibleCodeA = S1A[0]+delimiter+S2A[1]+"/"+S2A[0]+delimiter+S1A[1];
  flipCode(S2A[0]); S2A[0]=flipResults;
  flipCode(S1A[1]); S1A[1]=flipResults;
  caseB1possibleCodeB = S1A[0]+delimiter+S2A[0]+"]/["+S1A[1]+delimiter+S2A[1];
  String[] Cleaning = splitTokens(caseB1possibleCodeB, "[]");
  for(int i=0; i<Cleaning.length; i++){
    if(Cleaning[i].equals("/")==false){
      Cleaning[i] = "["+Cleaning[i]+"]";
    }
  }
  caseB1possibleCodeB = join(Cleaning,"");
}

//                                                                                  B2 SORTING
//                                                                                  B2 SORTING
//                                                                                  B2 SORTING

String caseB2possibleCodeA;
String caseB2possibleCodeB;
String caseB2possibleCodeC;
String caseB2possibleCodeD;
String caseB2possibleCodeE;
String caseB2possibleCodeF;
String newB2code1;
String newB2code2;

void caseB2Flip(String S, Geometry g, String delimiter){
  String[] lookUpSharedPoint = splitTokens(S, "[]/");
  IntDict Search = new IntDict();
  for(int i=0; i<lookUpSharedPoint.length; i++){
    Search.increment(lookUpSharedPoint[i]);
  }
  Search.sortValuesReverse();
  int[] counts = Search.valueArray();
  String[] SEARCH = Search.keyArray();
```

```
String delimiter1 = SEARCH[0];
String delimiter2 = SEARCH[1];
int c1 =0, c2 =0;
String[] B2search = split(S, "/");
String[] firstSide = splitTokens(B2search[0],"[]");
for(int p=0; p<firstSide.length; p++){
  if(delimiter1.equals(firstSide[p])){
    c1++;
  }
  if(delimiter2.equals(firstSide[p])){
    c2++;
  }
}
if(c1==1 && c2==2){
 String[] B2split = split(S, "/");
 partialA1Flip(B2split[0], g);
 B2split[0] = partialA1flipResults;
 newB2code1 = B2split[0]+"/"+B2split[1]; /// Main CODE 2
 caseB1Flip(S, g, delimiter1);
 caseB2possibleCodeA = caseB1possibleCodeA; /// Main CODE 3
