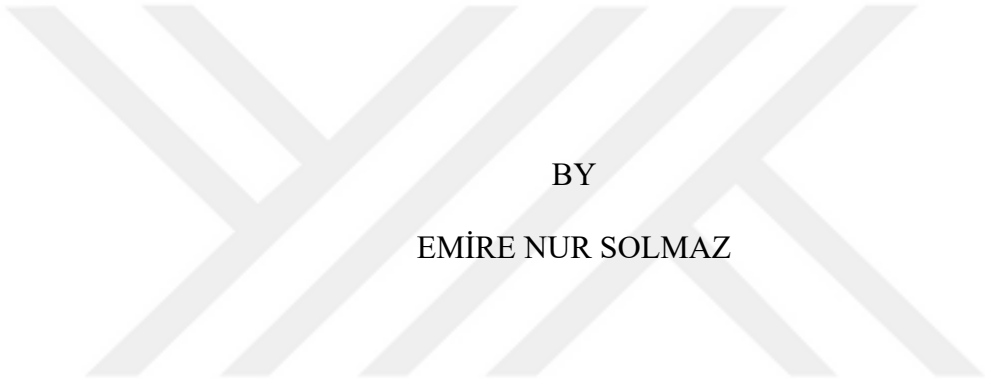


A MACHINE LEARNING AND ROBOTIC SIMULATION FRAMEWORK FOR
THE GENERATION AND ASSEMBLY OF STRUCTURES USING DISCRETE
IRREGULAR ELEMENTS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY



BY
EMİRE NUR SOLMAZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
BUILDING SCIENCE IN ARCHITECTURE

JUNE 2025

Approval of the thesis:

**A MACHINE LEARNING AND ROBOTIC SIMULATION FRAMEWORK
FOR THE GENERATION AND ASSEMBLY OF STRUCTURES USING
DISCRETE IRREGULAR ELEMENTS**

submitted by **EMİRE NUR SOLMAZ** in partial fulfillment of the requirements for
the degree of **Master of Science in Building Science in Architecture, Middle East
Technical University** by,

Prof. Dr. Naci Emre Altun
Dean, **Graduate School of Natural and Applied Sciences**

Assoc. Prof. Dr. Ayşem Berrin Çakmaklı
Head of the Department, **Architecture**

Prof. Dr. Arzu Gönenç Sorguç
Supervisor, **Architecture, METU**

Examining Committee Members:

Assoc. Prof. Dr. Mehmet Koray Pekerli
Architecture, METU

Prof. Dr. Arzu Gönenç Sorguç
Architecture, METU

Assoc. Prof. Dr. Ayşem Berrin Çakmaklı
Architecture, METU

Assist. Prof. Dr. Işıl Ruhi Sipahioğlu
Architecture, TOBB ETÜ

Assoc. Prof. Dr. İlker Erkan
Architecture, Süleyman Demirel Üniversitesi

Date: 18.06.2025



I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Last name : Solmaz, Emire Nur

Signature :

ABSTRACT

A MACHINE LEARNING AND ROBOTIC SIMULATION FRAMEWORK FOR THE GENERATION AND ASSEMBLY OF STRUCTURES USING DISCRETE IRREGULAR ELEMENTS

Solmaz, Emire Nur
Master of Science, Building Science in Architecture
Supervisor: Prof. Dr. Arzu Gönenç Sorguç

June 2025, 126 pages

Architectural structures are built by different techniques including masonry which is associated with modularity. The main idea of masonry is composing discrete units with various options that could be both regular/conventional and irregular/non-conventional to build a structure. In terms of irregularity, existing materials like salvage materials can be suitable for innovative configurations that could be challenging due to the precision problem. This research intends to contribute to the literature that addresses this challenge by adopting a stereotomic approach which is a long-established technique for combining discrete units. Machine learning (ML) and deep learning (DL) methods namely, decision tree, K-means clustering, and convolutional neural networks (CNNs) are used to advance the process of creating the intended configurations, and robotics is used to assemble them in a simulation environment. Masonry wall images are selected to comprise the dataset since they could reflect the properties of both stereotomy and stacking. After gathering and eliminating the images based on quality, a total of 393 images were used for the main dataset. A data preprocessing focusing on an edge detection algorithm is applied to

extract the features of the pattern that are used to train the decision tree, K-means, and CNN models. A pipeline consisting of these models, mainly the decision tree model is proposed to detect possible base patterns for the final configurations that are created with a placement algorithm. It is shown in this research that the proposed pipeline based on a series of geometric operations, predictions, and a placement algorithm regarding stereotomic principles gives compelling configurations for a set of discrete pieces that can be further assembled by robot simulation.

Keywords: Discrete Elements, Stereotomy, Machine Learning, Placement Configuration, Robotics



ÖZ

AYRIK DÜZENSİZ ELEMANLARLA YAPILARIN OLUŞTURULMASI VE MONTAJI İÇİN MAKİNE ÖĞRENMESİ VE ROBOT SİMÜLASYONU TABANLI BİR ÇERÇEVE

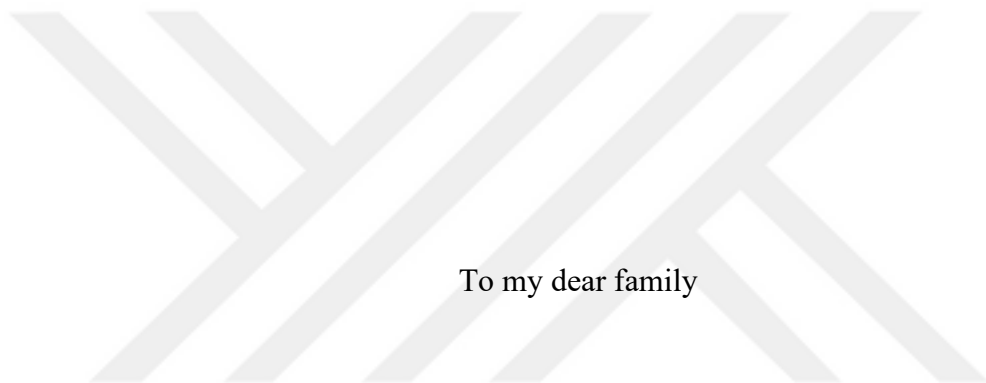
Solmaz, Emire Nur
Yüksek Lisans, Yapı Bilimleri, Mimarlık
Tez Yöneticisi: Prof. Dr. Arzu Gönenç Sorguç

Haziran 2025, 126 sayfa

Mimari yapılar, modülerlikle ilişkilendirilen yığma yapılar da dahil olmak üzere farklı tekniklerle inşa edilebilmektedir. Yığma yapıların ana mantığı, düzenli/konvansiyonel ve düzensiz/alışılmadık olabilen çeşitli seçeneklere sahip ayrı birimleri bir araya getirmektir. Düzensizlik açısından değerlendirmek gerekirse, kurtarılan malzemeler gibi mevcut bulunan malzemeler, hassasiyet konusunda zorlayıcı olabilecek yaratıcı konfigürasyonlar için uygun olabilmektedir. Bu araştırma, ayrı birimleri birleştirmek için uzun zamandır bilinen bir teknik olan stereotomik bir yaklaşımı benimseyerek bahsedilen zorluğu ele alan literatüre katkıda bulunmayı amaçlamaktadır. Makine öğrenmesi ve derin öğrenme yöntemleri, yani karar ağaçları, K-ortalamalar modeli ve evrişimli sinir ağları (CNN'ler), konfigürasyon oluşturma sürecini ilerletmek için kullanılmaktadır ve son konfigürasyonu birleştirmek için robot simülasyonu kullanılması önerilmektedir. Yığma duvar görselleri, stereotomi ve istiflemenin özelliklerini yansıtabildiklerinden dolayı veri kümesini oluşturmak için seçilmiştir. Görselleri toplayıp kalitelerine göre eledikten sonra, ana veri kümesi için toplam 393 görsel

kullanılmıştır. Karar ağacı , Kısıtlanmış K-ortalamlar ve CNN modellerini eğitmek için kullanılan görüntülerin özelliklerini çıkarmak için bir kenar algılama algoritmasını kullanan bir veri ön işleme uygulanmaktadır. Bu modellerden oluşan bir iş akışı, son konfigürasyonlar için olası temel desenleri algılamak üzere önerilmiştir. Bu araştırmada, stereotomik ilkelerle ilgili bir dizi geometrik işlem, tahmin ve yerleştirme algoritmasına dayanan önerilen iş akışının, robot simülasyonunda birleştirilebilecek bir dizi ayrı eleman için ikna edici konfigürasyonların oluşturulabildiği gösterilmiştir.

Anahtar Kelimeler: Ayrık Elemenlar, Stereotomy, Makine Öğrenmesi, Yerleştirme Konfigürasyonu, Robot Teknolojisi



To my dear family

ACKNOWLEDGMENTS

I would like to begin by expressing my deepest gratitude to Prof. Dr. Arzu Gönenc Sorguç. She has been more than an advisor, guiding me throughout my academic journey. This thesis would not have been possible without her contributions. Her extensive knowledge and experience have broadened my vision and improved my technical skills. I was thrilled when she accepted me as her student, and it has been an honor to work under her guidance. It is also very encouraging to know that I will continue to feel her support in the future, as I did during this thesis process.

I would also like to sincerely thank Dr. Müge Kruşa Yemişçioğlu for her support, with her advanced knowledge and technical expertise during my studies. Her kindness and patience helped me stay on track while preparing me for my future career. I also want to thank Res. Asst. Ozan Yetkin for broadening my perspective with his extensive knowledge. Together, they, along with the environment in DDS, provided me with the motivation to continue my work.

I want to show my respect and thank to the valuable jury members, Assoc. Prof. Dr. Mehmet Koray Pekerçli, Assoc. Prof. Dr. Ayşem Berrin Çakmaklı, Assist. Prof. Dr. Işıl Ruhi Sipahioğlu and Assoc. Prof. Dr. İlker Erkan for their substantial comments and contributions during my defence.

I would like to thank the TÜBİTAK 2210-A Domestic General Postgraduate Scholarship Program for supporting me in pursuing my degree.

I would like to wholeheartedly acknowledge that I would not be here if it weren't for my parents, İsmail Solmaz and Hatice Solmaz. They pushed their limits and did their very best for me. I also sincerely thank Sevde Ceyda Solmaz—*my twin, my other half*—and Zeynep Gül Solmaz—*my precious little sister*—for always being there for me. I can't imagine life without them.

I would like to say that even thanking my dear friend and sister Rumeysa Dönmez feels insufficient. Even when she is not with me, I know she is always there for me, just a phone call away, whether I need her or not. It is a wonderful feeling to have taken every step with her since the day we met.

I would like to note that I gained invaluable friendships during my master's studies that I could not have even dreamed of. In particular, I am grateful to Sena Nur Cabadağ, who was with me through both difficult and memorable times. She is also one of those exceptional people who can always teach me something new while we spend time together.

After all, I would like to thank my dear friend and companion, Bade Eloğlu. When I started my master's degree, I never imagined that such a valuable and close person would come into my life—someone with whom I could share everything. After the three years we spent together, our friendship felt like it was worth much more, and she became my family in Ankara. She never withheld her support throughout the entire process and was always there to help me develop my study, as well as soothe me emotionally and mentally. We took many steps together toward earning our degrees. I am very grateful for this friendship, which I am sure will last a lifetime.

TABLE OF CONTENTS

ABSTRACT	v
ÖZ.....	vii
ACKNOWLEDGMENTS.....	x
TABLE OF CONTENTS	xii
LIST OF TABLES	xv
LIST OF FIGURES.....	xvi
LIST OF ABBREVIATIONS	xx
1 INTRODUCTION.....	1
1.1 Problem Definition	4
1.2 Hypothesis and Research Questions.....	4
1.3 Research Objectives	5
1.4 Methodology.....	5
1.5 Disposition.....	9
2 LITERATURE REVIEW	11
2.1 Modularity	11
2.2 Masonry as Construction Practice	13
2.3 Building with Blocks.....	14
2.4 Stacking	14
2.5 Stereotomy.....	17
2.6 Cyclopean Walls.....	17
2.7 Digital Stereotomy.....	19
2.8 Related Works	20
2.8.1 Reuse	20

2.8.2	Artificial Intelligence	27
2.8.3	Robotics	28
2.9	Keyword Maps	30
2.10	Critical Review	35
2.11	Contributions.....	36
3	METHODOLOGY AND DATA PREPROCESSING	39
3.1	Data Collection	40
3.2	Creating the Configuration	43
4	TRAININGS OF MACHINE LEARNING MODELS	45
4.1	Classification with Tree-Based Machine Learning Models.....	45
4.1.1	Decision Tree Model.....	47
4.1.2	Results of the Decision Tree Model.....	48
4.1.3	Extreme Gradient Boosting Model	57
4.1.4	Results of the Extreme Gradient Boosting Model	57
4.2	Classification with Convolutional Neural Networks	63
4.3	K-Means Clustering	71
5	PLACEMENT AND ROBOT SIMULATION	75
5.1	Implementation of Proposed Method.....	75
5.2	Placement of Pieces into Matched Patterns	78
5.3	Robot Simulation	82
5.4	Validation of the Method	85
5.5	Evaluation of the Placement	86
5.5.1	Similarity Measures	86
5.5.2	Placement with Similarity Measures.....	88

5.5.3	Applicability of the Placement with Tetris.....	94
5.5.4	Applicability of the Placement with the City Plan of Barcelona.....	96
6	CONCLUSION	101
6.1	Discussion.....	101
6.2	Conclusion.....	103
6.3	Future Remarks	103
6.4	Limitations.....	104
	REFERENCES	105
7	APPENDICES.....	115
A.	CSV File	115
B.	Images.....	126

LIST OF TABLES

TABLES

Table 4.1 Results of the Scikit-Learn Decision Tree Model	49
Table 4.2 Classification Report of the 5th Decision Tree Model	51
Table 4.3 Results of the Custom Decision Tree	56
Table 4.4 Results of the Extreme Gradient Boosting	58
Table 4.5 Classification Report of 4th XGBoost Model.....	60
Table 4.6 Results of Pattern Classification with CNN.....	64
Table 4.7 Architecture of the VGG-16 Model (Zhang et al., 2016)	66
Table 4.8 Classification Report of Pre-trained VGG-16 Model for Pattern Classification.....	66
Table 4.9 Results of Individual Stone Classification with CNN	69
Table 4.10 Classification Report of Pre-trained VGG-16 Model for Individual Stone Classification	70
Table 5.1 Feature Distances	92
Table 5.2 Overlaps	93
Table 5.3 Coverage Distances.....	93
Table 5.4 Evaluation of the Placements Based on the Barcelona City Plan.....	99

LIST OF FIGURES

FIGURES

Figure 1.1 Structure of the Research	3
Figure 1.2 Constructivist Research Guideline	6
Figure 1.3 Methodology of the Research	8
Figure 2.1 Modules of Alessi Tea & Coffee Towers (M Rocker, 2006).....	12
Figure 2.2 Robotic Construction Analysis of a Stone Masonry Arch in Viterbo, Italy (Paris et al., 2023)	14
Figure 2.3 Compression-based structure with wooden pieces (Hunkar & Lee, 2023)	16
Figure 2.4 The designed stacked skyscraper (Larsson et al., 2012).	16
Figure 2.5 Incan Cyclopean Wall (Clifford & McGee, 2018).....	18
Figure 2.6 Cyclopean Walls in Mycenae (Barun & Kranjec-Markešić, 2015).....	18
Figure 2.7 A Cyclopean Wall of Inca (Squier, 1877).....	19
Figure 2.8 Sectional Prototype of Plasticity Pavilion (Diles, 2018).....	20
Figure 2.9 European Council Building, Brussels (Rojas, 2016) retrieved from: https://www.archdaily.com/802379/european-council-and-council-of-the-european-union-samyn-and-partners	21
Figure 2.10 The scanning process of the rubble concrete at Tinguely Recycle SA, Vaud, Switzerland (Marshall & Lausanne, 2024)	22
Figure 2.11 Training Data (Marshall & Lausanne, 2024)	22
Figure 2.12 The proximity between the matched pairs for the one-to-one examples (Marshall & Lausanne, 2024).....	23
Figure 2.13 Reclaimed stair (Jhons, 2023)	23
Figure 2.14 An incorrect placement (Jhons, 2023)	24
Figure 2.15 Reclaiming, collecting, and numbering stone blocks (Wilton et al., 2025).....	24
Figure 2.16 Mine The Scrap Art pieces (Certain Measures, n.d.) retrieved from: https://www.certainmeasures.com/projects/mine-the-scrap	25

Figure 2.17 Workflow of the study (Liu et al., 2024).....	26
Figure 2.18 Building Aggregation, Pavilion Scale (Yu and Fingrut, 2022)	27
Figure 2.19 Clustering Results (Wang et al., 2024).....	28
Figure 2.20 Produced Example a) original wall b) optimized wall (Wang et al., 2024)	28
Figure 2.21 Tower Stacking Workflow (Wermelinger et al., 2018).....	29
Figure 2.22 Irregular Object Stacking Based on Loads (Zhou et al., 2023)	30
Figure 2.23 Keyword Map "Stereotomy" and "Robotics" 22.04.2025	31
Figure 2.24 Keyword Map "Stereotomy" and "Masonry" 22.04.2025	31
Figure 2.25 Keyword Map "Stereotomy" and "Masonry" – Focused on Masonry 22.04.2025.....	32
Figure 2.26 Keyword Map "Stereotomy" and "Masonry" – Focused on Stereotomy 22.04.2025.....	32
Figure 2.27 Keyword Map "Robotics" and "Masonry" 22.04.2025	33
Figure 2.28 Keyword Map "Stereotomy" and "Artificial Intelligence" 22.04.2025.....	33
Figure 2.29 Keyword Map "Machine Learning" and "Masonry" 28.04.2025	34
Figure 2.30 Keyword Map "Machine Learning" and "Masonry" - Focused on Machine Learning 28.04.2025	34
Figure 2.31 Keyword Map "Machine Learning" and "Masonry" - Focused on Masonry	35
Figure 3.1 Examples of the Masonry Wall Images.....	40
Figure 3.2 Pseudo-Code of Edge Detection.....	40
Figure 3.3 Examples of Stone Contours	42
Figure 4.1 Pair Plotting	45
Figure 4.2 Correlation Heatmap	46
Figure 4.3 Confusion Matrix of 5th Decision Tree Model	52
Figure 4.4 Structure of the 5th Decision Tree Model	53
Figure 4.5 True Predictions.....	54
Figure 4.6 False Predictions.....	54
Figure 4.7 Confusion Matrix of 4th XGBoost Model.....	60

Figure 4.8 True Predictions	61
Figure 4.9 False Predictions	61
Figure 4.10 Log Loss of the 4th XGBoost Model.....	62
Figure 4.11 Receiver Operating Curve of 4th XGBoost Model.....	62
Figure 4.12 Illustration of the Convolutional Operation (Wu, 2017).....	63
Figure 4.13 Confusion Matrix of the Pre-trained VGG-16 Model for Pattern Classification	67
Figure 4.14 True Predictions	67
Figure 4.15 False Predictions	67
Figure 4.16 Receiver Operating Characteristics Curve of Pre-trained VGG-16 Model for Pattern Classification.....	68
Figure 4.17 Training and Validation Loss of Pre-trained VGG-16 Model for Pattern Classification	68
Figure 4.18 Confusion Matrix of the Pre-trained VGG-16 Model for Individual Stone Classification.....	70
Figure 4.19 True Predictions	70
Figure 4.20 False Predictions	71
Figure 4.21 Training and Validation Loss of Pre-trained VGG-16 Model for Individual Stone Classification.....	71
Figure 4.22 Elbow Method for Optimal K	72
Figure 4.23 Pair Plotting with Clusters	73
Figure 4.24 Examples from Clusters	73
Figure 5.1 Sampled Pieces and Predicted Base Patterns of Batch 0	77
Figure 5.2 Sampled Pieces and Predicted Base Patterns of Batch 5	77
Figure 5.3 Sampled Pieces and Predicted Base Patterns of Batch 5	78
Figure 5.4 Pseudocode of Placement Algorithm	79
Figure 5.5 Example from Batch0 - Random Rubble	81
Figure 5.6 Example from Batch5 - Polygonal Rubble	81
Figure 5.7 Example from Batch10 - Square Rubble	81
Figure 5.8 Pseudo-code of Robot Simulation.....	82

Figure 5.9 Robot and Vacuum Gripper.....	83
Figure 5.10 Imported Pieces	84
Figure 5.11 Selected Pattern for Validation.....	85
Figure 5.12 Polygonal Curve and Its Turning Function (Veltkamp, 2001).....	87
Figure 5.13 Example from Batch 0 - Random Rubble - Hausdorff Distance.....	88
Figure 5.14 Example from Batch 0 - Random Rubble - Minkowski Distance.....	88
Figure 5.15 Example from Batch 0 - Random Rubble - Minkowski Distance.....	89
Figure 5.16 Example from Batch 0 - Random Rubble - Turning Functions Distance	89
Figure 5.17 Example from Batch 5 - Polygonal Rubble - Hausdorff Distance	89
Figure 5.18 Example from Batch 5 - Polygonal Rubble - Minkowski Distance	90
Figure 5.19 Example from Batch 5 - Polygonal Rubble - Bottleneck Distance	90
Figure 5.20 Example from Batch 5 - Polygonal Rubble - Turning Functions Distance.....	90
Figure 5.21 Example from Batch 10 - Square Rubble - Hausdorff Distance	91
Figure 5.22 Example from Batch 10 - Square Rubble - Minkowski Distance	91
Figure 5.23 Example from Batch 10 - Square Rubble - Bottleneck Distance	91
Figure 5.24 Example from Batch 10 - Square Rubble - Turning Functions Distance	92
Figure 5.25 Pattern with Tetris Blocks	94
Figure 5.26 Conflicted Tetris Placement	95
Figure 5.27 Tetris Blocks that Belong to 2 Different Convex Hulls	95
Figure 5.28 Simple Tetris Configuration with Distinct Blocks	96
Figure 5.29 Division of the City of Barcelona for Testing the Applicability	96
Figure 5.30 Placement of Blocks of Mountainside to Patterns of Seaside	97
Figure 5.31 Placement of Blocks of Seaside to Patterns of Mountainside	98

LIST OF ABBREVIATIONS

ABBREVIATIONS

AI	Artificial Intelligence
ANNs	Artificial Neural Networks
AUC	Area Under the Curve
CNNs	Convolutional Neural Networks
CNC	Computer Numerical Control
CV	Computer Vision
DL	Deep Learning
EPS	Expanded Polystyrene
ML	Machine Learning
PCA	Principal Component Analysis
ROC	Receiver Operating Curves
XGBoost	Extreme Gradient Boosting

CHAPTER 1

INTRODUCTION

The field of architecture has always involved modularity as a construction method since it enables us to constitute a structure from discrete elements by mainly stacking them. The operation of stacking can be seen in both man-made structures as well as nature-based structures like bird nests. Accordingly, the concept of modularity, which can be commonly seen in the masonry structures mostly made with stones and bricks, is a topic that has evolved throughout history with the use of different structural members. Yet, the modules don't have to be structural members; they can also be architectural components. Correspondingly, the scale in structures created by stacking differs according to the context, which can be small modules composing an architectural component like a wall or large modules composing a whole building. This study focuses on smaller modules that can be stacked by masonry principles. As technology progresses, construction techniques are also developing, including masonry, along with the increase in the variety of materials. These novel materials could provide various kinds of shapes, including non-conventional forms. Besides, the existing materials, like reclaimed ones that could have irregular forms, could be utilized in such structures.