 caseB2possibleCodeB = caseB1possibleCodeB; /// Main CODE 4
 if(g.CAD != 234){
   caseB1Flip(caseB2possibleCodeA, g, delimiter2);
   caseB2possibleCodeC = caseB1possibleCodeA;
   caseB2possibleCodeD = caseB1possibleCodeB;
   caseB1Flip(caseB2possibleCodeB, g, delimiter2);
   caseB2possibleCodeE = caseB1possibleCodeA;
   caseB2possibleCodeF = caseB1possibleCodeB;
   B2split = split(caseB1possibleCodeB,"/");
   partialA1Flip(B2split[1], g);
   B2split[1] = partialA1flipResults;
   newB2code2 = B2split[0]+"/"+B2split[1]; /// Main CODE 2
 }
}
if(c1==0 && c2==1){
 String[] B2split = split(S, "/");
 partialA1Flip(B2split[1], g);
 B2split[1] = partialA1flipResults;
 newB2code1 = B2split[0]+"/"+B2split[1]; /// Main CODE 2
 caseB1Flip(S, g, delimiter2);
 caseB2possibleCodeA = caseB1possibleCodeA; /// Main CODE 3
 caseB2possibleCodeB = caseB1possibleCodeB; /// Main CODE 4
 caseB1Flip(caseB2possibleCodeA, g, delimiter1);