The use of modules with irregular/non-conventional forms to create a self-standing structure has become a prominent precision problem on a different scale. This precision problem arising from assembling various kinds of units has been attempted to be solved both in the past and present within the context of architecture. For such a problem, this study proposes adopting a stereotomic approach, as it has high potential in terms of its stacking characteristics. In this regard, modules are combined to have a complete structure with high precision, which ensures stability. Along with

the precision, accomplishing this kind of task requires several other aspects like time, skill, and labor.

Considering the recent advancements in the field of architecture, the types and even forms of structures are changing. This change also occurs in the structural elements with non-conventional forms that could arise from salvage materials with irregular shapes. In recent years, the use of salvage materials has become prevalent within the scope of circularity. Creating new assemblies with salvaged pieces serves to circular strategies by reducing the carbon emissions as well as waste. The outcomes of such studies, which are composed according to the decisions of architects and engineers, mostly fail to satisfy aesthetic needs (Grangeot et al., 2023). These materials can be gathered from the urban depositories, which are the inevitable results of demolition. This increased amount of demolition is caused by rapid urbanization. Since the existing materials from waste are generally irregular despite the new materials, architects and designers should put more effort and creativity into reusing them (Huang et al., 2021).

The change in the forms of materials creating precision problems, including those gathered from urban depositories, can be overcome by adapting the stereotomy concept, incorporating machine learning (ML) methods, which is a featured subject. Following this, the use of robotics can help to manage the need for labor and skill to produce the output structure generated by the proposed method based on stereotomy. The figure below outlines the structure of the research depending on the argument explained above.

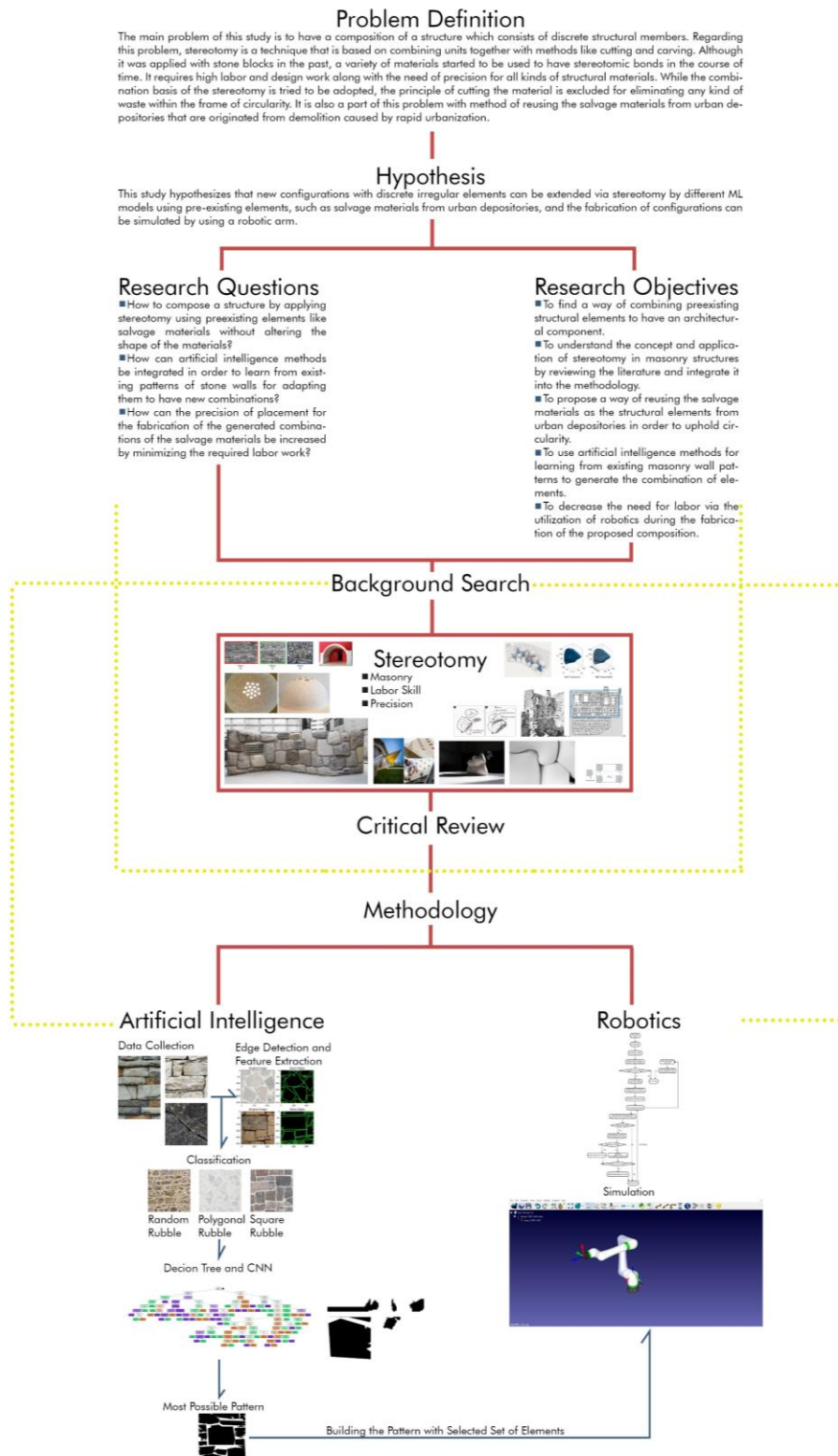


Figure 1.1 Structure of the Research

1.1 Problem Definition

The main problem of this study is to create a configuration of a structure to stack that consists of discrete members that could have non-conventional shapes. Regarding this problem, stereotomy is a technique that is based on combining units to have a tightly fitted structure, commonly applying methods like cutting and carving. Although it was applied with stone blocks in the past, a variety of materials started to be utilized for establishing stereotomic bonds over time. In fact, it is generally associated with works that use stone, yet the etymology of the word does not indicate a material (Fallacara & Stigliano, 2012). It requires high labor and design work, along with the need for precision for all kinds of structural materials. While the combination logic of the stereotomy is tried to be adopted, the principle of cutting the material is excluded to eliminate any kind of waste within the frame of circularity that is also a concern of this study with the proposal of reusing the salvage materials from urban depositories that originated from demolition caused by rapid urbanization.

1.2 Hypothesis and Research Questions

This study hypothesizes that new configurations with discrete irregular elements can be extended via stereotomy by different ML models using pre-existing elements, such as salvage materials from urban depositories, and the fabrication of configurations can be simulated by using a robotic arm. Regarding this hypothesis, the research questions are as follows:

- How to compose a structure by applying stereotomy, using pre-existing elements like salvage materials without altering the shape of the materials?
- How can machine learning methods be integrated into the process of learning from existing patterns of stone masonry walls and adapting them to have new configurations?

- How can the precision of placement for the fabrication of the generated configurations with the salvage materials be increased by minimizing the required labor work?

1.3 Research Objectives

- To find a way of combining pre-existing structural elements to have an architectural component.
- To understand the concept and application of stereotomy in masonry structures by reviewing the literature and integrating it into the methodology.
- To propose a way of reusing the salvage materials as structural elements from urban depositories in order to uphold circularity.
- To use machine learning methods for learning from existing masonry wall patterns to generate a combination of elements.
- To decrease the need for labor via the utilization of robotics during the fabrication of the proposed composition.

1.4 Methodology

This research builds a methodology to generate and assemble a composition of structural elements, which are salvage materials in this case. Firstly, the constructivist research guideline proposed by Hevner et al. (2004) is filled up to see the general approach of the study, which can be seen in Figure 1.2. The guideline consists of design as an artifact, problem relevance, design evaluation, research contributions, research rigor, design as a search process, and communication of research.

Guideline	Description in the Context of This Research
Design as an Artifact	This research proposes a framework to generate combinations of stereotomical bonds of urban depository materials using decision trees and CNN models. The final production through robotics serves as a practical artifact of this process.
Problem Relevance	Understanding the stereotomy of elements Classifying existing patterns to have options of compositions Ensuring circularity by reuse of elements Decreasing the labor work by utilizing high tech as robotics
Design Evaluation	The effectiveness of the machine learning models is evaluated by testing the viability of generated designs. Further assessment of robotic production confirms the feasibility of these combinations.
Research Contributions	This work introduces a novel approach that benefits from machine learning techniques with a gathered knowledge from stereotomical design principles, adding value to sustainable urban development and automated construction.
Research Rigor	The study employs robust machine learning techniques, including CNNs for image classification and decision trees for decision-making to have stereotomic bonds of nonuniform elements differing from classical stereotomy with resuing elements from urban depositories in the light of deduction from literature review.
Design as a Search Process	This research views design as an iterative process of refining configurations. Machine learning models are used to generate and adjust combinations of reuse elements, which are ultimately produced through robotic fabrication.
Communication of Research	The findings should be presented to audiences of both academy and industry to enhance their applicability.

Figure 1.2 Constructivist Research Guideline

Then the methodology starts with literature mainly based on stereotomy, masonry, and modularity, and beyond that, the related work that can be helpful for the method is also examined. A critical review of this literature is composed after surveying it according to the requirements of the aim of this study. This review addresses the deficiencies in the related field that can be filled with the contributions made by this research. After that, a keyword analysis of the literature is done to see the concepts that are leading the field. The contributions of this study, which will be explained in

the following parts, will be planned to be initiated with machine learning models to introduce stereotomy for solving the above-mentioned research problems.

There is a need for training data for utilizing the intended ML models, which are decided as existing masonry wall patterns. Since there are more than one type of ML models that will be trained, two types of data processing are applied following the edge detection operation from the images. Due to the limited number of data, 3 different classes are used amongst various types of masonry walls, which are determined by observation that is confirmed via literature. After the first models for the classification task, an additional labeling strategy is introduced to find the most probable patterns for the output structure with a set of structural elements. Finally, the fabrication of the composition according to these patterns will be simulated with a robot arm. The general pipeline of the methodology can be seen in the following figure.

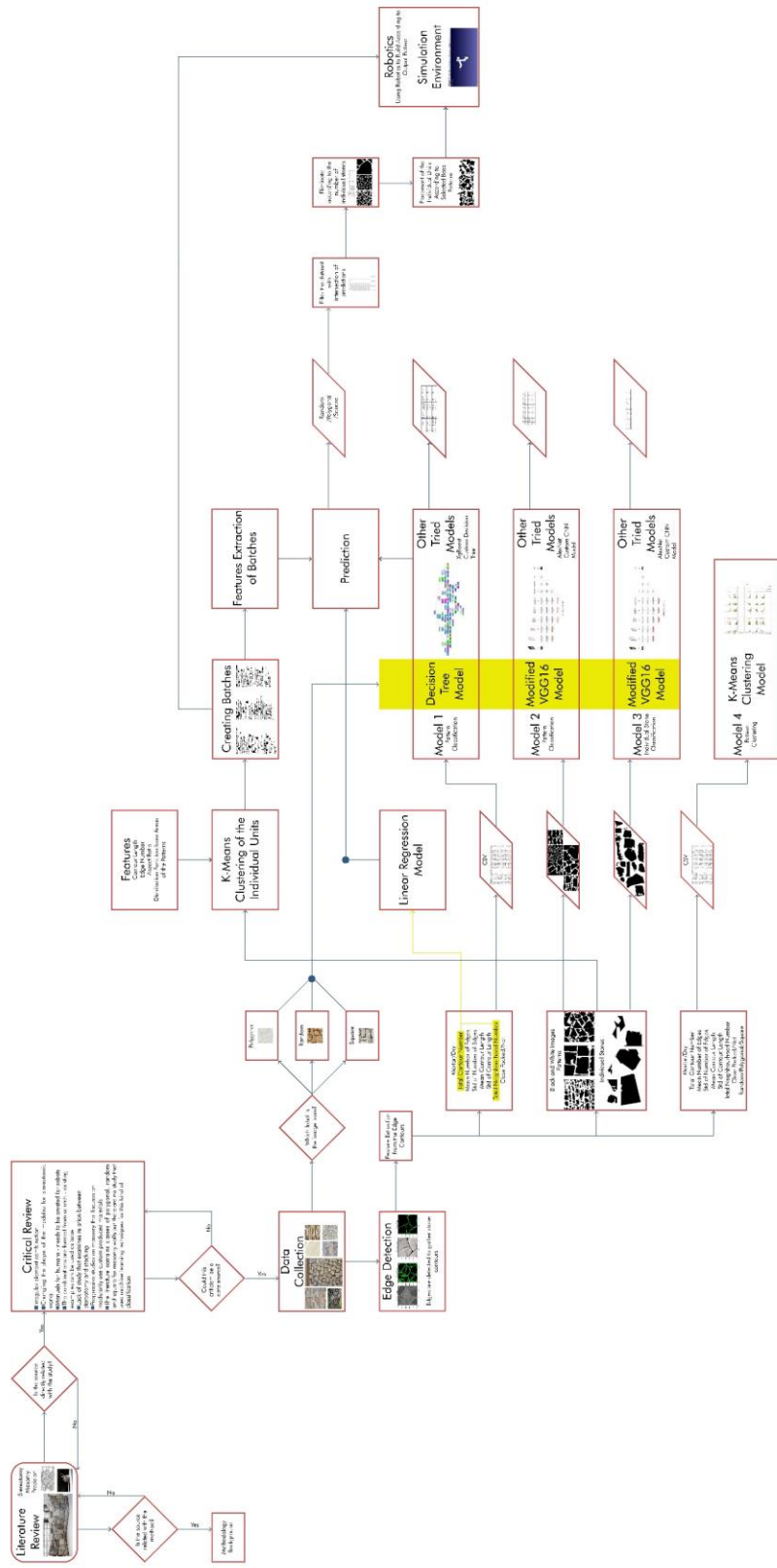


Figure 1.3 Methodology of the Research

1.5 Disposition

The rest of the study comprises five main chapters and related subheadings. Chapter 2 gives information on related work that is done in the field as a literature review that mainly focuses on modularity, stereotomy, and masonry. Chapter 3 contains the general approach of the methodology by starting from data collection and processing. Chapter 4 gives the results of different machine learning trainings, and Chapter 5 represents the placement logic, implementation of the proposed model, the robot simulation, and evaluation of the placement algorithm. As the last step, Chapter 6 concludes the research with a discussion and future remarks.



CHAPTER 2

LITERATURE REVIEW

This chapter reviews the literature related to construction techniques for modular elements under three main headings. The first part examines modularity, specific masonry, and stacking in construction. The second part presents the stereotomy and its evolution throughout history. Finally, the last part reviews the studies on combining different units, reuse, and robotics, especially for stacking problems, as a background search with respect to the methodology of this research.

2.1 Modularity

Modularity is a construction technique that consists of smaller pieces, which is an alternative approach to a monolithic way of construction. It is even possible to build shell structures with this technique, like sectioning the structure into smaller units and fabricating and assembling them on the construction site. Differing through the connection between the modules, they can be under both compression and tension forces (Ivaniuk et al., 2022), so that they should be designed to resist both. There are various applications of modularity in the fields of architecture and design. Besides the early studies, contemporary approaches are expanding, such as works done by Greg Lynn are valuable examples, including the tea and coffee towers, whose modules can be seen in Figure 2.1 (M Rocker, 2006).

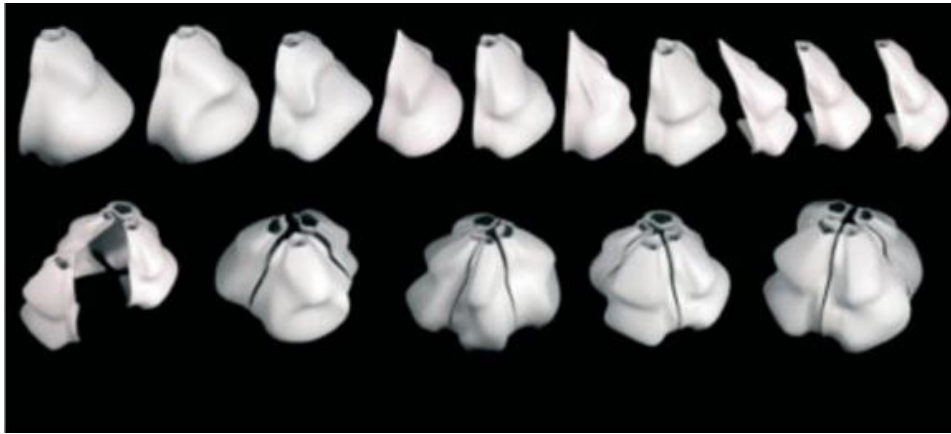


Figure 2.1 Modules of Alessi Tea & Coffee Towers (M Rocker, 2006)

Another study by him is the Blobwall, which consists of custom-built plastic modules that are lightweight and colorful and produced by Computer Numerical Control (CNC) machines (Lynn, 2009). Apart from the innovative examples, most modular systems are mainly composed of the same modules in repetition, which limits the application range because of the joint restrictions (Brütting et al., 2021). There are many benefits to applying modulation, as it may provide more functional and stable assemblies with effective time management through automation (Graur, 2020).

The modules used in the structure differ amongst various examples; another innovative example can be given as the coreless winding units to build pavilions designed by computational tools and built by robots (Knippers et al., 2016). A further example is containing expanded polystyrene (EPS) foam corrugated modules designed to facilitate robotic re-assembly using a simulation environment for optimization (Belousov et al., 2022). Modularity finds space in stereotomic works through the production of units by CNC milling, waterjet, and wire-cutting operations. After these operations, the aggregation strategy is one of the crucial parts, mostly depending on the material properties (Fernando, 2019). Modules in contemporary architectural studies could be used as an extension rather than comprising the main structure, like in the coral-inspired modules articulated onto a timber skeleton (Guillén et al., 2025).

2.2 Masonry as Construction Practice

The harmony between form and structure, which comes from the nature of the masonry construction technique, allows different structural elements from reduced dimensions to be connected via interlocking and can determine both expressive power, aesthetics, and structural durability at the same time (Fallacara & Gadaleta, 2021). Masonry is a construction method that depends on building units to compose layers to create courses from top to bottom (Zandavali & García, 2019). These units can be aggregated with a bonding material, such as mortar, or they can be dry-stacked. There are so many possibilities for arranging the units that could result mostly in overlapping, since a vertical joint alignment can cause disastrous weakness (Foraboschi, 2019). Each has advantages and disadvantages, like the mortar can provide a binding feature between the units to hold them together, while the ones without mortar can prevent cracking caused by shrinkage that conventional masonry usually has (Thamboo et al., 2020). The materials that are used for composing masonry structures, like stones and bricks, are mostly in compression, and they are not resistant to tensile stresses. The mortar provides the distribution of these stresses rather than only its binding feature (Rajabzadeh & Sassone, 2016).