 caseB2possibleCodeC = caseB1possibleCodeA;
 caseB2possibleCodeD = caseB1possibleCodeB;
 caseB1Flip(caseB2possibleCodeB, g, delimiter1);
 caseB2possibleCodeE = caseB1possibleCodeA;
 caseB2possibleCodeF = caseB1possibleCodeB;
 B2split = split(caseB1possibleCodeB,"/");
 partialA1Flip(B2split[0], g);
 B2split[0] = partialA1flipResults;
 newB2code2 = B2split[0]+"/"+B2split[1]; /// Main CODE 2
}
if(c1==1 && c2==0){
 String[] B2split = split(S, "/");
 partialA1Flip(B2split[1], g);
 B2split[1] = partialA1flipResults;
 newB2code1 = B2split[0]+"/"+B2split[1]; /// Main CODE 2
 caseB1Flip(S, g, delimiter1);
 caseB2possibleCodeA = caseB1possibleCodeA; /// Main CODE 3
 caseB2possibleCodeB = caseB1possibleCodeB; /// Main CODE 4
 caseB1Flip(newB2code1, g, delimiter1);
 caseB2possibleCodeC = caseB1possibleCodeA;
 caseB2possibleCodeD = caseB1possibleCodeB;
 B2split = split(caseB1possibleCodeA,"/");
 partialA1Flip(B2split[0], g);