Moreover, masonry's computerization and automation process started long ago, mostly focusing on engineering, while its design has become more significant lately (Zandavali & García, 2019). As a part of this automation, digital fabrication techniques have started to be used to yield stereotypical buildings for stone masonry (Varela & Sousa, 2019). To achieve this task, a variety of tools can be used, including drones and robotic arms that can handle discrete elements with moderate dimensions for building a structure with the technique of masonry. As an initial point, there is a need for a geometrical model of the structure that will be built to keep the information on the details, building units, and stereotomy of these units to understand their location and general composition (Paris et al., 2023). An example of a robotic construction analysis for an existing stone masonry can be seen in Figure 2.2. The construction information should be transferred to the agent who will produce it.

Since the data on dry-stacking is mainly gathered from manuals of masonry techniques written for humans, the information should be processed to be transferred according to the autonomous agent (Liu & Napp, 2023).

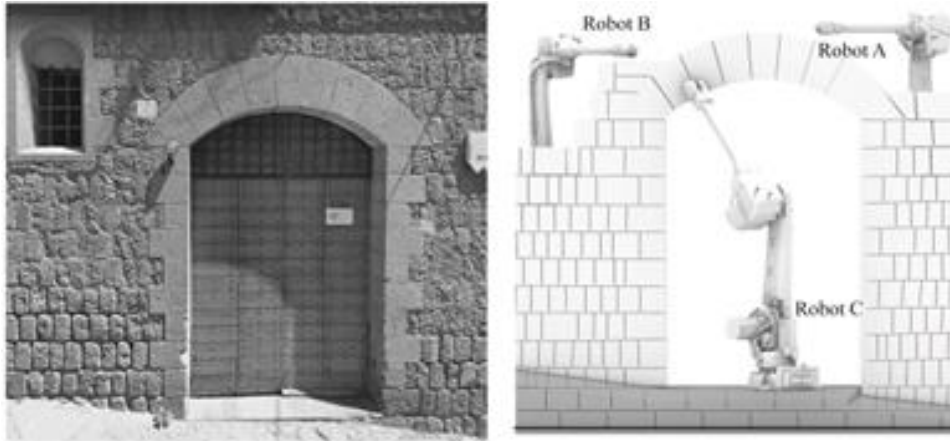


Figure 2.2 Robotic Construction Analysis of a Stone Masonry Arch in Viterbo, Italy (Paris et al., 2023)

2.3 Building with Blocks

Creating a structure is viable by using various kinds of blocks with a diversity of shapes and materials, which refers to modularity, which was mentioned earlier. In order to constitute a structure, it is possible to have dome-shaped and even corbelled structures with interlocking blocks. There can be alternate methods for aggregating blocks rather than simple interlocking systems that may enable more adjustability (Zhang et al., 2023). When composing a structure from discrete blocks, if the connection parts cannot be calculated carefully, there is a risk of collapsing (Foraboschi, 2019). Along with recent studies, blocks were used in historical examples to compose structures, mostly with cut stones.

2.4 Stacking

Stacking is an assembly technique that ensures stability and can be performed using prefabricated units like those used in the works of Mostafavi et al. (2023). There is

a high relation between stacking and masonry, considering it depends on accumulating units on each other; thus, a masonry wall is formed by stacking the modules together. When no adhesive material is used for bonding the elements, it is called dry stacking, which dates back to ancient times, as Neolithic humans built their shelters with this technique (Liu & Napp, 2023). Stonemasons built these early examples, and the information on the technique was transferred to other generations by practice (Flynn, 2015). The main target of stacking is to increase the number of elements while maintaining stability, and it is possible to achieve it using a trial-and-error method. Nevertheless, to have a dry-stack arrangement that is self-standing and has balance, planning algorithms are applied with the development of technology (Ko & Hsieh, 2023). For organizing the arrangement, it is important to categorize the elements that compose the system according to their features; the most crucial feature is the size to create the accumulation (Thangavelu et al., 2018).

Despite the packing and stacking problems having similar stances, stacking is a much more difficult task due to the number of constraints that it deals with, including stability. For instance, a set of objects can be packed flawlessly, while the same orientation may not stand without a container (Liu et al., 2019; Liu, 2023). To achieve stability, each object should be placed one by one, calculating the stamina of the general composition (Wermelinger et al., 2018; Furrer et al., 2017). Apart from conventional methods, stacking is also used in more innovative approaches, such as digital fabrications. One example is the structure composed of post-tensioned funicular masonry done by Bolhassani and Wisniewski (2022). This structure is composed of units that are stacked together according to the detailed model produced in Rhinoceros. As another example, Chiang et al. (2018) conducted research based on stacking strategies based on sequences from bottom to top to enable stability. These types of structures are compression-based structures, and one example that creates one of them is done by Hunkar & Lee (2023) without any binding materials (Figure 2.3).



Figure 2.3 Compression-based structure with wooden pieces (Hunkar & Lee, 2023)

Another example is stacking the modules with a different scale, which are prefabricated timber units, to form a skyscraper has different volumes by shifting the used modules. The form of this stacked skyscraper is planned to have different volumes as well as providing an internal connection within the building (Larsson et al., 2012).

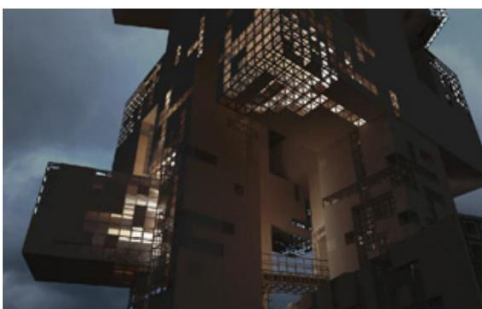


Figure 2.4 The designed stacked skyscraper (Larsson et al., 2012).

Another study based on stacking on the building scale was done by Müller-Zitzke (2024), which creates a checkered pattern by stacking prefabricated modules. The modules offer different spatial configurations for different functions.

2.5 Stereotomy

In order to understand the stacking of objects, the concept of stereotomy should be inspected since it is directly related to how pieces come together to compose a structure. The word stereotomy is considered to come from the combination of two words, which are "solid" and "cut" (Fallacara & Barberio, 2018). The art of stereotomy is indicated as precisely cutting the solids to fit them firmly (Sanabria, 1989). It emerges from the organic conjunction between small-scale elements of the structure with various materials. The refinement of geometry is the most crucial part of functional, aesthetic, and static structures (Fallacara & Gadaleta, 2021). So, it can be defined as a self-supporting system that is based on tessellations (Diles, 2018). After a period from its emergence, the concept of stereotomy in the construction field was re-emphasized in the early 1990s, in order to benefit from its potential of design via integrating the method into the education in Ecole des Beaux-Arts and the Ecole Polytechnique (Fallacara & Barberio, 2018; Fallacara & Stigliano, 2012; Varela & Sousa, 2016).

2.6 Cyclopean Walls

Cyclopean walls are ancient structures composed of large stone blocks, which can be called megaliths, that are generally dry-stacked. These stone blocks are fitted together by carving and cutting them. The megalithic walls built by the Inca civilization, which are the most important examples, are located on a mountain at an altitude of 3,400 m in their capital city of Cusco, which includes the well-known instance of Machu Picchu that has been under UNESCO protection since 1983. They used the method of dressing rather than carving, and a close-up view of a wall can be seen in Figure 2.5. Their construction methodology is based on placing the stone blocks one by one instead of creating a composition in advance (Clifford & McGee, 2018; Šaravanja, Oreč, & Kopilaš, 2023). Whereas the sequence of the elements is

important since these walls have irregular bonds. Apart from that, it can be observed from the remaining walls that they gave importance to precision (Protzen, 1985).



Figure 2.5 Incan Cyclopean Wall (Clifford & McGee, 2018)

These types of walls can be seen in other parts of the world including Mycenae which holds the origin of cyclopean since the Greek civilization believed that these types of huge structures could be built by mythical Cyclops and the walls in Mycenae can be seen in Figure 2.6 (Šaravanja, Oreč, & Kopilaš, 2023). The cyclopean wall of the Inca is illustrated in the travel book of Squier (1877) with a human scale that can be seen in Figure 2.7. He also stated that these walls were the most magnificent example of the Cyclopean style that exists in America.

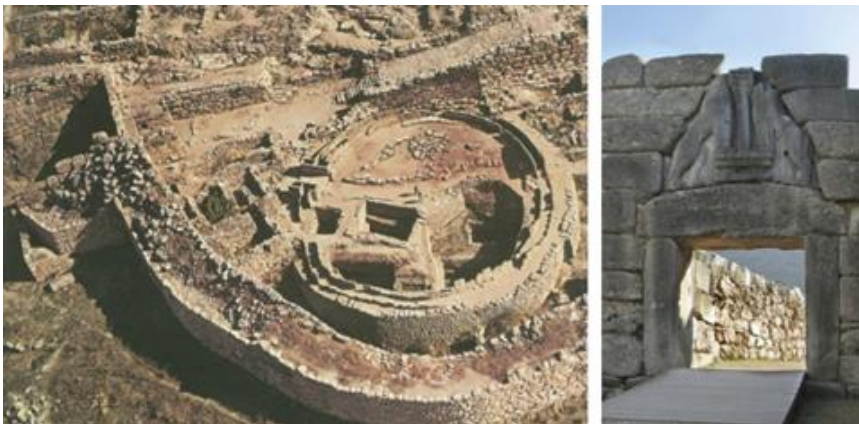


Figure 2.6 Cyclopean Walls in Mycenae (Barun & Kranjec-Markešić, 2015)

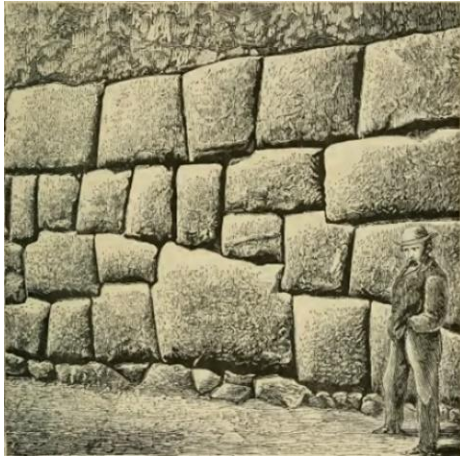


Figure 2.7 A Cyclopean Wall of Inca (Squier, 1877)

2.7 Digital Stereotomy

Utilization of computational methods enhances stereotomic architecture with advanced design possibilities by modeling that produces encouraging outputs. Thanks to these methods, the creation of complex forms and structures emerges within the context of Stereotomy 2.0, namely Digital Stereotomy (Fallacara & Barberio, 2018; Fallacara & Stigliano, 2012; Varela & Sousa, 2016). The characteristics of such complex architectural work can be defined by its geometry, which is determined by the stereotomy rather than traditional knowledge of experimental rules (Fallacara & Stigliano, 2012; Gadaleta, 2018). The computational design principles respond to the requirements of stereotomy both mathematically and geometrically (Colella, 2018). Lightweight materials can be used instead of stone blocks, which is investigated by Diles (2018) through a series of installations with curvilinear forms, and one of them can be seen in Figure 2.8.



Figure 2.8 Sectional Prototype of Plasticity Pavilion (Diles, 2018)

2.8 Related Works

Studies that are linked to reuse and robotics that could be helpful for the methodology of this research are examined in this section to evaluate which approaches can be adopted and what their limitations are, and later to detect the research gap to be able to fill it.

2.8.1 Reuse

Reuse is a concept that ensures a way of providing sustainability within the context of circularity since it decreases carbon emissions and energy consumption. Thus, the number of studies considering reusing locally available materials is increasing (Kang, Dy, & Kane, 2023). An important case designed by Samyn and Partners that uses old windows with an approximate number of a thousand for creating the facade of the European Council Building in Brussels can be seen in Figure 2.9 (Gorgolewski, 2018).



Figure 2.9 European Council Building, Brussels (Rojas, 2016) retrieved from: <https://www.archdaily.com/802379/european-council-and-council-of-the-european-union-samyn-and-partners>

Contemporary studies focusing on reusing waste materials are dealt with computational methods. Yet, it is more difficult than the ones with new materials like the assembly of parametric designs (Huang et al., 2021). The irregularity of materials from debris is the key factor in why these materials are not used, since they require proper organization and classification (Witt, 2022). Although it is possible to parameterize the salvage materials if the present materials are collected in an inventory. Several studies aim to reuse different kinds of materials with digital solutions. One of these studies uses three forks that are collected from a site on the MIT campus to create a gridshell. A matching algorithm is developed for these three forks, and they are connected via extensions. This method is used to build vaulted structures (Cousin et al., 2023). Another study uses reused rubble concrete masonry units to compose structures, which are initialized by scanning the rubble, which can be seen in Figure 2.10, to create the inventory (Marshall & Lausanne, 2024).



Figure 2.10 The scanning process of the rubble concrete at Tinguely Recycle SA, Vaud, Switzerland (Marshall & Lausanne, 2024)

The outputs of the scanning are exported into the Rhino environment to process them, and the inventory created by this method can be seen in the following figure. The study proposes a method to match two pieces of rubble via edge matches. These one-to-one matches are evaluated according to their proximity, which is illustrated in Figure 2.12 (Marshall & Lausanne, 2024). Another study done by Grangeot et al. (2024) manages to create a vaulted structure by using planar reclaimed concrete rubble, as they are referred to as adaptable materials.

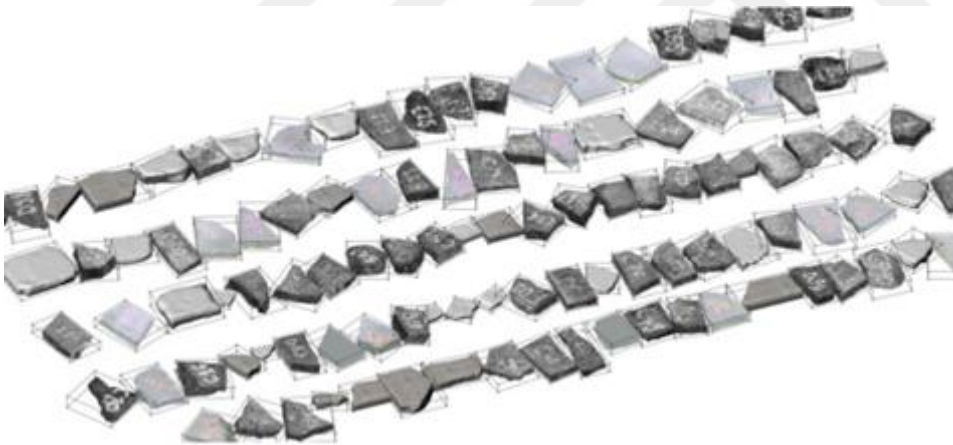


Figure 2.11 Training Data (Marshall & Lausanne, 2024)

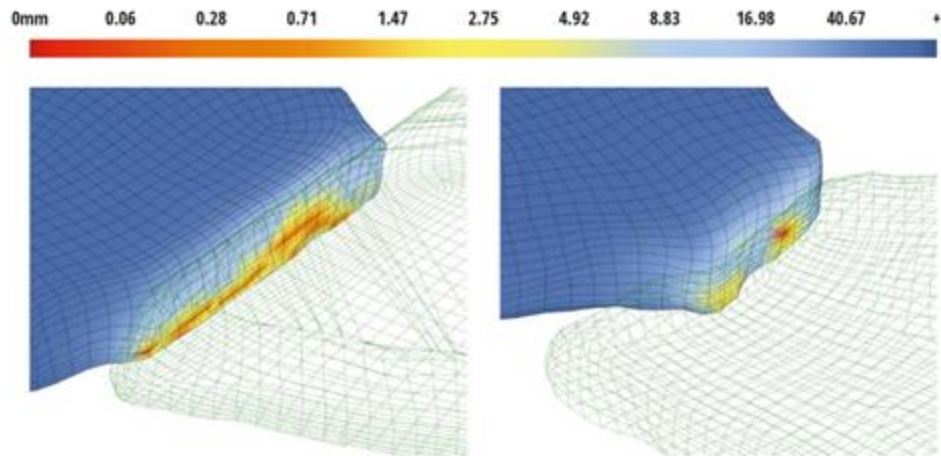


Figure 2.12 The proximity between the matched pairs for the one-to-one examples (Marshall & Lausanne, 2024)

Another similar study was done by Jhons (2023) in his dissertation, which aims to create structures by utilizing reclaimed concrete rubble from demolition sites. Other reclaimed elements are also used in this work, like a concrete stair that can be seen in Figure 2.13.



Figure 2.13 Reclaimed stair (Jhons, 2023)

The requirement of labor and time when employing nonstandard materials is also highlighted in this study. The materials that are gathered are subjected to measuring to obtain a perfect fit. Masonry manuals are used as a reference point with adapting them to the computational techniques used. One of the important things that ensures stability is the number of contacting elements, which is also emphasized in the manuals. They should lie on at least two other stones; an incorrect placement can be seen in the following figure (Jhons, 2023).

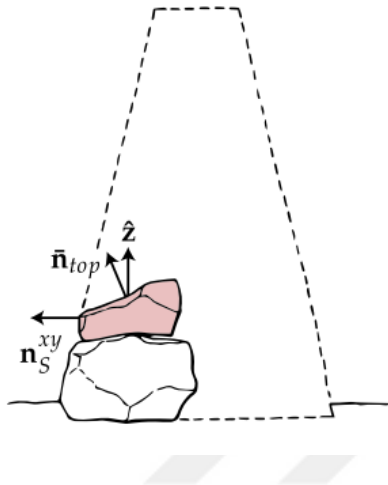


Figure 2.14 An incorrect placement (Jhons, 2023)

Wilton et al. (2025) also adopted a stereotomical approach to create structures with reclaimed stone that are cut according to the final structural configuration based on the ruins of a house. The reclamation and numbering stage of the stone blocks can be seen in Figure 2.15. They ensured to have exact shapes to fit the reclaimed stones through the cutting operation. The elements that are already found in their original forms as columns and beams ease their process.



Figure 2.15 Reclaiming, collecting, and numbering stone blocks (Wilton et al., 2025)

Machine learning methods can be used in the stage of design and planning stage of the assembly for producing complicated structures with such materials. The project Mine the Scrap by Certain Measures employs this strategy to compose art installations (Witt, 2022). Some of their works can be seen in the following figure.



Figure 2.16 Mine The Scrap Art pieces (Certain Measures, n.d.) retrieved from: <https://www.certainmeasures.com/projects/mine-the-scrap>

Another study done by Liu et al. (2024) utilizes waste materials with a flow of recycling, shredding, and classifying them, followed by an end assembly with a robot arm. The waste materials used in this study are the foam blocks that are combined with the implementation of AI-based models. The conducted process of that study, which is illustrated below, has an iterative approach to ensure stability and precision.



Figure 2.17 Workflow of the study (Liu et al., 2024)

At this stage, human intervention is required at some point, yet they plan to reduce it as a future objective, including increasing the adaptability of the system for larger and heavier materials (Liu et al., 2024). Another study created an inventory of timber waste by photogrammetry and laser scanning to be able to use it in different installations and structures. The gathered models are voxelized to have adaptability for different forms, like the example below (Yu and Fingrut, 2022).

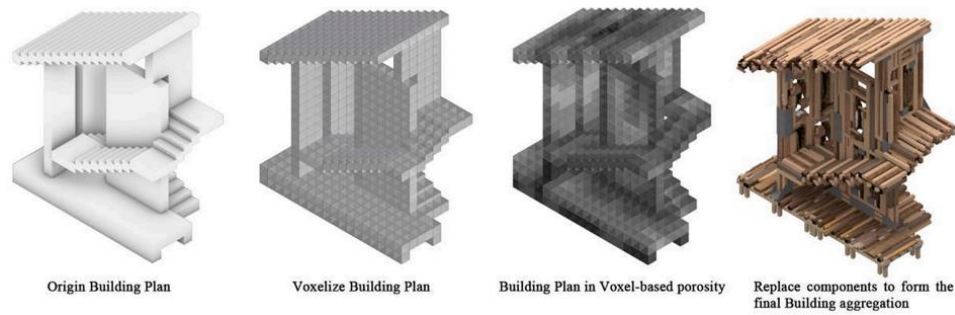


Figure 2.18 Building Aggregation, Pavilion Scale (Yu and Fingrut, 2022)

2.8.2 Artificial Intelligence

Artificial intelligence (AI) is a prominent area that can be applied to a wide range of fields, including architecture. Machine learning (ML) and deep learning (DL) are subtopics of AI that are used for deducing from data. DL mostly deals with a tremendous amount of data within the body of artificial neural networks (ANNs) with multiple hidden layers that could be both supervised and unsupervised (Macaulay & Shafiee, 2022). These methods are also used in stacking and packing problems. One of the inherent advantages of DL, parallelism, is that it allows multiple solutions to be presented in parallel and the best one to be kept, increasing the efficiency of the packing process (Xue et al., 2023).

Since architectural studies focus on 3D space, profitably, it is envisaged that deep neural networks can analyze 3D scenes directly in 3D space by taking advantage of the ability to perceive different scenes simultaneously (Pham et al., 2019). An image based method is adopted in the research of Wang (2024) for irregular stone packing depending on a clustering according to size and eccentricity of the stones. Their clustering result can be seen in Figure 2.19, while the produced example can be seen in Figure 2.20.