 B2split[0] = partialA1flipResults;
 newB2code2 = B2split[0]+"/"+B2split[1]; /// Main CODE 2
}}
```

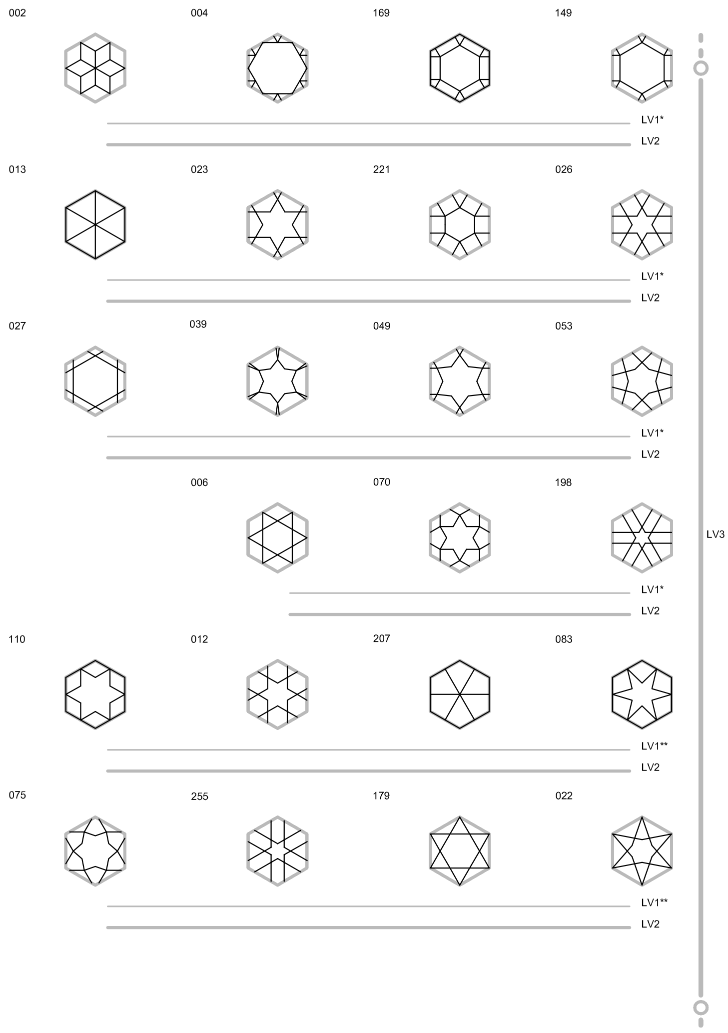
Appendix D

Morphological Correlations Flowcharts

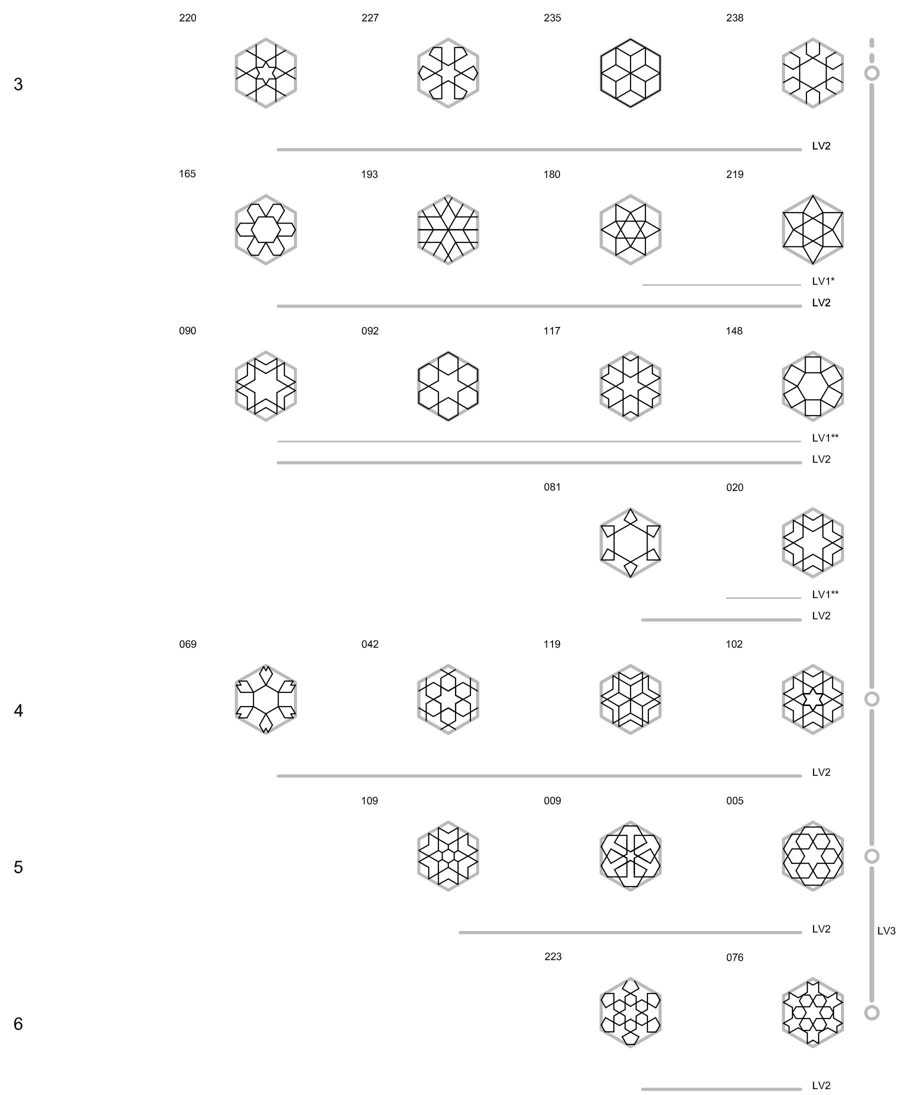

MORPHOLOGICAL
GROUPS (P6M)
A
B
C
A0
A1
A2
A3
A4
A8
B1
B2
B3
B4
B6
C2
C3
C5
1
2
3
4
5
6
7
8
9
10
11
12
13
14
18
20
21
22
23
29
36
37

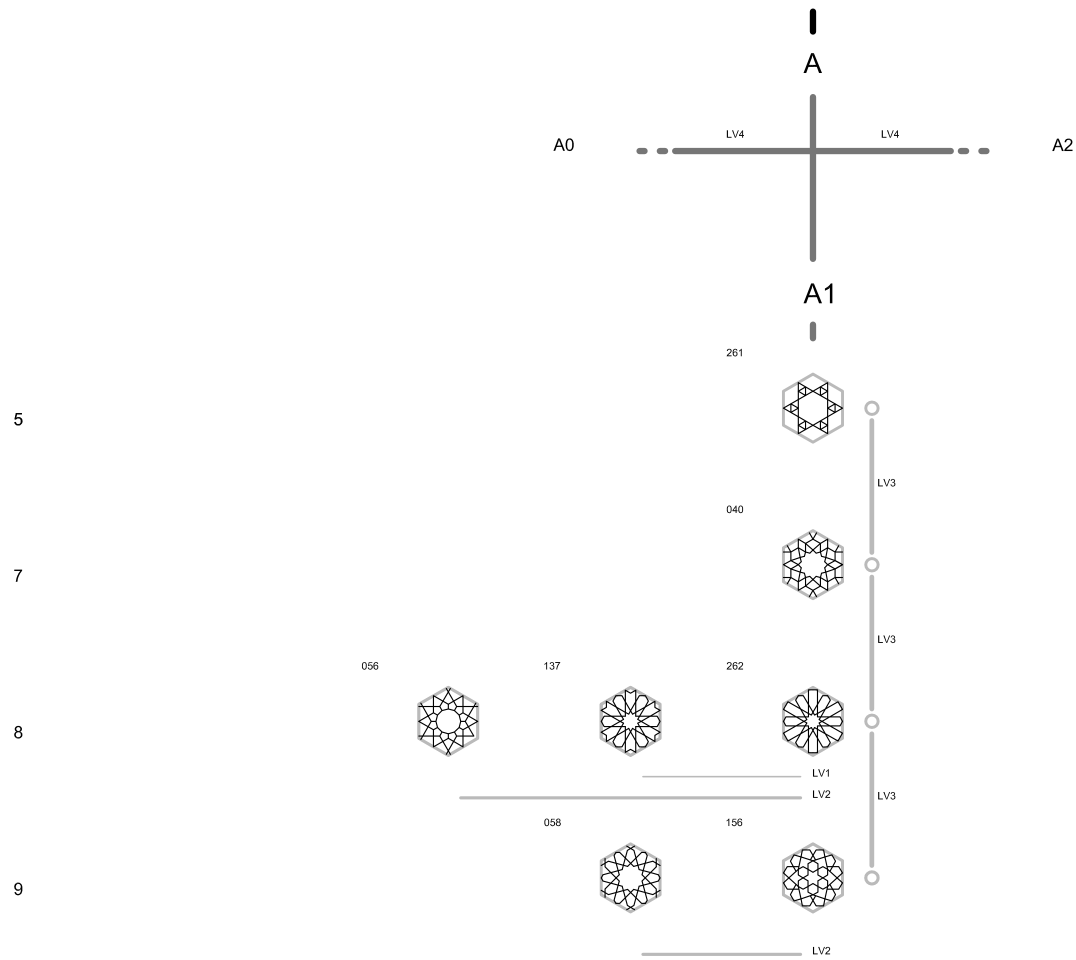

A
LV4
A1
A0
014
033
48
001
1
LV1*
LV2
011
044
LV3
LV1*
LV2
2

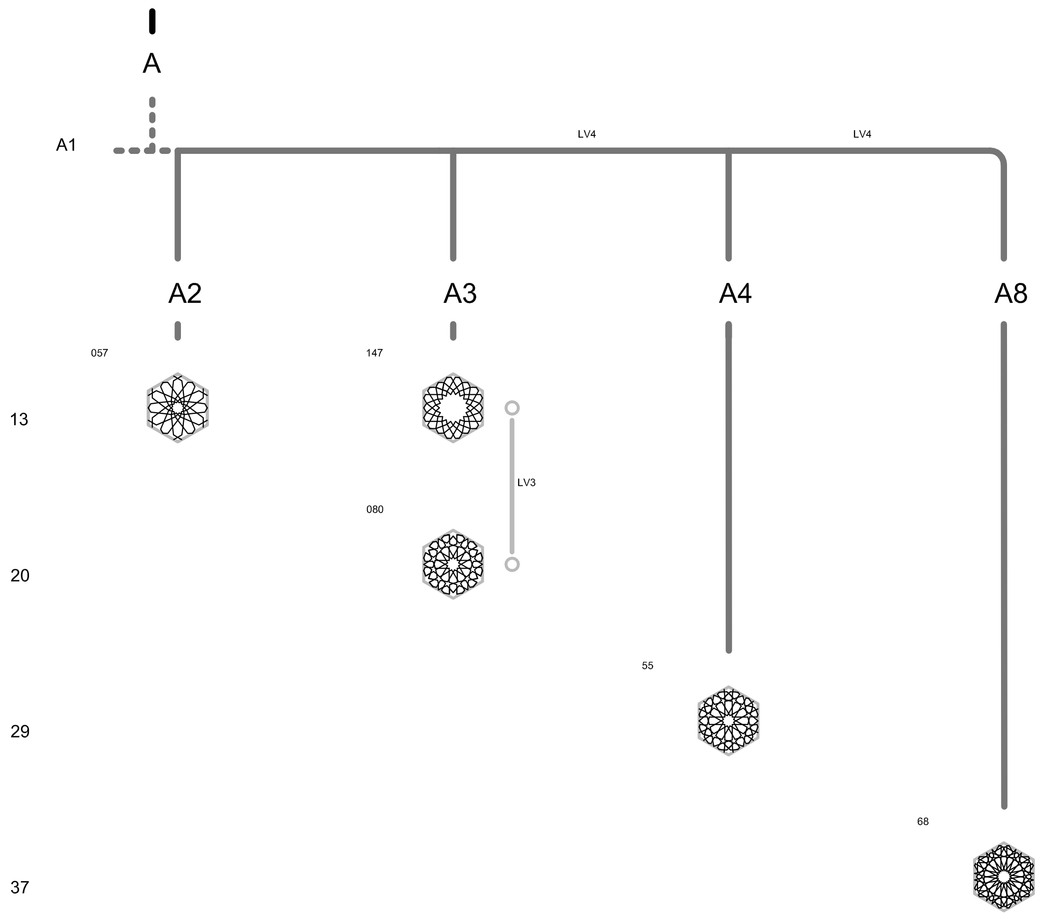
2

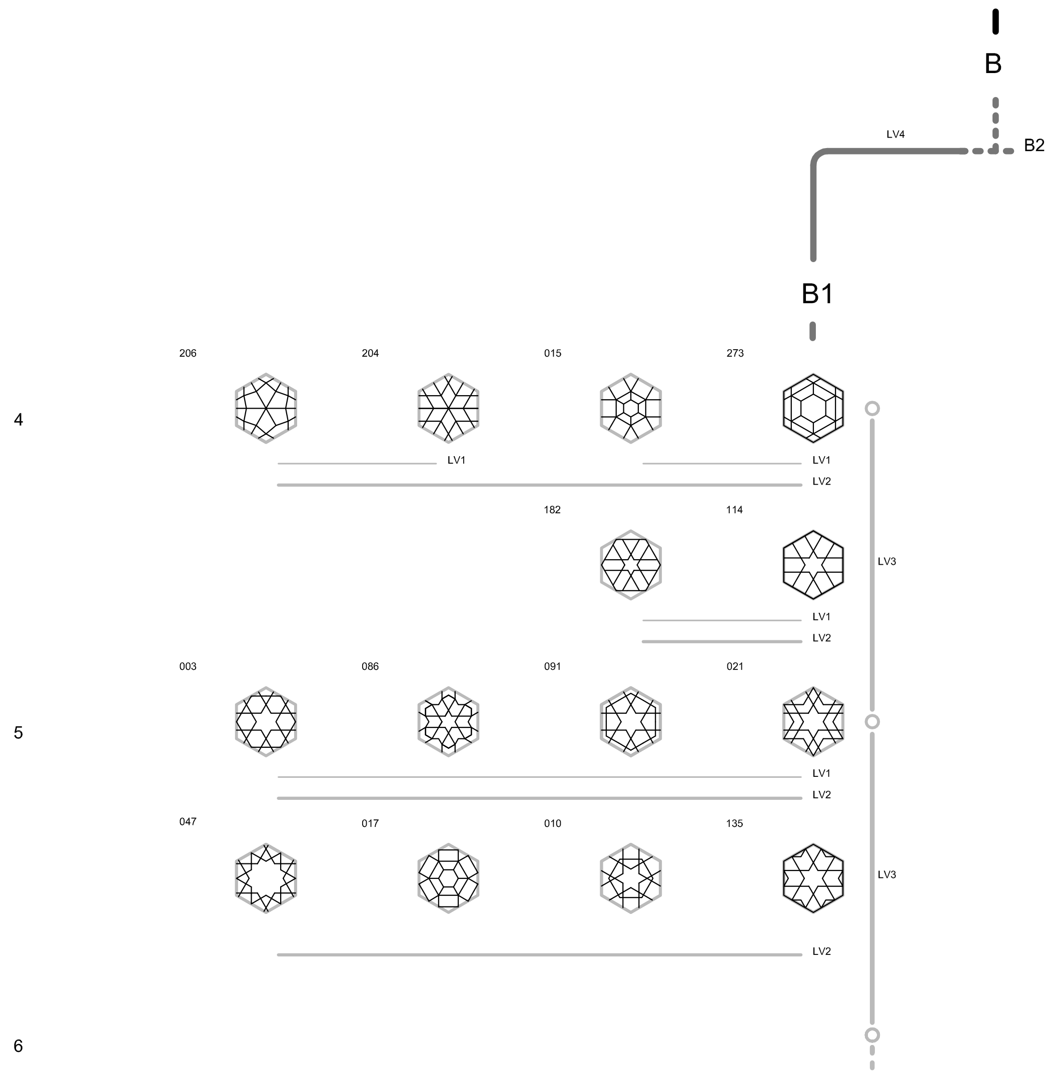

002
004
169
149
LV1*
LV2
013
023
221
026
LV1*
LV2
027
039
049
053
LV1*
LV2
006
070
198
LV1*
LV2
LV3
110
012
207
083
LV1**
LV2
075
255
179
022
LV1**
LV2

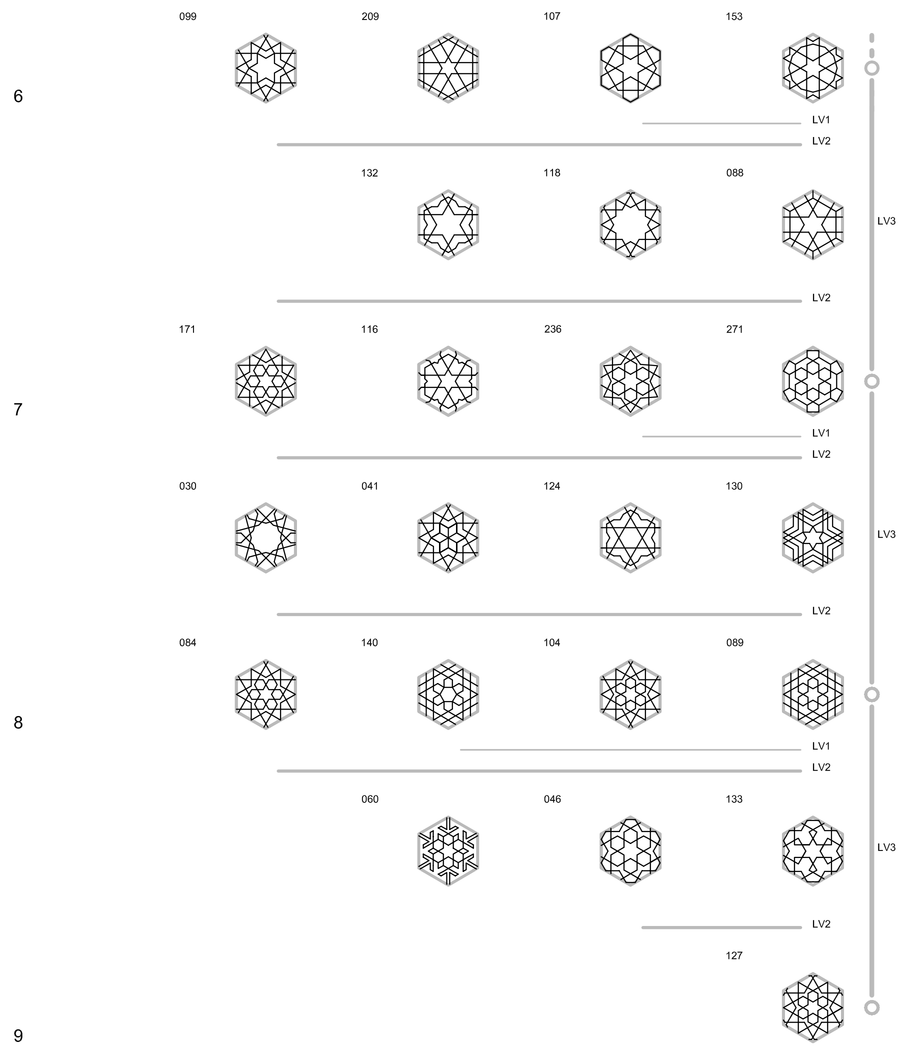
3


220
227
235
238
3
LV2
165
193
180
219
LV1*
LV2
090
092
117
148
LV1**
LV2
081
020
LV1**
LV2
069
042
119
102
4
LV2
109
009
005
5
LV2
LV3
223
076
6
LV2