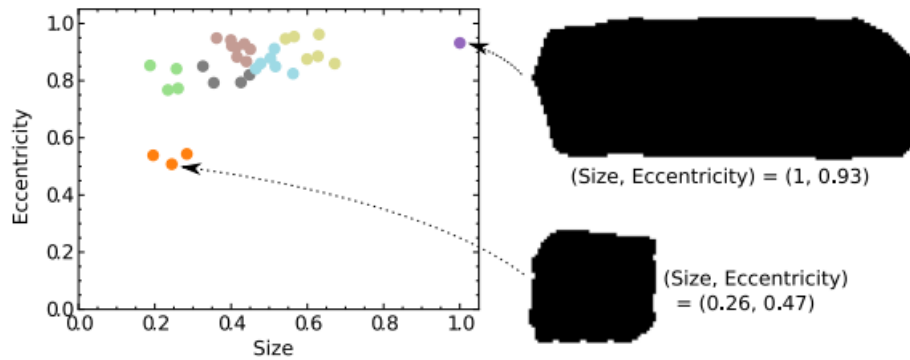


Figure 2.19 Clustering Results (Wang et al., 2024)



Figure 2.20 Produced Example a) original wall b) optimized wall (Wang et al., 2024)

2.8.3 Robotics

Robotics is a growing area that has been integrated into architectural studies at different levels. Within the scope of the construction industry, robots are also making notable progress in stone construction (Zhou et al., 2023). The main factors that are looked at to understand the integration of architecture and robotics are the ability to provide an uninterrupted digital information flow, tolerances in the manufacturing process, and the examination of different research methodologies used in previous studies (Willmann et al., 2012). Dry stacking is a concern of architectural and engineering studies that have been attempted to be solved via robotics. When planning for dry stacking with robotics, two things need to be taken into account that make the structure stand: friction and gravity. The study done by Liu et al. (2018) facilitates dry stacking with irregular pieces by combining them in situ one by one without knowing the places of each piece ahead. Moreover, the initial studies for dry stacking with robots started with building a single stack tower and continued with

wall construction. In order to use such systems, planning and object detection algorithms are needed (Ko & Hsieh, 2023).

When robotics and design are encountered, parameterization can be integrated into the solution stage for controlling the design using various environmental and elemental features (Chiang et al., 2018). Since stacking with irregular elements may be unstable, the structural evaluation plan may need to be repeated each time an object is placed. The workflow of an example done by Wermelinger et al. (2018) can be seen in Figure 2.21. Apart from stacking elements, robotics can also contribute to the design process of the structure by generating positions that can hold the structure up (Paris et al., 2023). As an example, Menezes et al. (2021) use irregular blocks to have a dry-stacked wall with a system that is consistent with a model-free reinforcement learning (RL) approach.

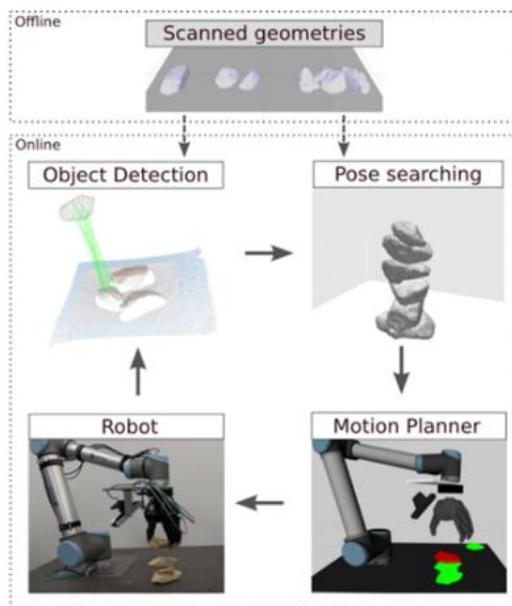


Figure 2.21 Tower Stacking Workflow (Wermelinger et al., 2018)

The workflow of another study by Zhou et al. (2023) for irregular object stacking with a tower-based approach on the sizes of the elements with a robot arm in sequential order can be seen in Figure 2.22. The method that they used investigated the loads and controlled the force to eliminate tilting, resulting in a successful

experiment of stacking four stones on top of each other. Lastly, a dissertation is done on robotic, dry stacking based on the heuristics of masons by Liu (2023), which works with the geometrical modeling of irregular stones and physics engines, along with real-life experiments with promising results.

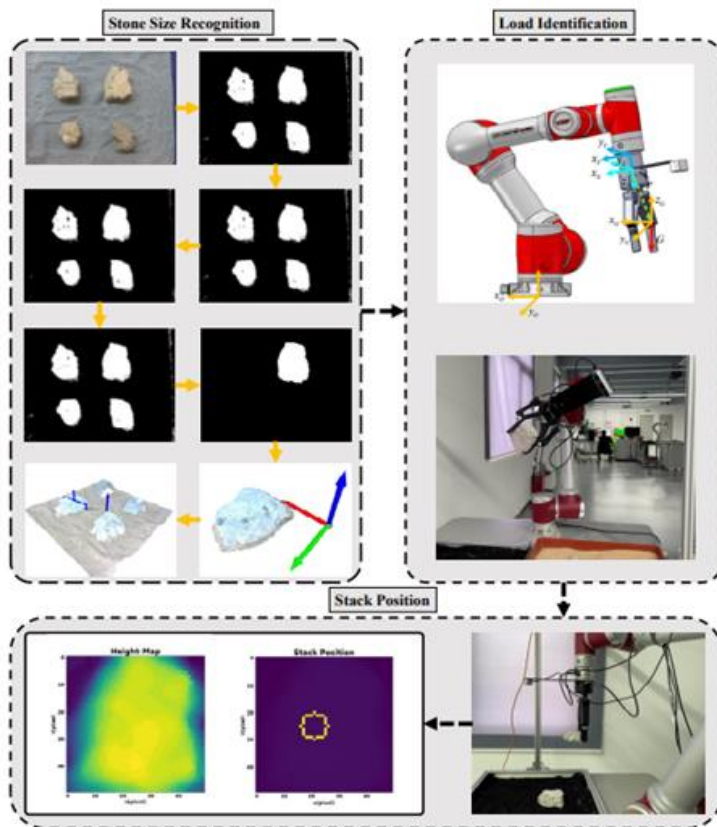


Figure 2.22 Irregular Object Stacking Based on Loads (Zhou et al., 2023)

2.9 Keyword Maps

An analysis of keywords from the existing literature is done through Vosviewer by extracting the information of the documents from Scopus. The subject areas are limited to Engineering, Computer Science, Arts and Humanities, and Materials Science, while the document types are limited to Article, Conference Paper, Book, and Book Chapter. Moreover, the number of co-occurrences is set to 5. The first pair of keywords is "Stereotomy" and "Robotics", which presents a limited number of

sources. Thus, the number of co-occurring keywords is too limited, with the number of 5, as it can be seen from the map in Figure 2.23. This study aims to contribute to the insufficient amount of literature.

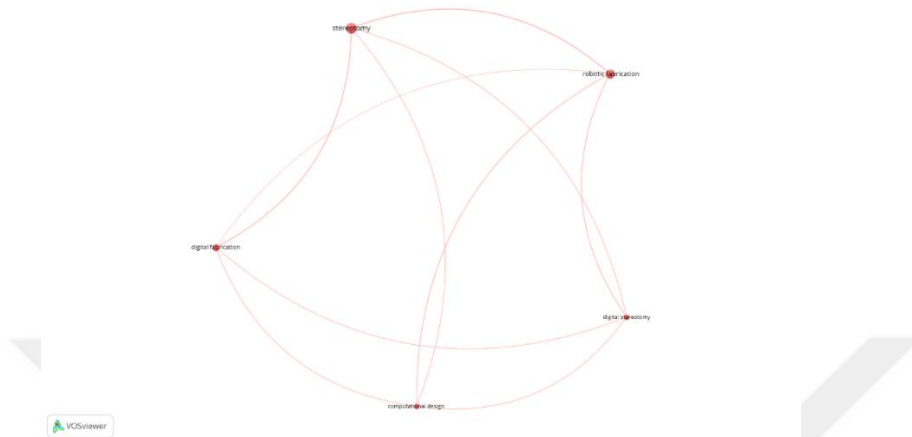


Figure 2.23 Keyword Map "Stereotomy" and "Robotics" 22.04.2025

The other pair of words is "Stereotomy" and "Masonry", which offers more sources in common, as it is expected that it also brings the topic of structural health monitoring related to the keyword of Masonry. Despite the increase in the number of articles, it is still not a sufficient number. Robotic Fabrication and Digital Fabrication are appearing through the connection with Stereotomy.

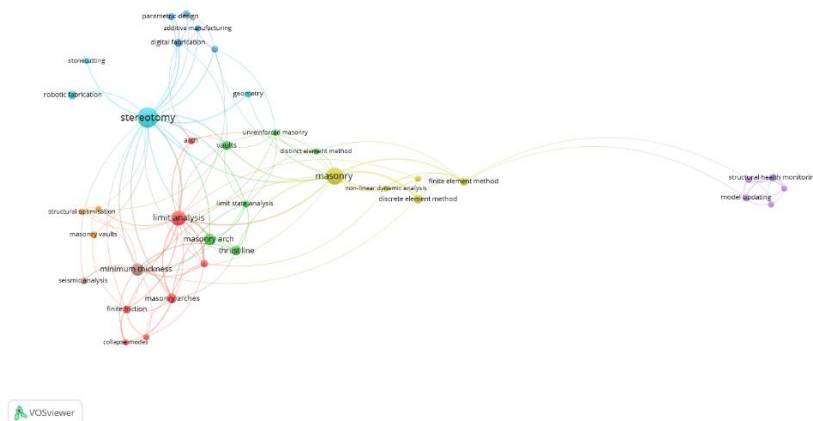


Figure 2.24 Keyword Map "Stereotomy" and "Masonry" 22.04.2025

The last pair is "Masonry" and "Machine Learning," which has the highest correlation due to the excessive amount of studies that merge these topics. All of the keywords are gathered together in the keyword map (Figure 2.29). Especially learning systems, masonry materials, walls, and deep learning are the foremost words according to the map. This progressive integration serves valuable source for this study.

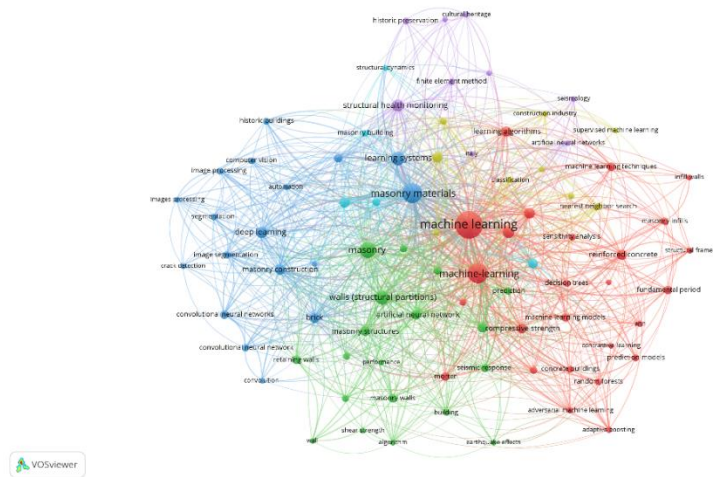


Figure 2.29 Keyword Map "Machine Learning" and "Masonry" 28.04.2025

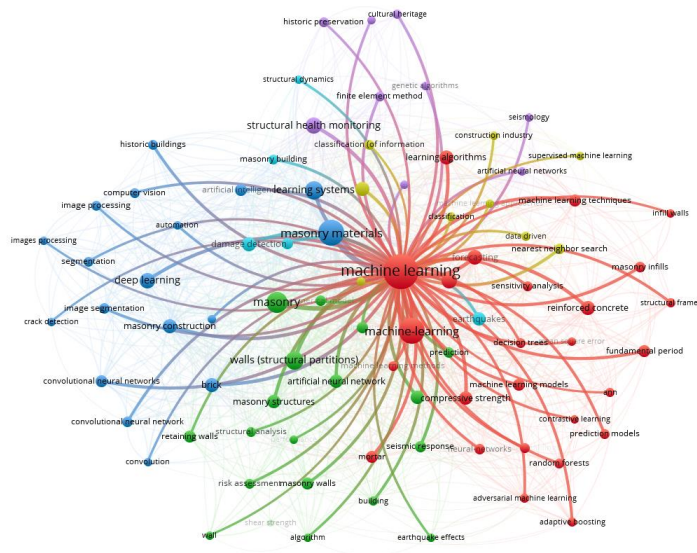


Figure 2.30 Keyword Map "Machine Learning" and "Masonry" - Focused on Machine Learning 28.04.2025

Forming a composition is viable through techniques that are valid from the past to the present. Stereotomy is one of them, and it is used with a variety of materials and even for reusing construction debris. The study of Cyclopean Cannibalism by Clifford & McGee (2018) deals with this type of problem, which is the most relevant study to the concern of this research. Computational methods are utilized in the study of Cyclopean Cannibalism, but unlike the approach of this work, AI is not used among these methods. Additionally, the works of stereotomy are done by way of cutting; nevertheless, it is possible to have such compositions without cutting the materials. The computational methods developed for reusing elements only focus on the material that is going to be used, while the existing patterns can be a guiding point to compose a structure. Moreover, a project called Mine the Scrap (Certain Measures, n.d.) utilizes the data from construction debris to create forms with artificial intelligence; on the other hand, the acquired outputs are art pieces.

Stacking is another technique that unifies discrete structural elements. There are studies that stack the units together, but there is no correlation built between techniques of stereotomy and stacking, although they usually focus on similar problems. Yet, stereotomy offers more complex forms considering its potential for building intricate structures. Both topics are highly correlated with masonry, which requires skill and precision. Several of the previously mentioned studies are tackling this concern by implementing robotics, but they are trying to deduce from masonry manuals that are written for humans. Hence, there is a need for a procedure that specializes in the agent itself, considering its constraints and advantages. Generally, the examples of stacking tend to put the elements with the larger sizes to the bottom, but this should not be the case for all the arrangements, since there can be more feasible solutions than that.

2.11 Contributions

The aim of this study is to propose a method for constituting a structure with discrete elements that could be both regular and irregular. Applying stereotomy to a structure

that will be generated is the general insight of this work. Machine Learning is the key point of this study to obtain options for patterns that are compatible with the set of elements that will be assembled. This pattern will be rooted in existing masonry wall patterns to increase constructability, as well as being a guiding component for robotic assembly.



CHAPTER 3

METHODOLOGY AND DATA PREPROCESSING

This study aims to build an approach that results in an end structure consisting of sub-elements. As the emergence point of the methodology, stereotomic bonds are selected as the way of composing the intended structure, as well as stacking them. Masonry walls are the most generic examples of these types of structures. Because of that, the masonry walls are decided to be used in the data collection part to comprehend the aggregation probabilities. Besides, stones as the module type provide an abundant amount of assemblage possibilities due to the endless variety of shapes. Since this study searches for ways of combining any kind of shapes, stone masonry patterns are very suitable for this task. After that, these patterns will be examined using AI models for classification to differentiate the examples from each other.

The features extracted from the images of masonry walls could have potential relations, and this interrelation makes this task suitable for tree-based models, which are in the scope of machine learning. On the other hand, since the initial data is images, deep learning can also be suitable for this task. Among the deep learning models, CNNs are the most compatible ones for this task, because the data is low-featured. Then, the most probable organization for a set of objects will be detected, and these will be placed precisely. This work should be done systematically and refined; thus, it is appropriate to be done by a robot arm. Computation and automation have become an important part of the construction technique for architectural studies of complex structures (Wei et al., 2023). Thus, this research contributes via these methods. Consequently, the robot arm should be utilized according to the structure and components that will be built.

3.1 Data Collection

Masonry walls are selected as an emerging point of the methodology, as it is mentioned earlier. In order to examine them, images of different types of masonry wall patterns that vary are gathered from the internet. These images are the basis of the proposed system, some of which can be seen in Figure 3.1. The first elimination of these images is done by subtracting the ones smaller than 256x256 pixels. After that, all of the images are set to 256x256 pixels. An OpenCV algorithm is produced for making edge detection of the stones. The general pseudo-code of this edge detection algorithm can be seen in Figure 3.2. The first step is reading the image. Then, the color space of the image is converted from BGR to grayscale. Adaptive thresholding is applied to the grayscale image, and its parameters are rearranged for each of them.



Figure 3.1 Examples of the Masonry Wall Images

```
1. Load the image and convert it to grayscale.
2. Apply adaptive thresholding to detect edges.
3. Find contours in the thresholded image.

4. Initialize variables:
   - 'min_contour_area'
   - 'approximation_accuracy'
   - 'stone_edges' (for drawing contours)

5. For each contour:
   - Calculate the area.
   - If the area is above the minimum:
     - Approximate the contour to a polygon.
     - Draw the contour on 'stone_edges'.

6. Display the original image and the image with detected edges.

7. Initialize lists for edge counts and contour lengths.

8. For each contour with sufficient area:
   - Calculate the length of each edge.
   - Print the edge lengths.

9. For each contour with sufficient area:
   - Calculate and store edge counts and contour lengths.
   - Draw the contours on 'stone_edges'.

10. Calculate and print:
    - Mean and standard deviation of the number of edges.
    - Mean and standard deviation of contour lengths.
```

Figure 3.2 Pseudo-Code of Edge Detection

Two different thresholding methods are used: Gaussian and Mean thresholds. Also, two different threshold types are used: binary and binary inverted. Block size is also determined, which specifies the size of a pixel neighborhood. It is used to calculate the threshold value for the pixel. The last parameter that is used is the constant value. Then, the minimum contour area is defined to eliminate the contours that can be negligible. Along with this parameter, the approximation accuracy value, which enables the reduction of complexity while preserving the overall shape, is also determined. The next step is to create an array of black pixels with the same dimensions as the image to penetrate the contours.

After that, a loop is created that iterates through contours. Inside this loop, the first step is to find the area of the contour and check whether it is larger than the minimum contour area set before. After that, the epsilon value, the approximation parameter, is calculated according to the approximation accuracy provided. In order to get a more detailed approximation, the epsilon value should be smaller. The contour approximation is done through the Douglas-Peucker (1973) algorithm, which is used to simplify a curve by decreasing the number of vertices with the use of the epsilon value. Then, the contour is drawn on the array that was previously produced. After arranging these images, the relatively successful ones are collected to create the main dataset. Examples of detected and plotted stone contours can be seen in Figure 3.3.

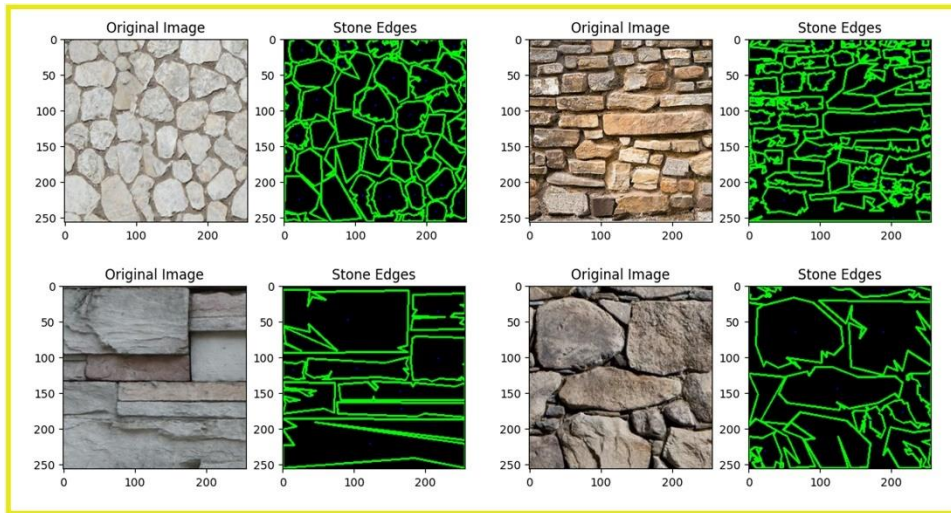


Figure 3.3 Examples of Stone Contours

For different kinds of classification models, various types of inputs are required. Decision tree and XGBoost models work with CSV file type, while the CNN model requires images themselves. For the tree-based models, a CSV file is composed by extracting different features. These features can be listed as Mortar/Dry, which is defined by the human eye, total contour number, the mean number of edges, the standard deviation of edges, the mean contour length, and the standard deviation of contour length, which are gathered from the contour lines extracted with OpenCV and the last feature is total neighborhood number which is collected via Grasshopper.

On the other hand, for the CNN model, images are arranged in a way so that the model can differentiate the patterns according to the corresponding cluster. The Ezdxf library is used for converting pixel data of the contours into a DXF file. The detected contour lines are transformed into vectors. After that, the center points of these closed contours are calculated. In order to save as a DXF file, a for loop is generated that iterates over vectors and center points, then adds vectors as LWPOLYLINE and center points as POINT entities into a new blank DXF file. The PIL library is used to create binary masks for each image. A blank white background is made, and each contour is filled with black by iterating LWPOLYLINE entities inside the DXF file. These images are saved as PNG files for CNN classification. For classifying each stone, the patterns are broken down into individual contours.

This process is done by placing each LWPOLYLINE entity into a separate file and extracting its vertices. These DXF files are also filled out using the previous method.