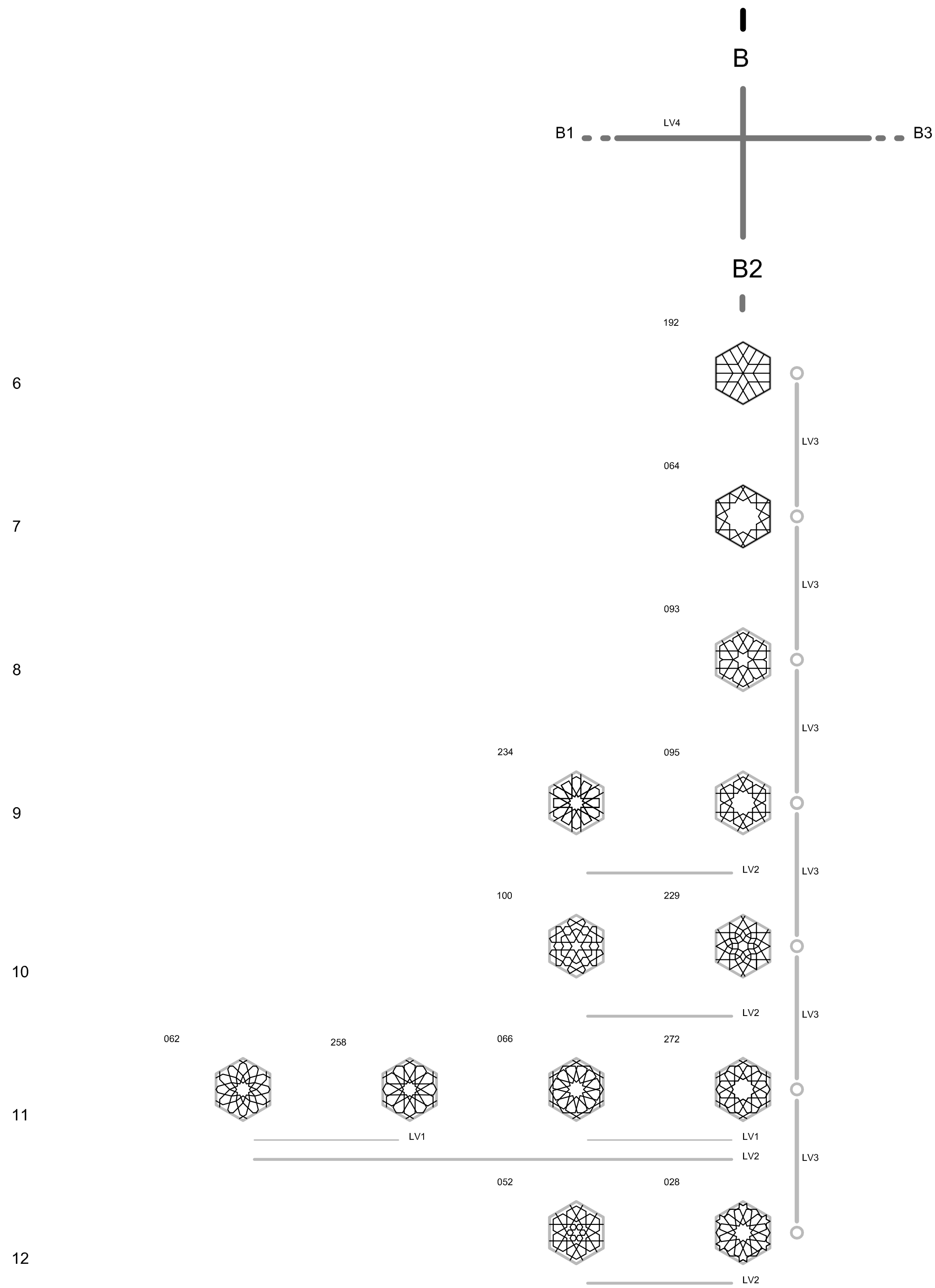

A
A0
LV4
LV4
A2
A1
261
5
LV3
040
7
LV3
056
137
262
8
LV1
LV2
LV3
058
156
9
LV2

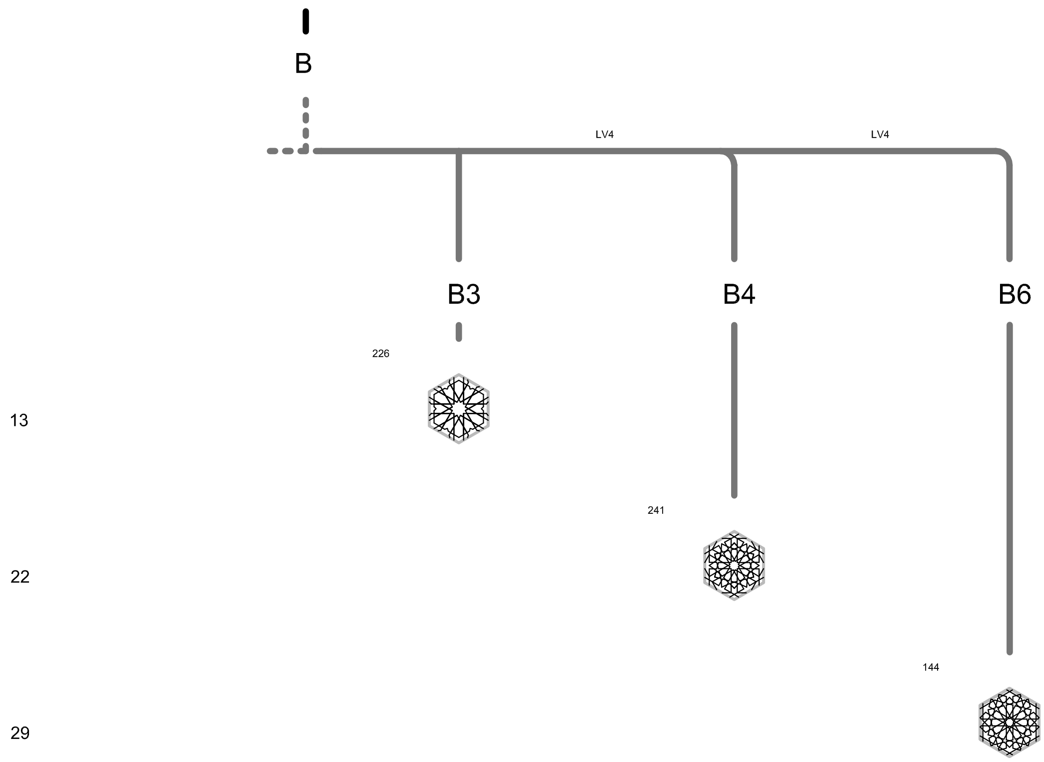

A
A1
LV4
LV4
A2
A3
A4
A8
057
147
13
LV3
080
20
55
29
68
37

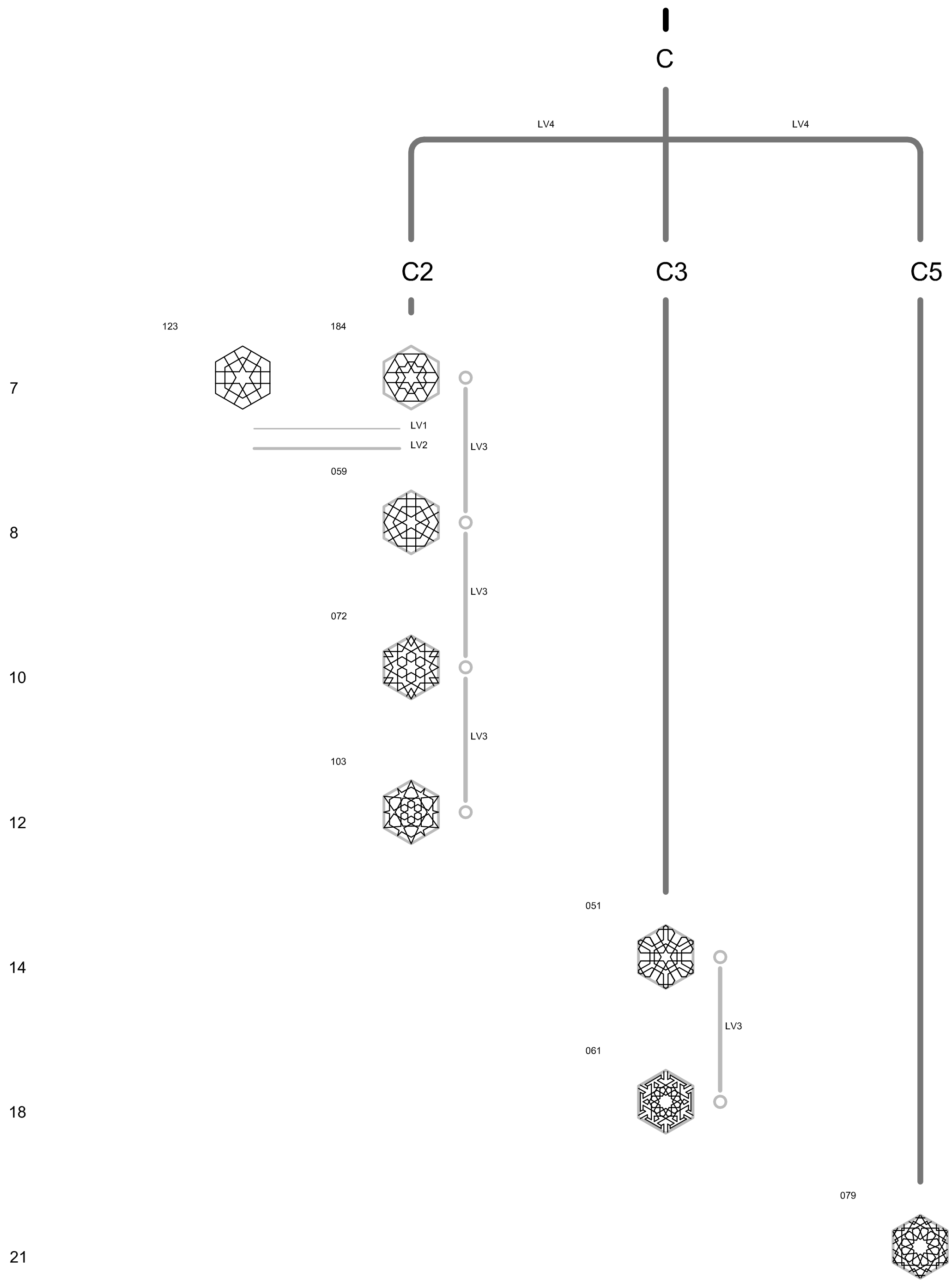

B
LV4
B2
B1
206
204
015
273
4
LV1
LV1
LV2
182
114
LV3
LV1
LV2
003
086
091
021
5
LV1
LV2
047
017
010
135
LV3
LV2
6

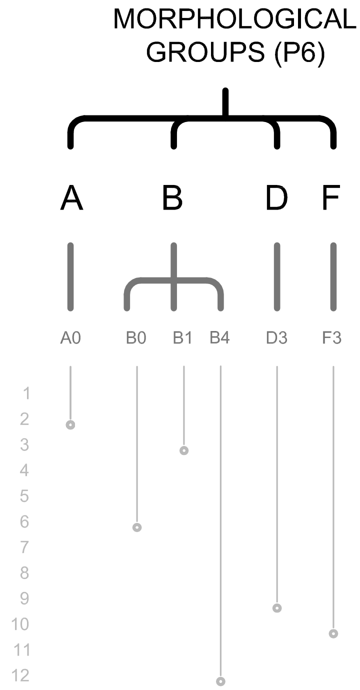


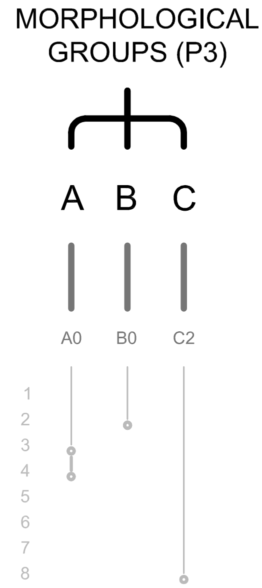
6
099
209
107
153
LV1
LV2
132
118
088
LV3
LV2
7
171
116
236
271
LV1
LV2
030
041
124
130
LV3
LV2
8
084
140
104
089
LV1
LV2
060
046
133
LV3
LV2
9
127

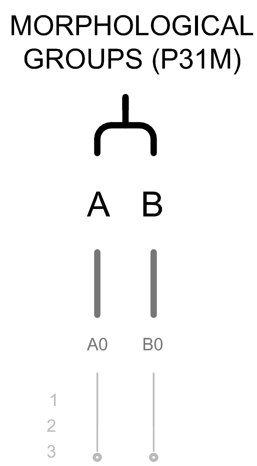

B
B1
LV4
B3
B2
192
6
LV3
064
7
LV3
093
8
LV3
234
095
9
LV2
LV3
100
229
10
LV2
LV3
062
258
066
272
11
LV1
LV1
LV2
LV3
052
028
12
LV2

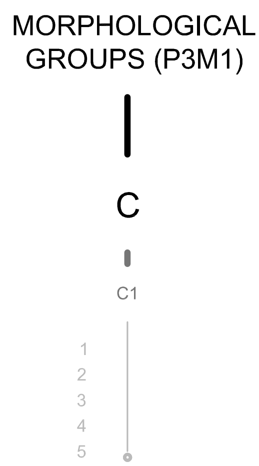

B
LV4
LV4
B3
B4
B6
226
241
144
13
22
29

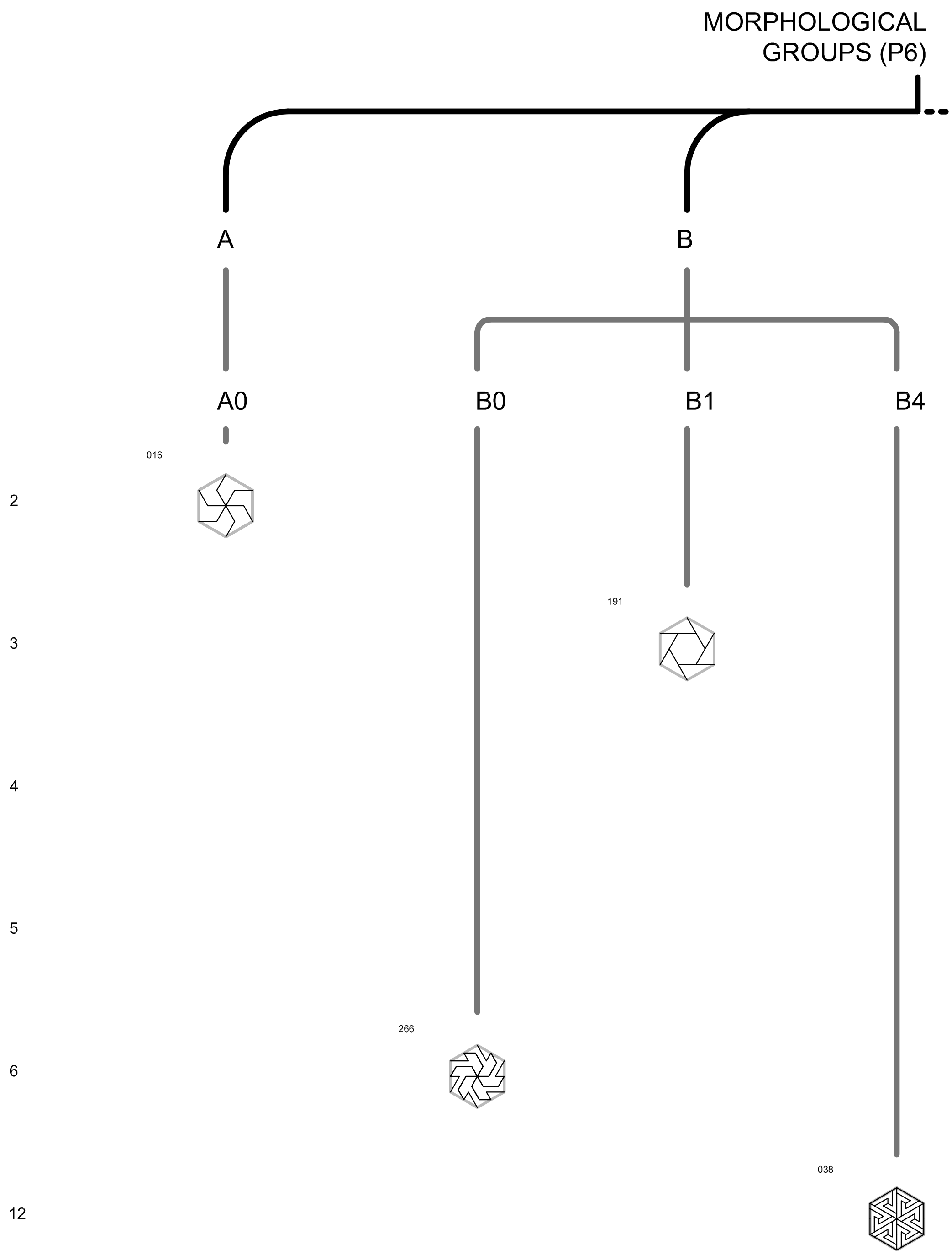

C
LV4
LV4
C2
C3
C5
123
184
7
LV1
LV2
LV3
059
8
LV3
072
10
LV3
103
12
051
14
LV3
061
18
079
21

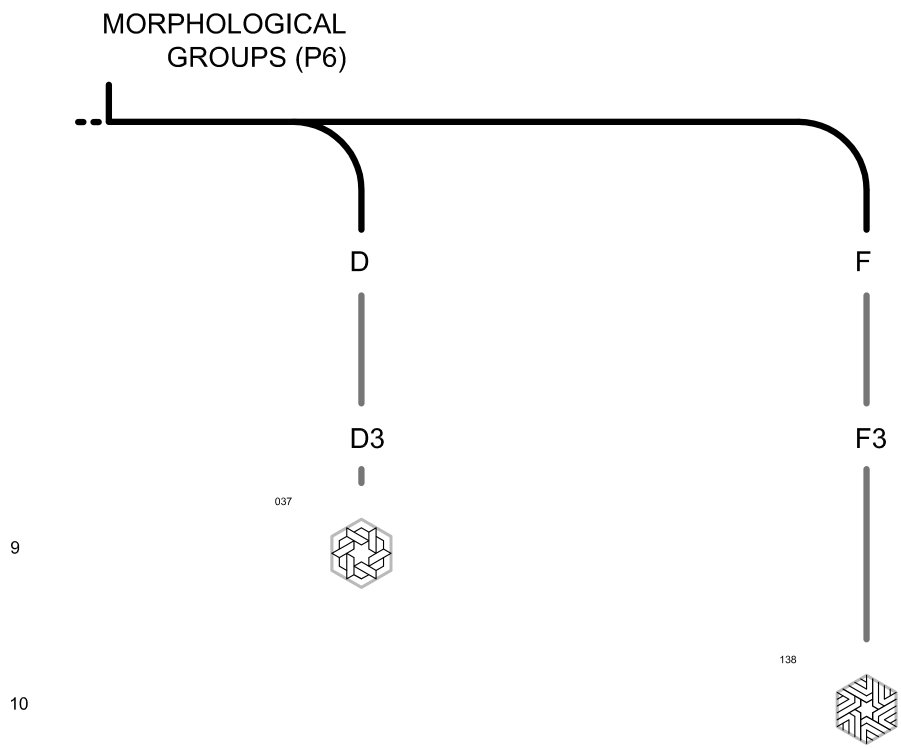

MORPHOLOGICAL
GROUPS (P6)
A
B
D
F
A0
B0
B1
B4
D3
F3
1
2
3
4
5
6
7
8
9
10
11
12


MORPHOLOGICAL
GROUPS (P3)
A
B
C
A0
B0
C2
1
2
3
4
5
6
7
8

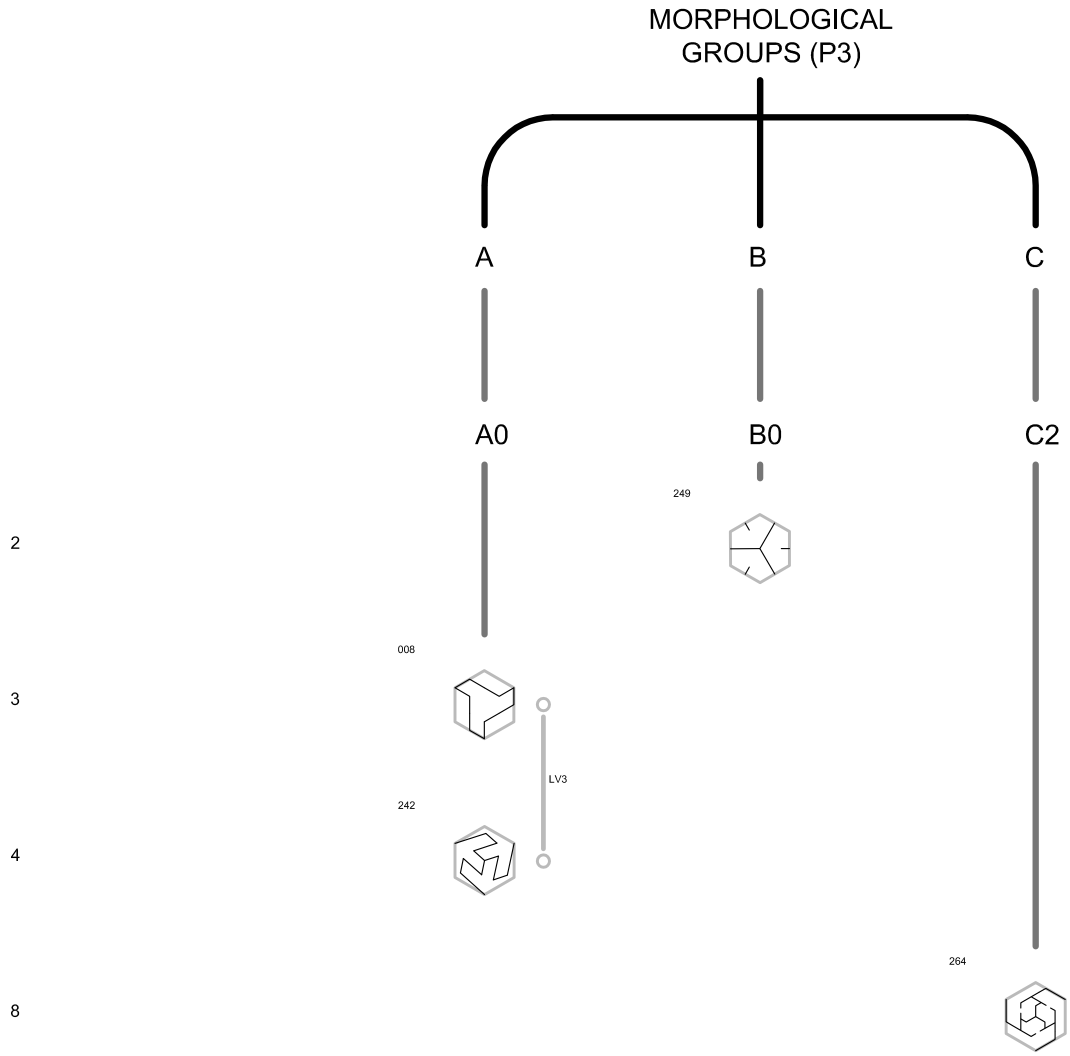

MORPHOLOGICAL
GROUPS (P31M)
A
B
A0
B0
1
2
3

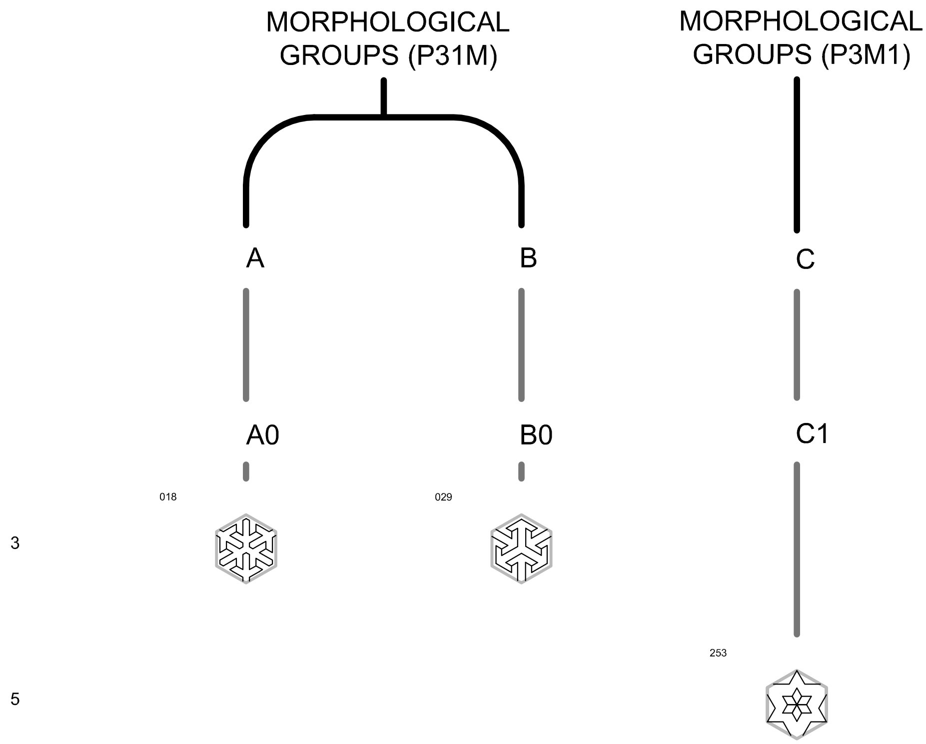

MORPHOLOGICAL
GROUPS (P3M1)
C
C1
1
2
3
4
5


MORPHOLOGICAL
GROUPS (P6)
A
B
A0
B0
B1
B4
016
2
191
3
4
5
266
6
038
12



MORPHOLOGICAL
GROUPS (P6)
D
F
D3
F3
037
9
138
10


MORPHOLOGICAL
GROUPS (P3)
A
B
C
A0
B0
C2
249
2
008
3
LV3
242
4
264
8


MORPHOLOGICAL GROUPS (P31M)
MORPHOLOGICAL GROUPS (P3M1)
A
B
C
A0
B0
C1
018
029
253
3
5

REFERENCES

Abas, S.J. & Salman, A.S. 1995, *Symmetries of Islamic geometrical patterns,* World Scientific.

Al Ajlouni, R.A. 2012, "The global long-range order of quasi-periodic patterns in Islamic architecture", vol. 68, no. 2, pp. 235-243.