3.2 Creating the Configuration

A system for combining different structural elements can be learned from the existing patterns. Certain ML, DL, and computer vision (CV) approaches can be useful in this system. As a beginning point, the masonry wall patterns are selected as stated earlier. In order to learn these patterns, data preprocessing is needed. For this stage, OpenCV, which is a computer vision library, is used as it is mentioned to have promising results for understanding the existing examples. Outputs of this process can be classified into labels that are set. This classification can be done with different types of models. One option can be the tree-based models with features extracted from these images that are suitable for this task. On the other hand, CNN architecture is ideal for directly classifying images. For the following steps of this study, the clusters of each stone are also investigated as an unsupervised approach, rather than only classifying the patterns.



CHAPTER 4

TRAININGS OF MACHINE LEARNING MODELS

4.1 Classification with Tree-Based Machine Learning Models

Two different models, decision tree and extreme gradient boosting (XGBoost), are used for this classification. Varying sizes and shapes of stones can be observed through the dataset. The initial trial is done with the first 195 images to be classified as a binary classification that defines whether the pattern is close-packed or not. Before applying the main classification approach, different labeling methods were tried on the different portions of the dataset, and then the amount of data was increased. The extracted features mentioned earlier are accumulated in a CSV file to be used for training. The correlation between these features is examined through data visualization done by using Matplotlib pyplot and Seaborn libraries.

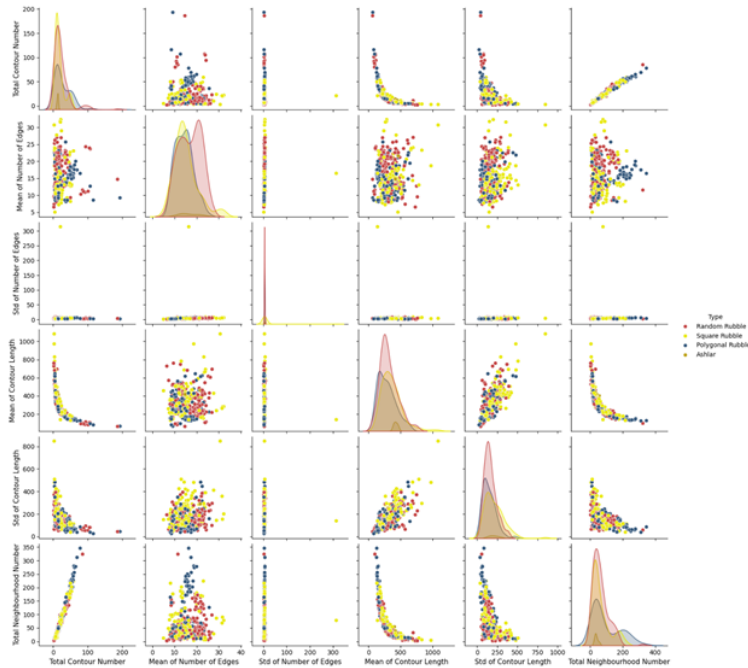


Figure 4.1 Pair Plotting

The extracted features have diverse types of correlations with each other that can be seen in the pair plotting in Figure 4.1. When these relations are examined, several of them have linear relationships as expected, like between the total contour number and the total neighborhood number. Also, the mean contour length decreases while the total contour number increases. A similar relation prevails between the mean contour length and the neighborhood number, as expected.

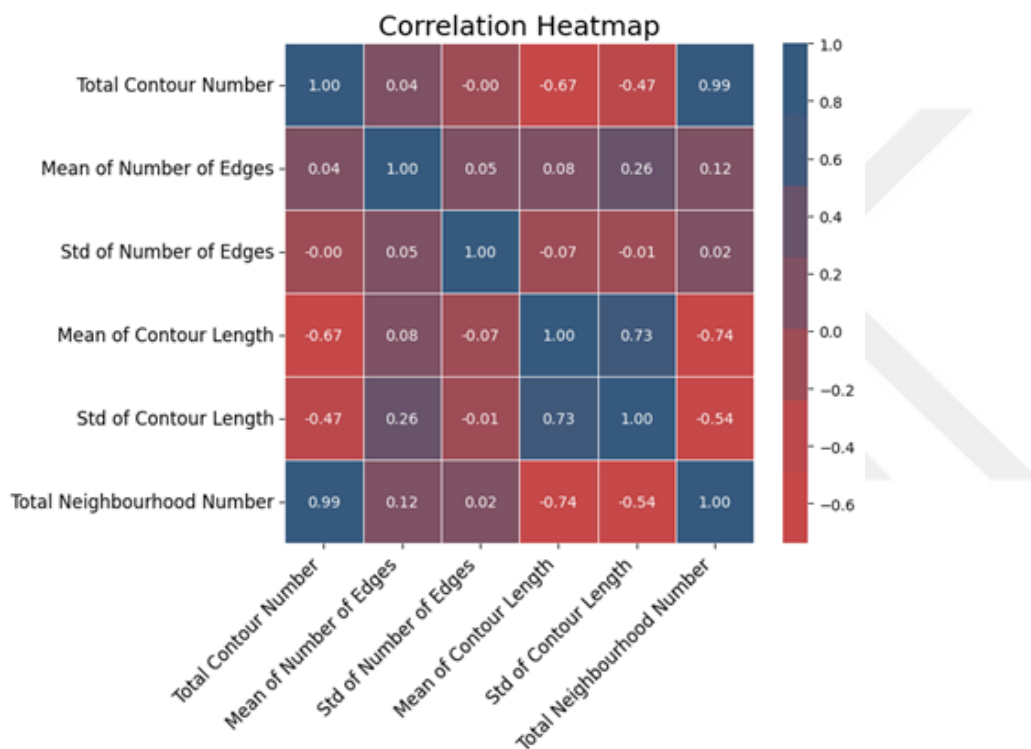


Figure 4.2 Correlation Heatmap

Along with the pair-plotting, the correlation heatmap in Figure 4.2 also shows the relationship between the features. This heatmap is associated with the pair plot. There are strong positive and negative relations as well as some weak relations. Strong relations are beneficial for the model to learn better, which can be listed as total contour number–total neighborhood number, mean of the contour length – std of the contour length, mean contour length–total contour number, and mean contour

length-total neighborhood number. The first two have positive relations, while the other two have negative relations.

4.1.1 Decision Tree Model

It is a basic tree model that starts with a root node and branches out according to the if/else or yes/no questions. It concludes with the leaf nodes that give the output. Division of the data according to the branches is done in line with a metric like Gini Impurity or Entropy. Both regressor and classifier models are used in this machine-learning method, while the classifier model is used in this study. The Scikit-learn library is used to build the principal model, and a model is built from scratch to be able to change the root question. The dataset is feasible for using a decision tree model. The decision tree model is appropriate for visualization. The tree model can show each node with the feature, the splitting criteria, and the metric that is used. The size of the tree can be controlled with the maximum depth parameter.

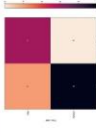
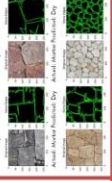
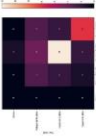
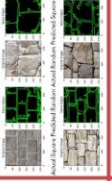
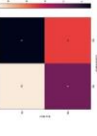
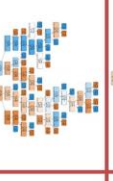
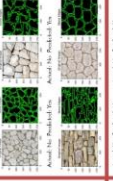
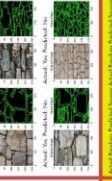
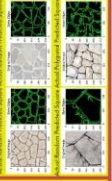
Different types of labeling are applied to see the capability of the data. 107 rows of the data are used for the initial trial. This trial is done with the labeling Mortar/Dry, which is a binary classification. The training test split ratio is set to 90% and 10% due to the small amount of data. The following trials are performed with 195 rows of data. Additionally, the train test split is set to 80% and 20%. The last trials done with the model from scratch were done with 398 rows of data, with the same train-test split. The second trial is done with multiclass labeling. The classes are ashlar, polygonal, square, and random rubble. The third trial has binary yes/no labeling that classifies whether the pattern is close-packed. The following trials are done with the model from scratch, which also uses the yes/no labels like the previous one. The final model, which is employed during the implementation part, is done with the labeling of polygonal, square, and random rubble that are valid labels since they are mentioned in masonry sources (Szabó, Funari, & Lourenço, 2023) using the scikit-learn library, which is reliable due to its identifiability. The scikit-learn library is also used to make one-hot encoding for the categorical features.

4.1.2 Results of the Decision Tree Model

Five different trials are done with the decision tree of the scikit-learn library. The overall results which comprise the most effective model that is the last one can be seen in Table 4.1 containing information on the labeling strategy, number of rows in the dataset, the measure of randomness, train-test split ratio, accuracy of the model, confusion matrix, output decision tree, and some examples of the false predictions.



Table 4.1 Results of the Scikit-Learn Decision Tree Model

	Labeling	Number of Rows	Measure of Randomness	Train/Test Split	Accuracy	Confusion Matrix	Decision Tree	Examples of False Predictions
Trial 1	Mortar/Dry	195	Gini Impurity	90/10	0.82			
Trial 2	Type (4 Class)	398	Gini Impurity	80/20	0.62			
Trial 3	Close Packed/Not	398	Gini Impurity	80/20	0.67			
Trial 4	Close Packed/Not	398	Entropy	80/20	0.72			
Trial 5	Type (3 Class)	393	Gini Impurity	70/30	0.66			

The first model reaches the accuracy level of 0.82, which seems promising, but since the data is too narrow and the percentage of the test set is too low, the result is not very reliable. It can be inferred from the confusion matrix that the features are appropriate for classifying this data, despite the limited size of the test data, which is 11. The model predicts the patterns that have gaps as mortar, even if they are dry. Similarly, when the amount of mortar is low, the stones are close to each other, and the prediction tends to be dry.

The accuracy score of the second model reached 0.62, which is not quite good. Thus, it can be said that the model could not learn sufficiently, as it can also be perceived from the confusion matrix, and the rise in the number of labels makes the tree structure more complex. The imbalance in the data for each class may cause these insufficient results. For this labeling strategy, more data is required to be distributed precisely. The model makes the most of the false predictions by predicting them as random. The ashlar type could not be predicted, especially since there are few examples of it. The main reason that the machine could not learn sufficiently is the data imbalance.

The accuracy is 0.67 in the third trial of the decision tree model that uses Gini impurity as the metric. Slightly better results are achieved with this trial, which indicates that the model learned moderately better. Another labeling that defines whether the pattern is close-packed is employed for this model. The decision tree structure is more organized than the previous one. Better results are enabled with the influence of more balanced data distribution. According to the false prediction examples of this model, the main problem with this model making false predictions is that some of the contour lines are drawn improperly, so that a few of the close-packed ones are drawn with gaps. This may be misleading for the model to predict, but the other possible errors are not that certain.

The accuracy increased by 0.05 and became 0.72 for the fourth model with the same labeling strategy as the previous one, yet the measure of randomness is changed to

Entropy from Gini impurity. Therefore, it can be observed that Entropy facilitates a more suitable split at the nodes for this model. The prediction challenge of this model is like the previous one, despite the better accuracy.

The accuracy drops to 0.66 for the last model, but the labeling strategy, which is random, polygonal, and square rubble, is more reliable since it is supported by the literature. A similar approach is used in the second model, but due to the insufficient amount of data in the ashlar type, they are eliminated, which results in more consistency in the data. Moreover, the train-test split is changed to 70-30 to get a more dependable model. This model will be the base for this research, as the further process depends on the classification result by proceeding to the relevant patterns. In order to understand the model properly, the classification report, confusion matrix, decision tree, and some examples of both false and true predictions of the model are provided in the following table and figures.

Table 4.2 Classification Report of the 5th Decision Tree Model

	precision	recall	f1-score	support
Polygonal Rubble	0.67	0.65	0.66	34
Random Rubble	0.61	0.72	0.66	43
Square Rubble	0.74	0.61	0.67	41
accuracy			0.66	118
macro avg	0.67	0.66	0.66	118
weighted avg	0.67	0.66	0.66	118

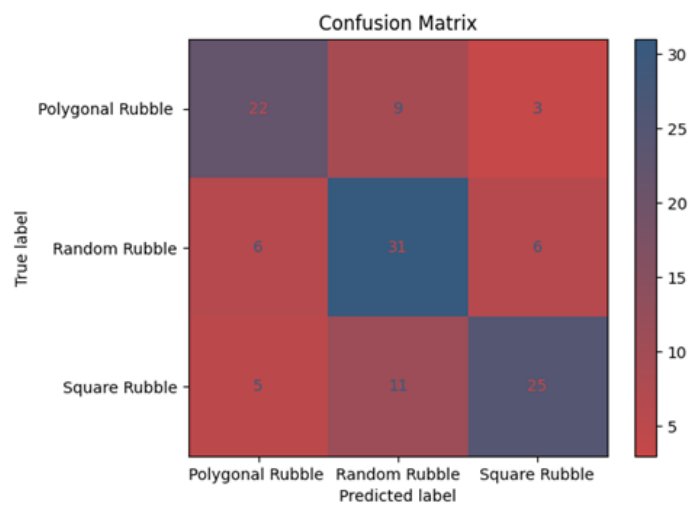


Figure 4.3 Confusion Matrix of 5th Decision Tree Model

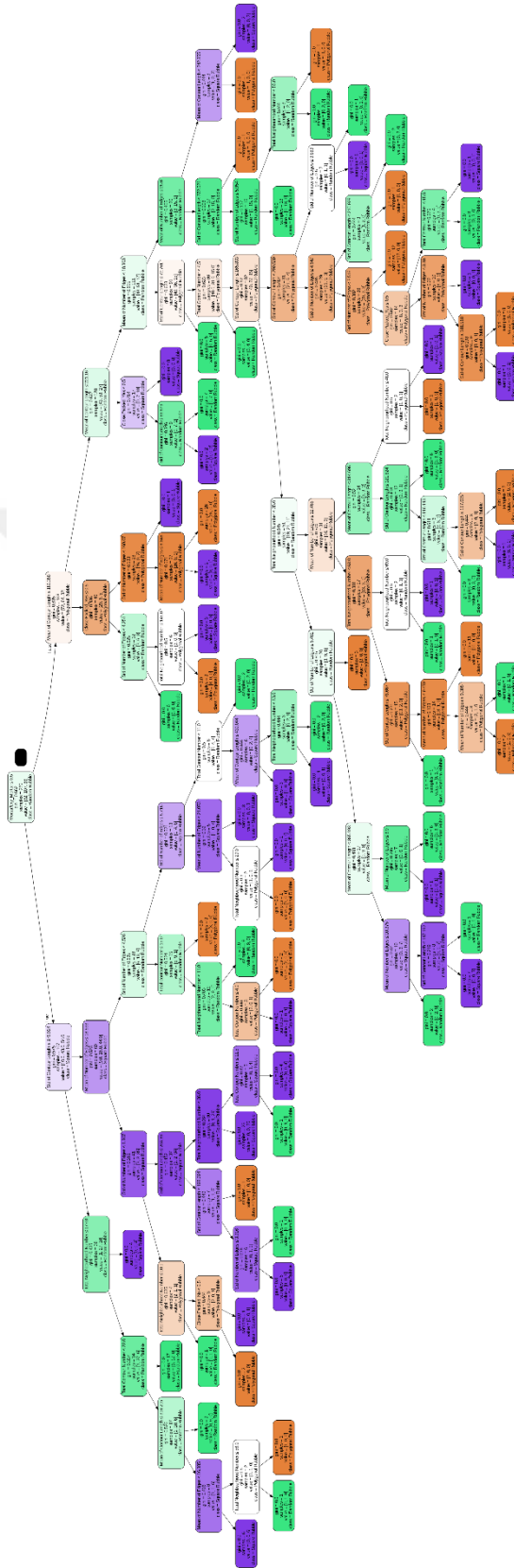


Figure 4.4 Structure of the 5th Decision Tree Model

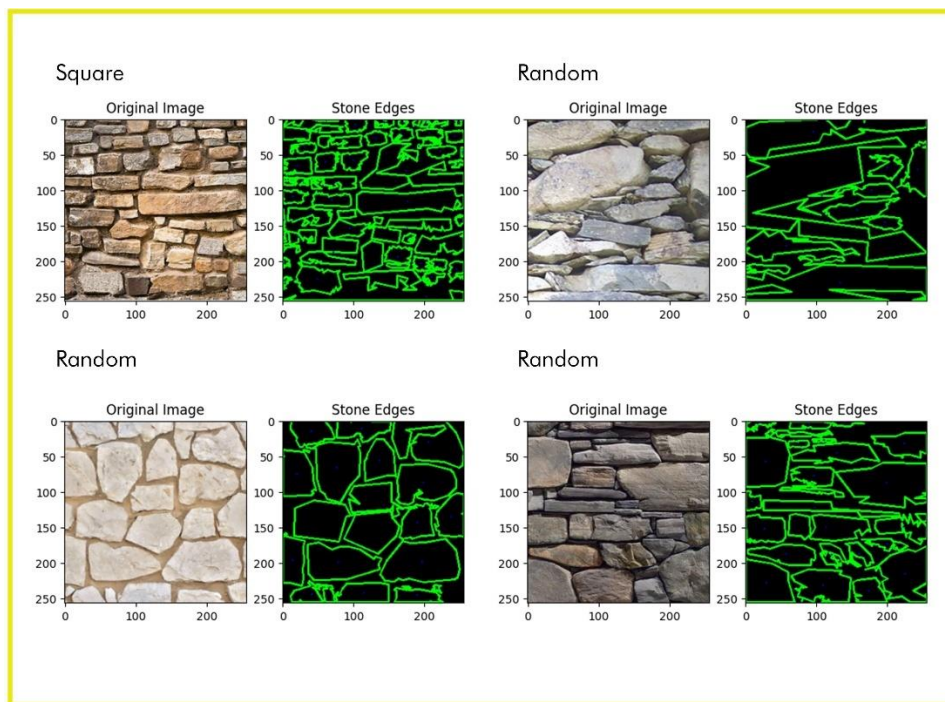


Figure 4.5 True Predictions

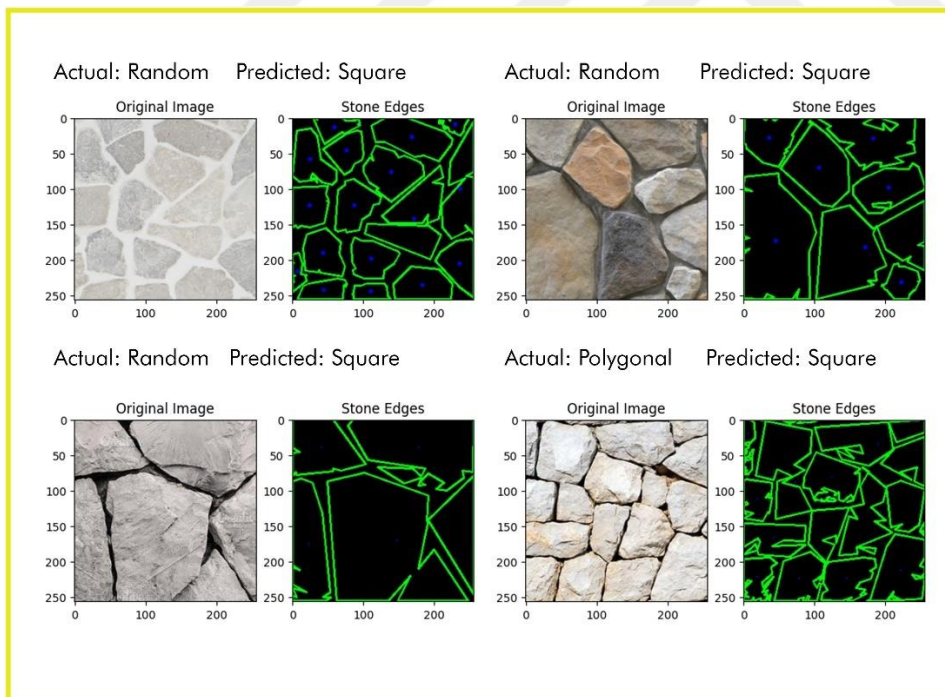


Figure 4.6 False Predictions

In order to understand the effects of the features on the stage of building the tree, a decision tree from scratch is built, since the root question can not be changed in the Scikit-Learn library. 6 different features are tested as the root question of the tree. The training and the test splits are set to 70-30 for all of these trials that use the main labeling strategy, which consists of the polygonal rubble, random rubble, and square rubble labels. The results can be seen in the table below.



Table 4.3 Results of the Custom Decision Tree

	Labeling	Number of Rows	Measure of Randomness	Train/Test Split	Root Question	Accuracy	Confusion Matrix	Decision Tree	Examples of False Predictions
Trial 1	Type (3 Class)	392	Gini Impurity	70/30	Total Contour Number	0.67			
Trial 2	Type (3 Class)	392	Gini Impurity	70/30	Mean Number of Edges	0.59			
Trial 3	Type (3 Class)	392	Gini Impurity	70/30	Std of Number of Edges	0.38			
Trial 4	Type (3 Class)	392	Gini Impurity	70/30	Mean of Contour Length	0.66			
Trial 5	Type (3 Class)	392	Gini Impurity	70/30	Std of Contour Length	0.62			
Trial 6	Type (3 Class)	392	Gini Impurity	70/30	Mean of Contour Length	0.67			

According to the results, it can be inferred that the total contour number and the mean contour length are the most appropriate features for the root question with an accuracy level of 0.66 for both while the standard deviation of the number of the edges is the most infeasible feature since there is no even any division after the root. Although it gives insight into the features, the custom decision tree model is not as dependable as the decision tree of the scikit-learn library, which is a known source, since it doesn't have recognition.

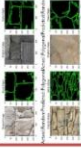
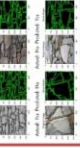
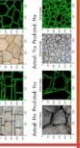
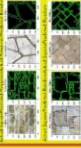
4.1.3 Extreme Gradient Boosting Model

XGBoost is a conventional ML method that consists of multiple trees at the same time, which differentiates it from a single decision tree. Trees are put together sequentially. As a result of this sequence, each learning influences the following ones; hence, ensemble learning is provided with this model. Multiple CPU cores are used for training due to the parallelization feature. Optimization of the model can be done by arranging different parameters like learning rate, estimator number, and maximum depth. Also, overfitting can be avoided with the application of early stopping. One 195 rows of data are used for the first 4 XGBoost models, while the last one is trained with 393 rows. Data is split into 80% and 20% ratios as a train and test for the initial 4 models, and it is 70% to 30% for the last model. A multiclass labeling strategy with 4 classes was applied for the first trial, the other three used binary labeling, and the last model adopted multiclass labeling with 3 classes. For the multiclass labeling, the labels are ashlar, polygonal, square, and random rubble. Besides, for the binary labeling, the labels are yes and no for detecting the close-packed patterns.

4.1.4 Results of the Extreme Gradient Boosting Model

Four trials were done with the XGBoost model training, and the overall results of these models can be seen in the table below.

Table 4.4 Results of the Extreme Gradient Boosting

	Labeling	Number of Rows	Booster	Train/Test Split	Number of Estimator/ Early Stopping	Dropout	Learning Rate	Accuracy	Confusion Matrix	Examples of False Predictions
Trial 1	Type (4 Class)	195	GBTree	80/20	100	No	0.3	0.60		
Trial 2	Close Packed/Not	195	GBTree	80/20	100	No	0.3	0.74		
Trial 3	Close Packed/Not	195	DART	80/20	5000/1771	Yes	0.01	0.79		
Trial 4	Close Packed/Not	195	GBTree	80/20	5000/3997	Yes	0.005	0.87		
Trial 5	Type (3 Class)	393	GBTree	70/30	3600/490	No	0.01	0.72		

The accuracy score is 0.60 in the first trial of the xgboost model used for multiclass classification. Since the estimator number parameter is not used, it runs with the default value that is 100. The metrics are even lower than the same classification as the decision tree model. Yet, the reason for unsuccessful results is the same as the decision tree, which is the data imbalance. Thus, because of this imbalance, the model fails to learn, and it predicts most of them as the label with the most data, which is random. The default parameters are used for the second trial of the xgboost model, which makes binary classification. The accuracy is 0.74 for this model, which uses the default values of Gbtree booster and 100 trees, which means that this labeling strategy enables better learning with relatively higher results.

In the third trial of the xgboost model, the parameters are tuned using DART booster, application of dropout, 0.01 learning rate, initial 5000 estimators, and early stopping. Early stopping interrupted learning at the 1771st estimator to prevent overfitting, resulting in an accuracy of 0.79, which is reasonably better than the previous one. Nevertheless, the yes label is hard to predict in this model, as well as in the previous model. Like the decision tree model for this binary classification, it can be understood from the results that most of the false predictions are made according to the misleading contour lines. The model could learn better with more proper lines inferred from the false predictions.

A 0.87 accuracy level is achieved in the fourth trial of the XGBoost model. GBtree, which is the default booster, is used in this model along with the 0.005 learning rate. Training is terminated at the 3997th iteration due to the early stopping, whose round is set to 50. As the last parameter, the number of parallel trees is set to 18. Decreasing the learning rate increases the capability of learning. In addition, the feature of the total neighborhood number has some NaN values. These are caused by an error in the Grasshopper algorithm that occurs when the number of contours is too high. In the previous trials, these ones were filled with the maximum value. However, since the XGBoost model allows Nan values, these are left as they were in the last trial. The increment in that accuracy can be caused by this strategy. Compared to the other

models, the number of false predictions is lower in this model, thanks to its high performance.

The last trial is done with 3 classes of polygonal, random, and square rubble, which are settled for the research. The model reached an accuracy of 0.72 with the parameters of Gbtree booster with 0.01 learning rate, 3600 estimators with early stopping rounds of 10, which stopped at 490th estimator, and the number of parallel trees set to 20. The general performance of the multiclass classifications is under the binary classification, but the labels used are more dependable, so these will be the base models for finding a suitable pattern for production. The classification report, confusion matrix, and several examples of true and false predictions of this last model can be seen below.

Table 4.5 Classification Report of 4th XGBoost Model

	precision	recall	f1-score	support
polygonal	0.74	0.68	0.71	34
random	0.65	0.81	0.72	43
square	0.82	0.66	0.73	41
accuracy			0.72	118
macro avg	0.74	0.72	0.72	118
weighted avg	0.73	0.72	0.72	118

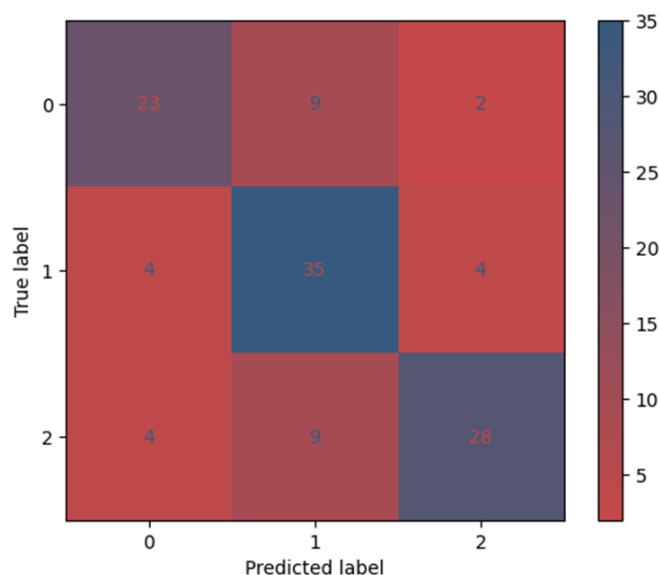


Figure 4.7 Confusion Matrix of 4th XGBoost Model

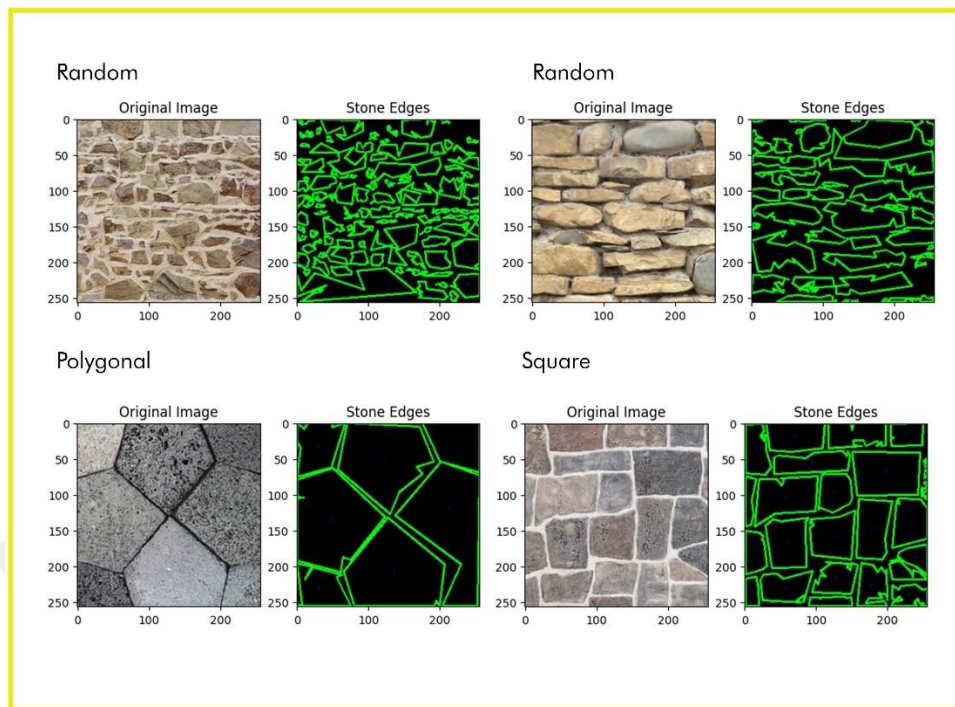


Figure 4.8 True Predictions

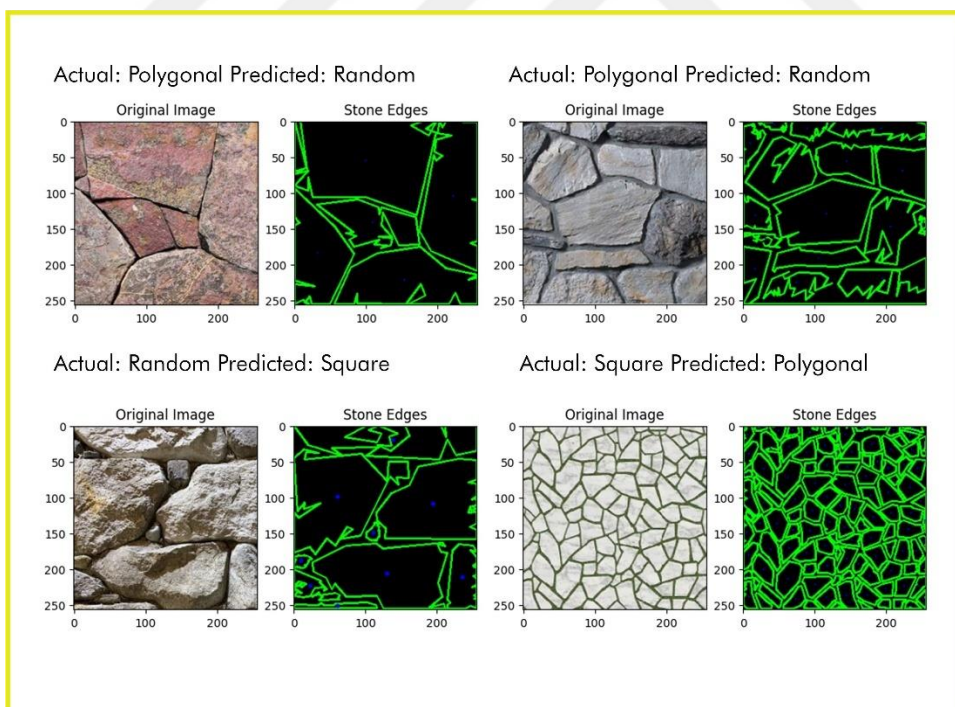


Figure 4.9 False Predictions

Also, the log loss and receiver operating curves (ROC) of the model are gathered to evaluate the model, which can be seen in the following figures. As can be seen in the table, the log loss of both test and training is decreasing as expected. When the log loss of the test set starts to increase slightly, the training has stopped thanks to the early stopping. The ROC curve and area under the curve (AUC) number reflect the prediction ability of the model, which is calculated for each class for the multiclass classification models. When it is close to 1, it means the prediction ability of the model is high. According to that, since the numbers of this model are 0.85, 0.84, and 0.86 for each class, it can be said that this model has the ability to predict. The most accurately predictable class is square rubble.

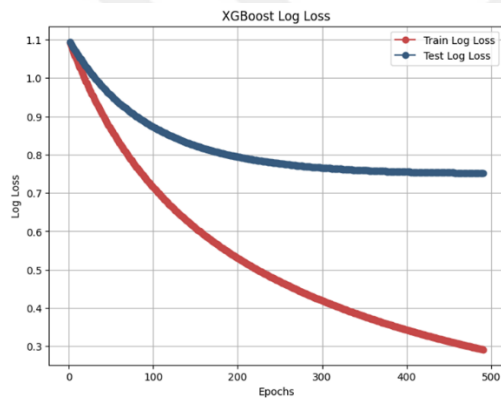


Figure 4.10 Log Loss of the 4th XGBoost Model.

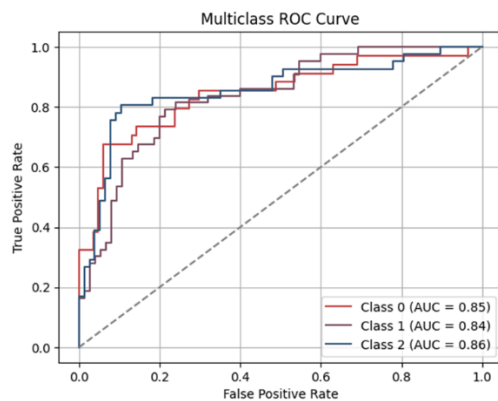


Figure 4.11 Receiver Operating Curve of 4th XGBoost Model

4.2 Classification with Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are deep learning models that consist of different layers that execute tensor operations and facilitate the numbers inside these tensors to be converted into a vector. Among different types of layers, the convolution layer is the most important one, whose working mechanism is illustrated in Figure 4.12 with an image size of 3×4 and a convolution kernel size of 2×2 . A wide range of tasks can be done via CNNs, in particular with image operations containing object detection, semantic segmentation, and image classification (Wu, 2017). Image classification with CNN models tasks will be performed within the scope of this research.

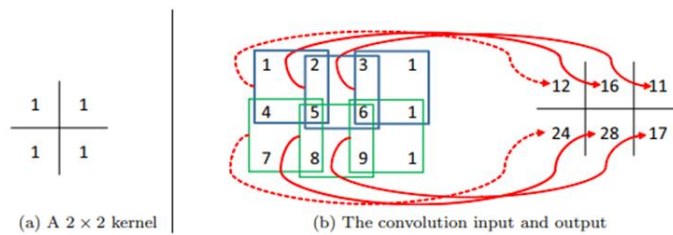
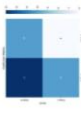
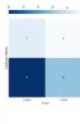
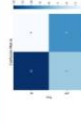
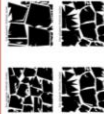
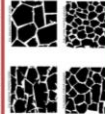


Figure 4.12 Illustration of the Convolutional Operation (Wu, 2017)

It is used for two different tasks with different labeling methods. The first task is to evaluate the whole pattern by defining whether it is close-packed or not, and the main labeling of polygonal, random, and square rubble. The second task is to classify the stones individually with the main labeling strategy. In order to reach higher results, several trials are done via CNNs with a different number of layers. The shapes of the stones and the ratio of the gaps are emphasized through the black-and-white images produced by the explained steps. The stones are black, and the gaps between them are white. The amount of data is increased throughout the trials. The table consists of the CNN results for classifying the patterns that can be seen below.

Table 4.6 Results of Pattern Classification with CNN

	Labeling	Number of Rows	Activation Function	Train/Test Split	Type of CNN	Dropout	Accuracy	Confusion Matrix	Examples of False Predictions
Trial 1	Close Packed/Not	195	Softmax	80/20	Custom with 10 layers	No	0.49		
Trial 2	Close Packed/Not	195	Sigmoid	80/20	Custom with 10 layers	No	0.56		
Trial 3	Close Packed/Not	195	Sigmoid	80/20	Custom with 7 layers	No	0.66		
Trial 4	Close Packed/Not	195	Sigmoid	80/20	Custom with 9 layers	Yes/0.1	0.69		
Trial 5	Type (3 Class)	281	Softmax	80/20	Custom with 11 layers	No	0.72		
Trial 6	Type (3 Class)	385	Softmax	80/20	Custom with 11 layers	No	0.72		
Trial 7	Type (3 Class)	393	Softmax	70/30	Pretrained VGG-16	Yes/0.5	0.79		

The first CNN model for binary classification, which consists of 10 layers, has an accuracy of 0.49 with a SoftMax activation function. As a result of the imbalance between yes and no labels, the model fails to predict the yes label. The activation function is changed to Sigmoid, while the rest of the CNN architecture remains the same for the second trial of the binary classification. As a result, the accuracy increased to 0.56 after this revision. Like the first trial, the yes label is also hard to predict in this model.

A simpler architecture with 7 layers is used in the third trial of binary classification, which achieved an accuracy level of 0.66. It can be seen from the confusion matrix that the issue with predicting the yes label continues, yet the overall scores are better. After several training sessions that had lower accuracy, the fourth training in the table was done with a different model architecture consisting of 9 layers, also it uses dropout with a 0.1 rate, different from the previous examples. Thus, the highest accuracy for the binary classification is reached with this model, which is 0.69. Unlike the previous models, the no label is hard to predict for this one.

Furthermore, the multiclass classification with three classes of polygonal, random, and square rubble is performed using CNN. The first trial is trained with 281 images. The model architecture changes to 11 layers with a SoftMax activation function, and a 0.72 accuracy level is achieved after the training of 18 epochs. It can be seen from the false predictions that the jagged lines in some images are causing the model to fail. Another expansion of the data is done as the number of images is increased to 385. The same model with early stopping is trained with the increased dataset. Training stopped at the 21st epoch. On the other hand, the accuracy level did not change.

For the last and the base model for this study, a pre-trained VGG-16 model is used, and the architecture of the model can be seen in Table 4.7 (Zhang et al., 2016). This model was developed by Simonyan & Zisserman (2015) and trained with the ImageNet dataset, which consists of images of 1.3 million for training, 50k for

validation, and 100k for testing, belonging to 1000 classes. The last version of this dataset was made by Russakovsky et al. (2015).

Table 4.7 Architecture of the VGG-16 Model (Zhang et al., 2016)

layer	filter size	# channels	# filters	stride	output size	complexity (%)	# of zeros
Conv1 ₁	3 × 3	3	64	1	224 × 224	0.6	0.48
Conv1 ₂	3 × 3	64	64	1	224 × 224	12.0	0.32
Pool1	3 × 3			2	112 × 112		
Conv2 ₁	3 × 3	64	128	1	112 × 112	6.0	0.35
Conv2 ₂	3 × 3	128	128	1	112 × 112	12.0	0.52
Pool2	2 × 2			2	56 × 56		
Conv3 ₁	3 × 3	128	256	1	56 × 56	6.0	0.48
Conv3 ₂	3 × 3	256	256	1	56 × 56	12.1	0.48
Conv3 ₃	3 × 3	256	256	1	56 × 56	12.1	0.70
Pool3	2 × 2			2	28 × 28		
Conv4 ₁	3 × 3	256	512	1	28 × 28	6.0	0.65
Conv4 ₂	3 × 3	512	512	1	28 × 28	12.1	0.70
Conv4 ₃	3 × 3	512	512	1	28 × 28	12.1	0.87
Pool4	2 × 2			2	14 × 14		
Conv5 ₁	3 × 3	512	512	1	14 × 14	3.0	0.76
Conv5 ₂	3 × 3	512	512	1	14 × 14	3.0	0.80
Conv5 ₃	3 × 3	512	512	1	14 × 14	3.0	0.93

Since the input image format of the VGG-16 model is with the size of 224x224 and RGB channel, the images are preprocessed by resizing and duplicating them to have three channels. After that, the pre-trained VGG-16 model is downloaded from Tensorflow Keras. Considering the classes are different from the ImageNet dataset, the top fully connected layers of the model are excluded and replaced according to the number of labels that are used in this study, which is three. The model is trained to extract the features in line with the new classification by freezing the pre-trained layers. The training continued for 10 epochs and reached a 0.79 accuracy level. The classification reports the confusion matrix, and examples of true and false predictions of the model can be seen in the following table and figures.

Table 4.8 Classification Report of Pre-trained VGG-16 Model for Pattern Classification

	precision	recall	f1-score	support
polygonal	0.74	0.92	0.82	25
random	0.71	0.84	0.77	44
square	0.97	0.68	0.80	47
accuracy			0.79	116
macro avg	0.81	0.81	0.80	116
weighted avg	0.82	0.79	0.79	116

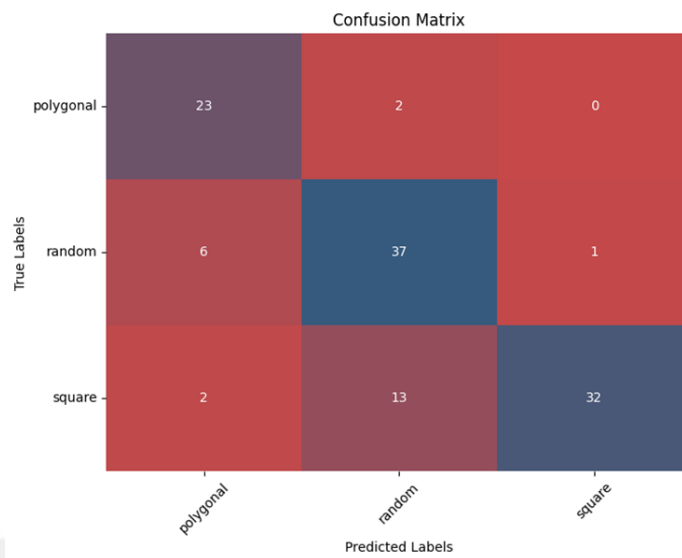


Figure 4.13 Confusion Matrix of the Pre-trained VGG-16 Model for Pattern Classification

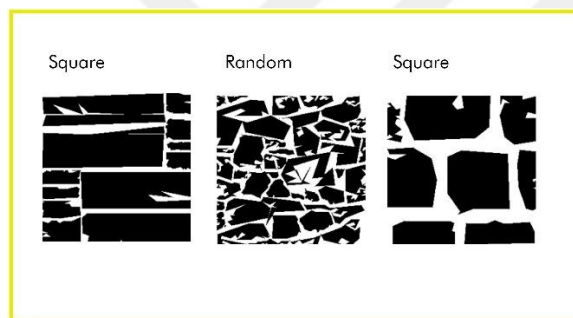


Figure 4.14 True Predictions

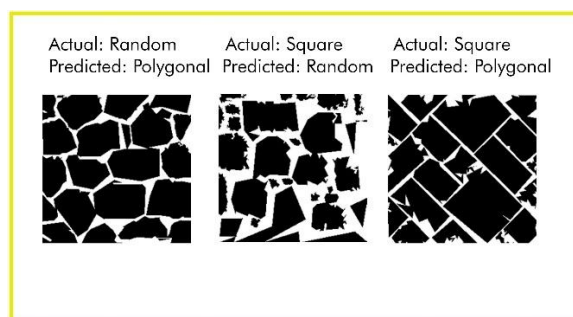


Figure 4.15 False Predictions

It can be inferred from both the confusion matrix and the ROC curve that the model is making the most successful predictions for the polygonal rubble label, while it is

making the worst predictions for the random rubble label. Most of the false predictions are the square rubbles that are predicted as random rubble.