al-Būzjānī, A.a. 998, *Kitāb fī mā yaḥtāj ilayh al-ṣāniʿ min al-aʿmāl al-handasiyya (A book on those geometric constructions which are necessary for a craftsman),* , Baghdad.

Aljamali, A. & Banissi, E. 2003, "Normalization and exploration design method of Islamic geometric patterns", *Geometric Modeling and Graphics, 2003. Proceedings. 2003 International Conference on*IEEE, , pp. 42.

Al-Khalili, J. 2011, *The house of wisdom: How Arabic science saved ancient knowledge and gave us the Renaissance,* Penguin.

Al-Khalili, J. 2010, *Science and Islam, The Language of Science.*

Bourgoin, J. 2012, *Arabic geometrical pattern and design,* Courier Corporation.

Broug, E. 2013a, *Internation Workshop on Geometric Patterns in Islamic Art*, Turkey.

Broug, E. 2013b, *Islamic Geometric Design,* Thames & Hudson; 1 edition.

Broug, E. 2008, *Islamic geometric patterns,* Thames & Hudson London.

Burckhardt, T. 2009, *Art of Islam: Language and meaning,* World Wisdom, Inc.

Chorbachi, W.K. 1989, "In the Tower of Babel: Beyond symmetry in islamic design", vol. 17, no. 4, pp. 751-789.

Creswell, K.A.C. 1919, "Some Newly Discovered Ṭûlûnide Ornament", *The Burlington Magazine for Connoisseurs,* , pp. 180-188.

Deleuze, G. 1994, *Difference and repetition,* Columbia University Press.

Deleuze, G. 1993, *The fold: Leibniz and the Baroque,* U of Minnesota Press.

Deleuze, G. & Guattari, F. 1988, *A thousand plateaus: Capitalism and schizophrenia,* Bloomsbury Publishing.

Djibril, M.O. & Thami, R.O.H. 2008, "Islamic geometrical patterns indexing and classification using discrete symmetry groups", *Journal on Computing and Cultural Heritage (JOCCH),* vol. 1, no. 2, pp. 10.

Dürer, A. 1528, *Hierinn sind begriffen vier Bücher von menschlicher Proportion,* Formschnyder.

El-Said, I., El-Bouri, T. & Critchlow, K. 1993a, *Islamic art and architecture: the system of geometric design,* Garnet Pub Ltd.

El-Said, I., El-Bouri, T. & Critchlow, K. 1993b, *Islamic art and architecture: the system of geometric design,* Garnet Pub.

El-Said, I. & Parman, A. 1976, *Geometric concepts in Islamic art,* Dale Seymour Pubn.

Greg Lynn 1993, *Folding in architecture,* Academy Editions Limited.

Groat, L. & Wang, D. 2002, *Architectural research methods,* Springer.

Grünbaum, B., Grünbaum, Z. & Shepard, G. 1986a, "Symmetry in Moorish and other ornaments", *Computers & Mathematics with Applications,* vol. 12, no. 3, pp. 641-653.

Grünbaum, B., Grünbaum, Z. & Shepard, G. 1986b, "Symmetry in Moorish and other ornaments", *Computers & Mathematics with Applications,* vol. 12, no. 3, pp. 641-653.

Herzfeld, E. 1942, "Damascus: Studies in Architecture: I", *Ars islamica,* vol. 9, pp. 1-53.

Hill, D. & Grabar, O. 1967, *Islamic Architecture and Its Decoration AD 800-1500: A Photographic Survey,* Faber.

Hillier, B. & Hanson, J. 1989, *The social logic of space,* Cambridge university press.

Izadi, A., Rezaei, M. & Bastanfard, A. 2010, "A Computerized Method to Generate Complex Symmetric and Geometric Tiling Patterns" in *Intelligent Computer Graphics 2010* Springer, , pp. 185-210.

Jones, O. 1868, *The grammar of ornament,* B. Quaritch.

Kaplan, C.S. & Salesin, D.H. 2004, "Islamic star patterns in absolute geometry", vol. 23, no. 2, pp. 97-119.

Keshani, H. 2012, "Towards digital Islamic art history", *Journal of Art Historiography Number,* .

Kolarevic, B. & Parlac, V. 2015, *Building dynamics: exploring architecture of change,* Routledge.

Kolarevic, B. 2004, *Architecture in the Digital Age: Design and Manufacturing,* Taylor & Francis.

Lalvani, H. 1989, "Coding and generating complex periodic patterns", vol. 5, no. 4, pp. 180-202.

Loeb, A. 2012, *Concepts & images: Visual mathematics,* Springer Science & Business Media.

Loeb, A.L. 1978, *Color and symmetry,* Krieger Pub Co.

Marans, R.W. & Stokols, D. 2013, *Environmental simulation: Research and policy issues,* Springer Science & Business Media.

March, L. & Steadman, P. 1974, "Geometry of Environment: an Introduction to Spatial Organisation in Design."

Müller, E. 1944, *Gruppentheoretische und strukturanalytische Untersuchungen der maurischen Ornamente aus der Alhambra in Granada,* Buchdruckerei Baublatt ag.

Necipoğlu, G. 1992, *Geometric Design in Timurid/Turkmen Architectural Practice: Thoughts on Recently Discovered Scroll and Its Late Gothic Parallel.*

Necipoğlu, G. & Al-Asad, M. 1995, *The Topkapı Scroll: Geometry and Ornament in Islamic Architecture: Topkapı Palace Museum Library MS,* Getty Center for the History of Art and the Humanities.

Oxman, R. & Oxman, R. 2014, *Theories of the Digital in Architecture,* Routledge.

Özdural, A. 2000, "Mathematics and arts: Connections between theory and practice in the medieval Islamic world", vol. 27, no. 2, pp. 171-201.

Özdural, A. 1995, "Omar Khayyam, mathematicians, and" conversazioni" with artisans", *Journal of the Society of Architectural Historians,* vol. 54, no. 1, pp. 54-71.

Pérez-Gómez, R. 1987, "The four regular mosaics missing in the Alhambra", *Computers & Mathematics with Applications,* vol. 14, no. 2, pp. 133-137.

Rabbat, N. 2012, ***Re-Branding Islamic Architecture***, Indianapolis Museum of Art.

Rabbat, N. 2004, "Islamic Architecture as a Field of Historical Enquiry", *Architectural design,* , no. 6, pp. 18-23.

Riether, G. & Baerlecken, D. 2012, "Digital Girih, a digital interpretation of Islamic architecture", *International Journal of Architectural Computing,* vol. 10, no. 1, pp. 1-11.

Simon, H.A. 1996, *The sciences of the artificial,* MIT press.

Singleton Jr, R.A., Straits, B.C. & Straits, M.M. 1993, *Approaches to social research,* Oxford University Press.

Steadman, P. 1983, *Architectural morphology: An introduction to the geometry of building plans,* Taylor & Francis.

Stevenson, A. 2010, *Oxford dictionary of English,* Oxford University Press, USA.

Stronach, D. & Young, T.C. 1966, "Three Seljuq tomb towers", *Iran,* vol. 4, pp. 1-20.

Terzidis, K. 2006, *Algorithmic architecture,* Routledge.

Thompson, D.W. 1917, "On growth and form."

Wade, D. 2015, *Pattern in Islamic Art*.

Wilson, E. 1988, *Islamic designs,* British Museum Publications.

Woodbury, R. 2010, "Elements of parametric design."