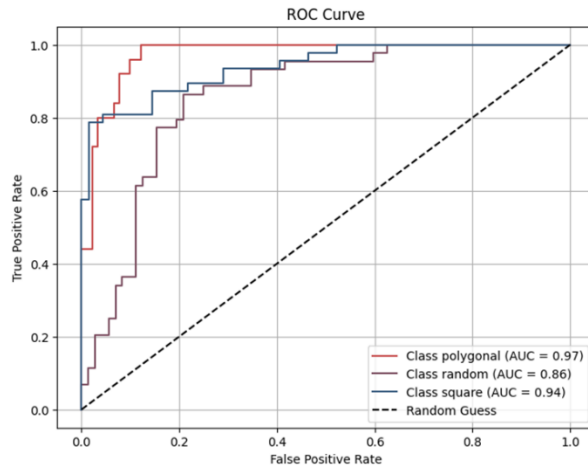


Figure 4.16 Receiver Operating Characteristics Curve of Pre-trained VGG-16 Model for Pattern Classification

The training loss is decreasing as expected, but the validation loss has a slight bump. However, it is still sufficient, due to the overall decrease, as can be seen in the following graph.

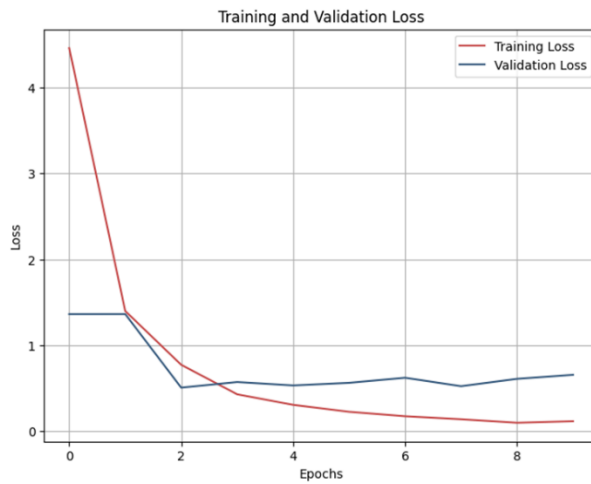


Figure 4.17 Training and Validation Loss of Pre-trained VGG-16 Model for Pattern Classification

The second task with CNN is to make the multiclass classification of the stones individually with the same labels. For this task, the patterns are broken down into

individual pieces one by one. Through this classification, the role of individual pieces in a pattern will be comprehended. The table below shows the results of this classification.

Table 4.9 Results of Individual Stone Classification with CNN

	Labeling	Number of Rows	Activation Function	Train/Test Split	Number of Layers	Dropout	Accuracy	Confusion Matrix	Examples of False Predictions
Trial 1	Type (3 Class)	5500	Softmax	80/20	Custom with 11 layers	No	0.56		
Trial 2	Type (3 Class)	6023	Softmax	80/20	Custom with 11 layers	No	0.63		
Trial 2	Type (3 Class)	6309	Softmax	70/30	Pretrained VGG-16	Yes/0.5	0.63		

The first training is done with 5500 images with the last custom CNN that was used in the previous classification. The level of accuracy is 0.56 for this model. Since the images are low featured, the accuracy rate dropped drastically. Another reason for that is that one stone can be part of different kinds of patterns. The second trial of the multiclass classification of individual stones is done with the same model, but this time, the data points are increased to 6023. After the data expansion, the accuracy increased to 0.63.

The last training is done with the pre-trained VGG-16 model. The model stopped training, which was done with 6309 images, at the 13th epoch due to early stopping and reaching 0.63 accuracy as well. Nevertheless, the credibility of the VGG-16 model makes this model the basis of this research. The classification report, confusion matrix, examples of true and false predictions, and the loss graph are given in the following figures.

Table 4.10 Classification Report of Pre-trained VGG-16 Model for Individual Stone Classification

	precision	recall	f1-score	support
polygonal	0.66	0.72	0.69	657
random	0.57	0.51	0.54	615
square	0.67	0.66	0.66	621
accuracy			0.63	1893
macro avg	0.63	0.63	0.63	1893
weighted avg	0.63	0.63	0.63	1893

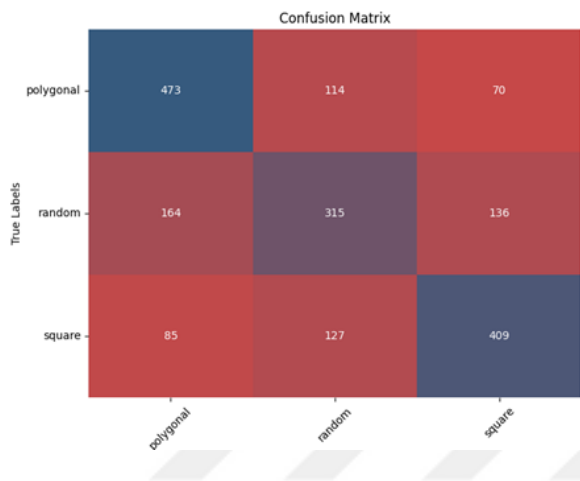


Figure 4.18 Confusion Matrix of the Pre-trained VGG-16 Model for Individual Stone Classification

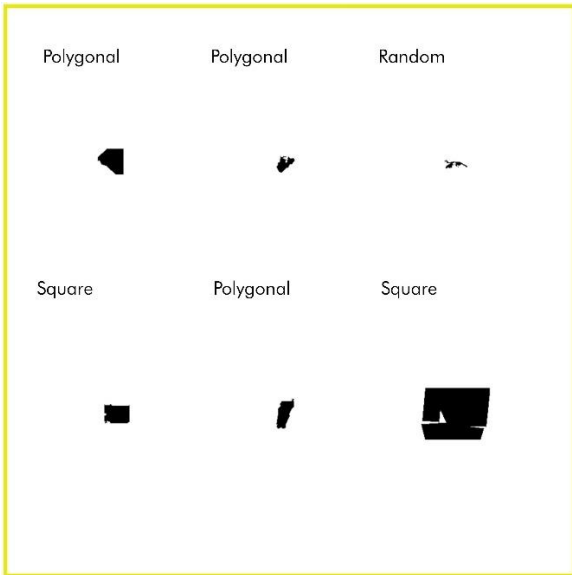


Figure 4.19 True Predictions

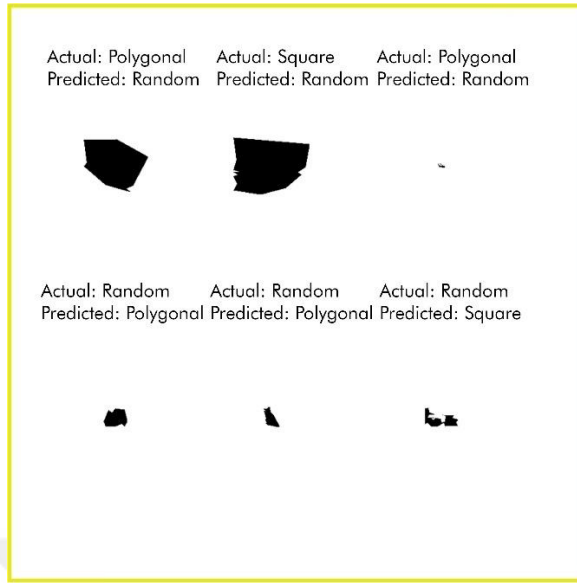


Figure 4.20 False Predictions

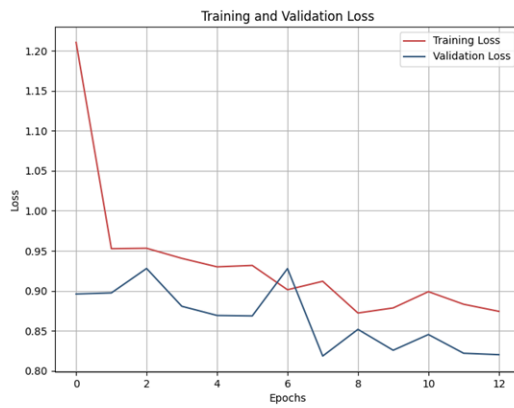


Figure 4.21 Training and Validation Loss of Pre-trained VGG-16 Model for Individual Stone Classification

4.3 K-Means Clustering

Following the supervised models, an unsupervised approach is adopted to comprehend the relations within the gathered patterns properly by clustering them according to the previously extracted features. K-means clustering, that is proposed by MacQueen (1967), is decided to be used for this task, which could work with the

extracted features. Aside from the features used for the decision tree, the labels, which are polygonal random and square rubble, are also used as a feature for the K-means model to have clusters that contemplate their types. K-means is a model that creates clusters of data based on the proximity among the features.

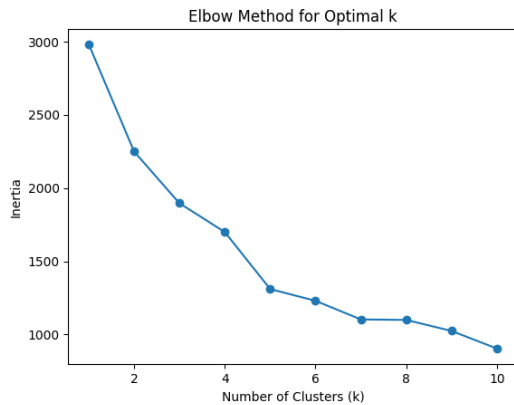


Figure 4.22 Elbow Method for Optimal K

The number of clusters is detected as 3 according to their inertia values, which is referred to as the elbow method. A training pipeline is formed to be able to incorporate both categorical and numerical features. The clusters that are created using this algorithm contain 89, 99, and 204 elements, respectively. The visualisation of the clusters based on pair plotting of the numerical features can be seen in the figure below.

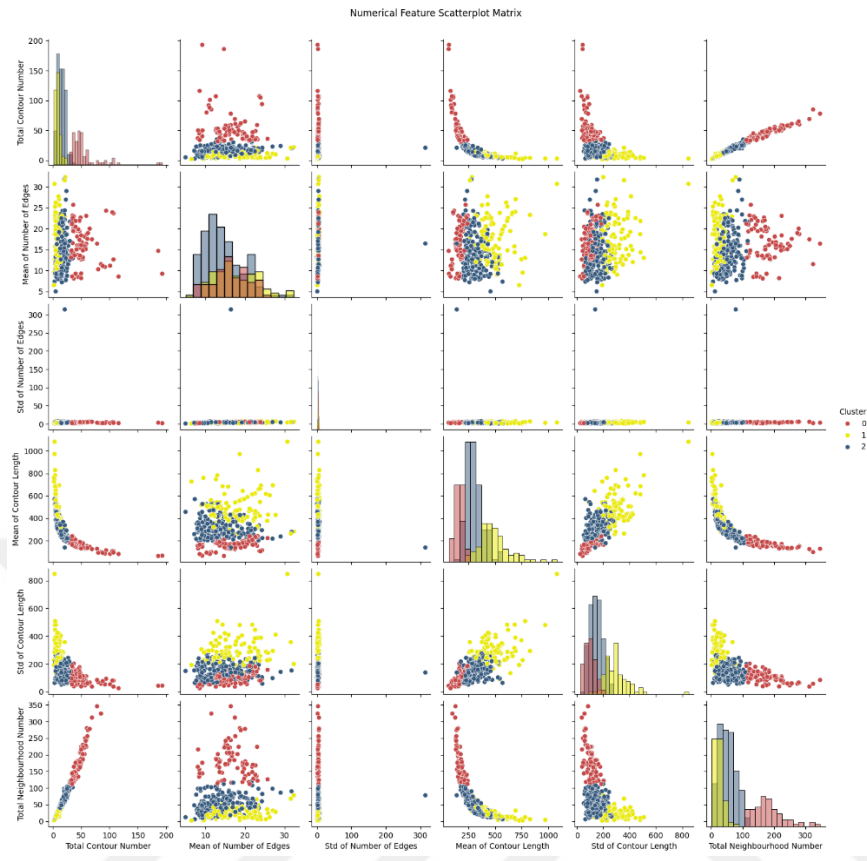


Figure 4.23 Pair Plotting with Clusters

The distribution of the data aligns with the correlations between the features, as it can be seen from the pair plotting. The separations are distinct throughout the clusters according to each different feature correlation. Examples from each cluster can be seen in Figure 4.24, which shows that the clusters are respecting the sizes and the shapes of the stones. The most common cluster is 2 since it has more patterns than the others.

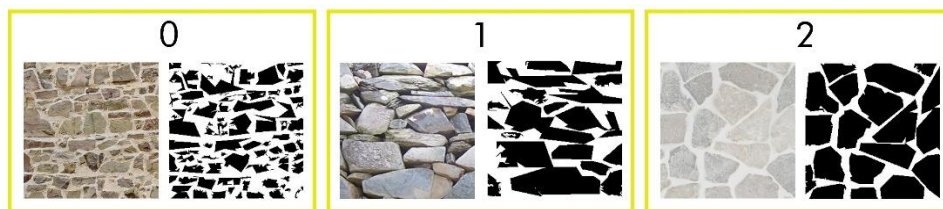


Figure 4.24 Examples from Clusters

CHAPTER 5

PLACEMENT AND ROBOT SIMULATION

5.1 Implementation of Proposed Method

The implementation of the pipeline based on the trained models starts with the sampling of individual pieces through the extracted individual contours from the original dataset. The first approach is to perform random sampling on the dataset of 8859 individual pieces that have the potential to reflect any possible salvage material. Since the individual pieces are extracted, without altering the image quality and size, from the original patterns with the same dimension for each, the proportion of the pieces and patterns is kept intact. In order to define the number of sampled stones, the black pixels of the patterns that indicate the stones are counted. After that, they are sorted from lowest to highest. Since there are some outlier values, the range is set to $1/3$ to $2/3$ of the list, which is 46948 - 52742 pixels. The algorithm stops sampling when the number of pixels reaches this range.

The acquired samples show that the sampling strategy needs to be developed due to the inconsistencies between the pieces. To achieve more systematic sampling, the Monte Carlo method was adopted in the sampling algorithm. Additionally, the pieces are clustered with a K-means model according to the features of a number of edges, contour length, area, and the aspect ratio of the bounding box. The number of clusters is set to 4 for this initial clustering. The Monte Carlo sampling is performed within the clusters to ensure the balance throughout the sample space. Besides, the number of pixels is multiplied by 5 to have more options while placing the selected base patterns. The required number of contours for detecting the pattern is defined by dividing the number of pieces that are sampled.

The acquired sample spaces are better than plain random sampling; nevertheless, they are not satisfactory, since the Monte Carlo Sampling is still subject to

randomness. Due to this, a final approach is proposed applying clustering with the features gathered from the patterns which are the mean number of edges, standard deviation of number of edges, the mean contour length, standard deviation of the contour length, the mean aspect ratio of the contour bounding box, standard deviation of the aspect ratio and the last and the most determinative feature is the distribution function for the area of the contours within a pattern. After this clustering, rather than applying Monte Carlo sampling, the clusters are divided into batches regarding the mentioned features with roughly the total pixel area of 20 times of an average area of the patterns, which is 941074 pixels. Totally 20 batches are created by following this logic, with a varying number of pieces from 310 to 618.

After that, the features of the sampled pieces are extracted to make predictions with the firstly trained decision tree model with the labels of polygonal, random, and square rubble. The number of contours for the prediction is defined by dividing the number of pieces inside the intended batch by 20, considering the total area it comprises. Since the stones are not composing a pattern yet, the total neighborhood number is missing to make the prediction. In order to overcome this issue, a linear regression model is trained with the features of total contour number and total neighborhood number that have a linear relationship. Through this regression model, estimated neighborhood numbers are collected to fill the missing value for the decision tree prediction. As a result, the model predicts the type of potential pattern that will be produced with the pieces from the created sample space. 3 examples of sampled pieces and the related predicted patterns of batch 0, batch 5, and batch 10 can be seen in the following figures.

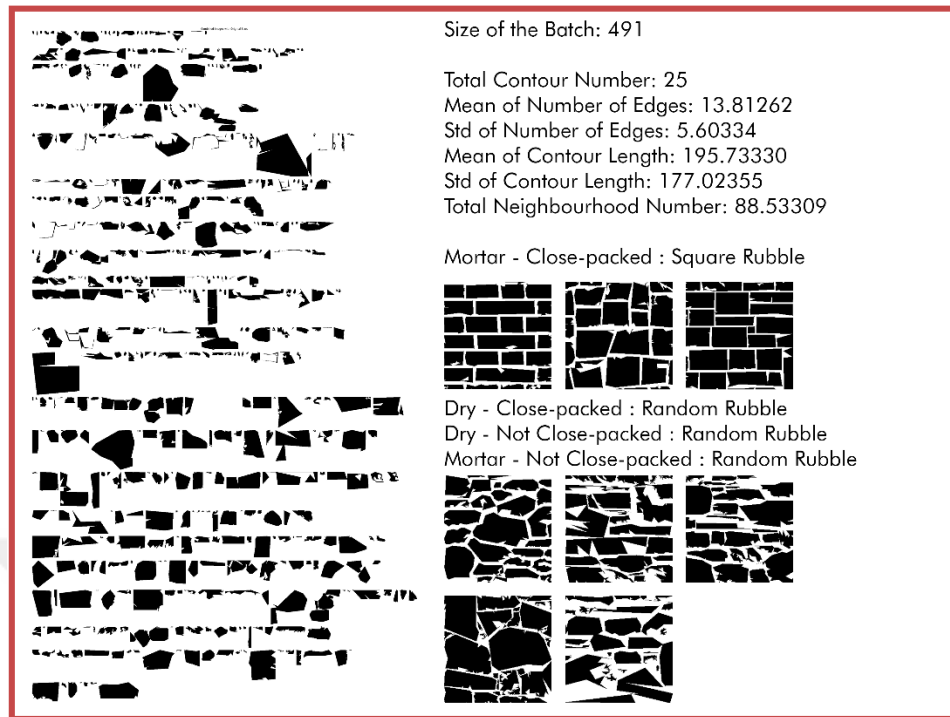


Figure 5.1 Sampled Pieces and Predicted Base Patterns of Batch 0

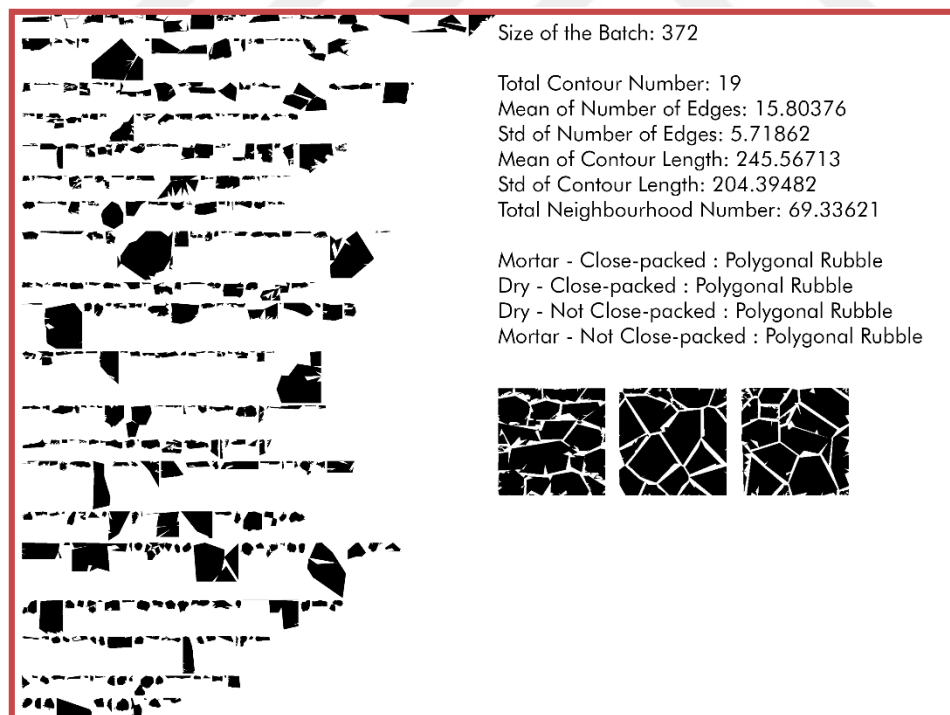


Figure 5.2 Sampled Pieces and Predicted Base Patterns of Batch 5

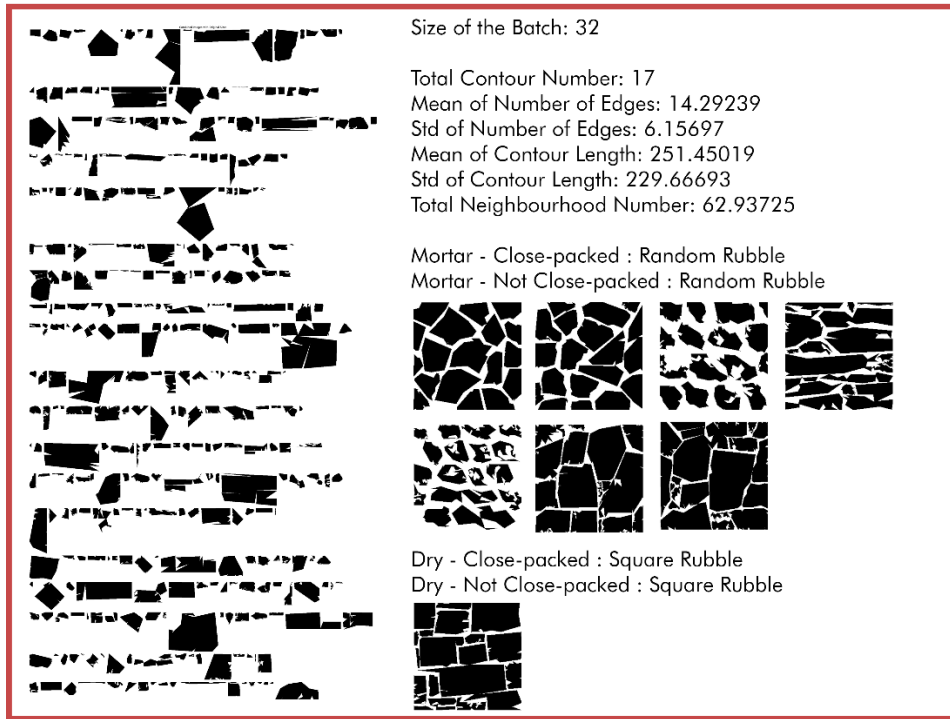


Figure 5.3 Sampled Pieces and Predicted Base Patterns of Batch 5

To find the pattern matches for sampled pieces, the dataset is filtered by using the prediction, which indicates if the pattern is polygonal, random, or square rubble. Then the remaining data is filtered again according to the number of pieces that the pattern should contain, according to the number of pieces in the sample space from the batch. The ones with the required number of contours are selected. If there is no exact number of elements with the existing patterns, the ones that have the closest number are selected as the most probable patterns.

5.2 Placement of Pieces into Matched Patterns

After acquiring the sample spaces and predicted base patterns for them, the most crucial part of this study is to have a placement approach that creates new configurations based on existing patterns. The pseudocode of the placement algorithm used in this study can be seen in the following figure, which shows the general approach to handling the placement.

```

1. extract_closed_polygons(doc)
Initialize empty polygon list
For each entity in modelspace:
    If entity is a closed POLYLINE or LWPOLYLINE:
        Convert vertices to polygon
        If polygon is valid:
            Add to polygon list
        Else:
            Try buffer(0) fix
            If fixed polygon is valid:
                Add to list
    If entity is CIRCLE or ELLIPSE:
        Approximate with a circular buffer & add to list
    If entity is HATCH:
        Extract loops & convert to polygon & validate and add
    If entity is SPLINE:
        Approximate with polygon from control points & validate and add
If no polygons found:
    Try to extract raw vertices from all supported entity types
    Compute bounding box around all points
    Return box as fallback
Return list of polygons

2. load_piece_files(folder, names)
pieces & empty list
for each filename in names:
    path & folder + filename
    if file exists:
        doc & load DXF
        polygons & extract_closed_polygons(doc)
        if polygons:
            pieces.append((filename, first polygon))
        else:
            log warning
    else:
        log missing file
return pieces

3. compute_feature_vector(poly)
hull & convex_hull(poly)
width, height & bounding box dimensions
aspect_ratio & width / height
return [area, perimeter, aspect_ratio]

4. match_pieces_to_contours(pieces, contours)
Compute feature vector for each piece and contour
for each contour:
    Sort all pieces by vector distance to contour
    Pick first unused piece as match
    If all used, pick closest one anyway
Return list of (piece_name, piece_poly, contour_poly) tuples

5. place_piece_no_overlap(piece_poly, contour, placed_pieces, pieces,
used_pieces)
placed_piece, fallback_count & align_piece_to_contour(piece_poly, con-
tour)
If placed_piece intersects any already placed piece:
    Try to find replacement_piece that doesn't overlap
    If found:
        realign new piece
        mark it as used
    Else:
        return None
return placed_piece

6. find_replacement_piece(original, pieces, used, contour, placed_pieces)
unused_pieces & all pieces not in used set and not original
Sort unused pieces by similarity to original (feature vector distance)
for each unused piece:
    align to contour
    if no intersection with placed_pieces:
        return that piece
return None

7. align_piece_to_contour(piece, contour)
angle_piece & PCA angle of piece
angle_contour & PCA angle of contour
Rotate piece to align angles
Translate piece centroid to contour centroid
Return translated piece and fallback counter

8. write_dxf(output_path, pattern polys, placed_pieces)
Create new DXF document
Create PATTERN and PLACED_PIECES layers
For each polygon in pattern polys:
    Add as polyline to PATTERN layer
For each polygon in placed_pieces:
    Add as polyline to PLACED_PIECES layer
Save document to output_path

9. Main Execution Flow
function main(pattern_file, pieces_folder, selected_piece_names, output_
file):
    pattern_doc & load DXF file from pattern_file
    pattern polys & extract_closed_polygons(pattern_doc)
    if pattern polys is empty:
        print "No valid contours found"
        exit
    pieces & load_piece_files(pieces_folder, selected_piece_names)
    matches & match_pieces_to_contours(pieces, pattern polys)
    placed_pieces & empty list
    used_pieces & empty set
    for each (name, piece_poly, target_contour) in matches:
        placed & place_piece_no_overlap(piece_poly, target_contour,
placed_pieces, pieces, used_pieces)

        if placed is not None:
            add placed to placed_pieces
            add name to used_pieces
        else:

```

Figure 5.4 Pseudocode of Placement Algorithm

The first step is to create a function that detects the polygons in related DXF files for both the patterns and the individual contours by going through the entities within the files. Another function is defined to load the individual contour files by extracting them from a folder that contains all of the individual files according to the names of the sampled pieces. Another function that is crucial is created to calculate the feature vectors of both contours in the patterns and the individual contours by gathering their convex hulls. The features used for forming the vector are the area of them, contour length, and the aspect ratio of their bounding boxes.

After that, another function is organized to match the individual contours with the contours extracted from the patterns based on acquired feature vectors. The pieces are sorted depending on their vector distance to the target contour, and then the closest piece that is not used is matched with the contour. From there on, principal component analysis (PCA) angles of the contours and pieces are used in a function to rotate and align, and translate the pieces from their centroids to the centroids of the matched contours. Considering any possible overlap as a result of initial matching, an additional function is defined to eliminate it by finding a replacement piece through applying the alignment process to inspect if there is any intersection. It iterates throughout the pieces according to the calculated feature distances. This process is done when all the contours are matched with a piece, although the ones are left empty if no match is found.

A separate function is defined to transfer the placed patterns into DXF files that contain both the reference base pattern in the PATTERN layer and the placed contours in the PLACED_PIECES layer. All the above-mentioned functions are organized step by step inside a main function that requires the reference pattern file and names of the sampled pieces, and it gives the output file. Several examples of the placement algorithm for different batches can be seen in the following figures.

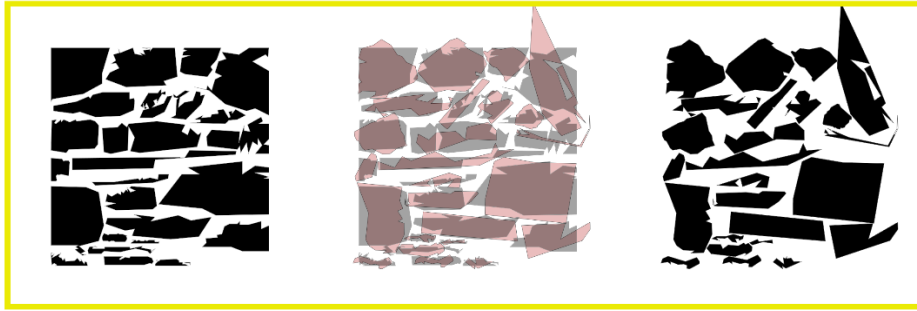


Figure 5.5 Example from Batch0 - Random Rubble

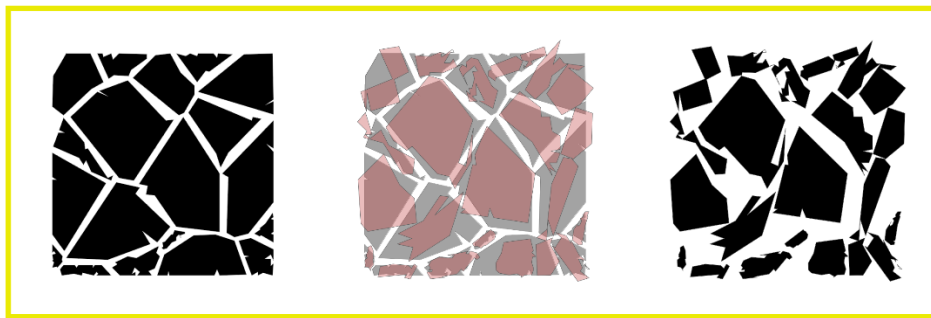


Figure 5.6 Example from Batch5 - Polygonal Rubble

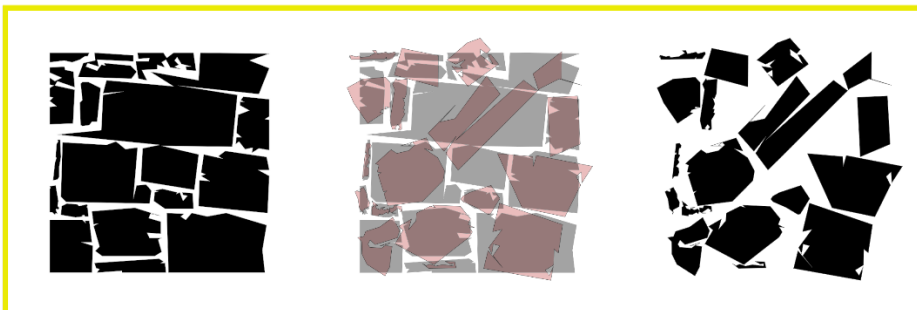


Figure 5.7 Example from Batch10 - Square Rubble

The prediction and placement algorithm is applied to all of the batches, resulting in 116 placements. 75 of them are random rubble, 24 of them are polygonal rubble, and 17 of them are square rubble.

5.3 Robot Simulation

The placement process is demonstrated in a simulation environment with a robot arm. In order to handle this step, one of the examples is selected among the placement options for the batches, that is, from batch 5, a pattern that is polygonal rubble. The simulation is implemented in the RoboDK environment by connecting it through the Python API. The flowchart that explains the script of the simulation is generally illustrated below.

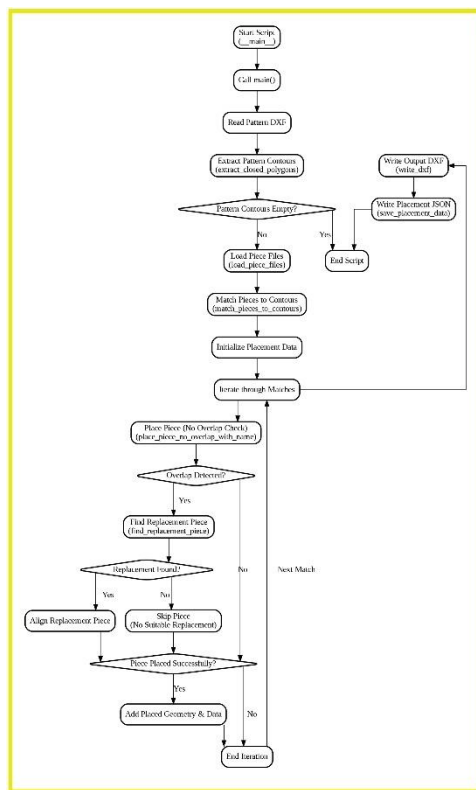


Figure 5.8 Pseudo-code of Robot Simulation

The initial step is to import required libraries and packages into Python, which are RoboDK, Robolink, Numpy, Os, Time, Shutil, Math, Stl, Ezdxf, Shapely, Scipy, Sklearn, Trimesh, and Pathlib. Then the Python environment is linked to the RoboDK through Robolink. The Fanuc Crx robot arm, which is available to use in real life, is imported into the program with a vacuum gripper (Figure 5.30).

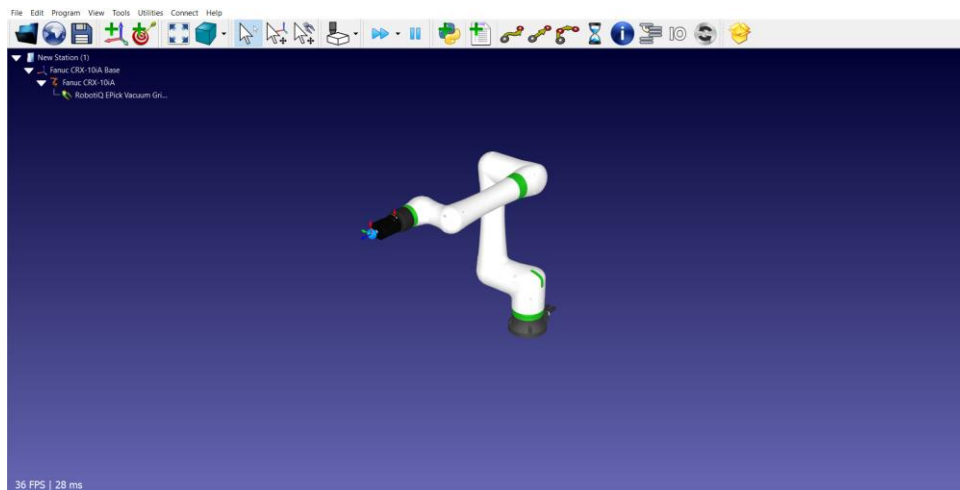


Figure 5.9 Robot and Vacuum Gripper

After that general formats of translation and rotation matrices are defined to control the robot's movement. Afterwards, the individual pieces within the selected sample space are extruded within Python by employing the trimesh and ezdxf libraries. Then they are saved in STL format, which is a standardized format that can be used in the RoboDK environment. After that, these pieces are imported into the simulation environment by following a circular layout around the robot base with a spacing that eliminates the collision between them. This layout algorithm also considers the robot's reach, whose information is extracted from the robot's workspace dimensions. The algorithm that defines the importing locations is also utilized in defining the pick locations (Figure 5.31). The robot is lifted 30 cm above to avoid any collision that might happen between the robot arm and the imported pieces. Later on, the DXF file that contains the contours of the base pattern for the placement is read within the Python environment that initiates the simulation. This reading regards the function of extracting closed polygons, which is used in the placement algorithm.

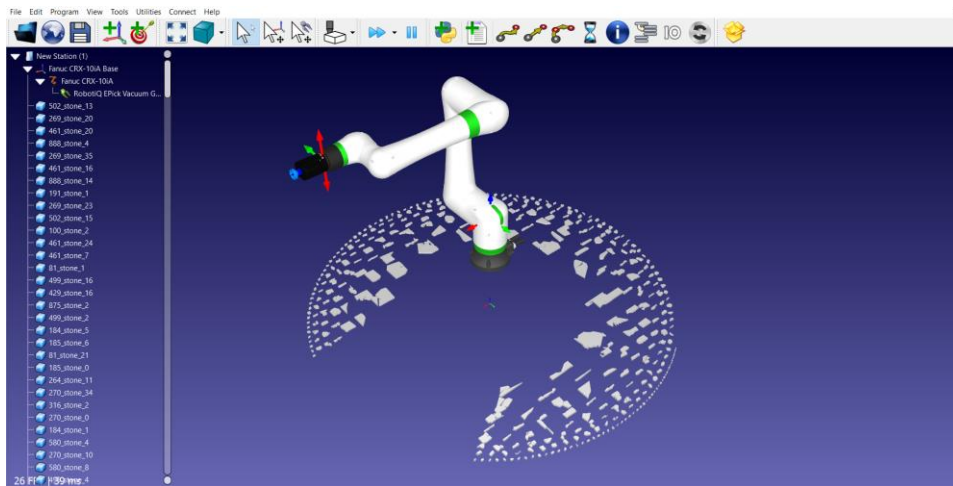


Figure 5.10 Imported Pieces

The location information of that DXF file is transferred to the RoboDK environment by shifting it to a place, within robot reach, that is left empty while generating the importing layout. Z coordinates of both the pieces and the pattern file are adjusted to the location of the robot base. The matching algorithm that was used in the DXF placement stage is also applied inside the simulation algorithm. Subsequently, the location information for the place positions is derived from the layout of the DXF.

The simulation of the pick and place task is defined within a function that respects the generated locations for both picking and placing by approaching them first and attaching and detaching the piece later. The approach poses are set 20 cm above the pick and place positions. The collision checker, which is available in both the interface and the API, is applied inside this function. There are also several side functions defined to smooth this process since the direct movement is not possible for some of the pieces. The first one is to find a configuration that prevents the robot from colliding with any object. The second function is defined to detect if there is an opposite movement that may cause the robot to collide with itself.

Accordingly, another pair of functions is defined to escape from these possible collisions by creating intermediate points that create an arc that takes the robot base as the center. These functions are put into operation according to the previous

functions that predict any possible collision. This is enabled by dynamically monitoring the simulation process.

Another important part of the simulation is defining the placement positions with the correct angle that aligns with the original pattern. The matching and alignment information is gathered from the proposed placement logic, as mentioned earlier, thus the robotic assembly is synchronized with the proposed method. Consequently, the robot is able to pick intended pieces and place them in the right place at the right angle.

5.4 Validation of the Method

Validation of the proposed method is done through selecting a pattern from the dataset and checking whether it brings itself back. For this aim, the pattern shown in Figure 5.8 is selected as the sample. Its extracted features are used for the prediction and filtering pipeline.

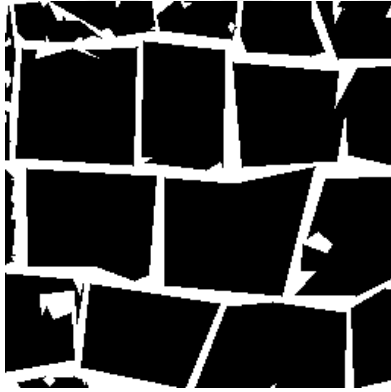


Figure 5.11 Selected Pattern for Validation

The features for this pattern are Total Contour Number: 19, Mean Number of Edges: 10, Standard Deviation of Number of Edges: 3.684, Mean Contour Length: 233.364, Standard Deviation of Contour Length: 122.74325954239046, and the Total Neighborhood Number is estimated by the linear regression model since it is used for the sampled ones even there is an exact number for the pattern.

The prediction from the decision tree model based on these values of the features resulted in Square Rubble. When the patterns that are both square rubble and have 19 pieces are filtered, 3 patterns remain that include the pattern that is used for the prediction. After that, the pattern and the individual contours extracted from this pattern are given to the proposed placement algorithm. As a consequence, the algorithm placed all of the contours in the right places with proper orientations. Hence, it can be confirmed that the proposed method has the potential to bring the pattern to itself, which could be a proving point of its utility.

5.5 Evaluation of the Placement

The matching and placing algorithm works well according to the resulting images. The overlaps are eliminated while the essence of the original patterns is kept within the configuration. Thus, the stereotomic bonds established in the original patterns are traced to the sampled pieces through their configuration. Despite the placement algorithm, the random sampling does not provide an adequate set of pieces such as the sizes and shapes are not consistent with each other. Another method of sampling is necessary to combine resembling pieces in order to have a more coherent configuration.

5.5.1 Similarity Measures

Matching can be defined as maximizing the level of similarity, specifically shape similarity, based on the comparison of the geometrical properties of shapes. Shape similarity can be used for different tasks, including pattern recognition, map generalization, and so on. Correspondingly, the measurement of similarity can be evaluated by implementing distance functions (Zhao et al., 2005; Qiaoping et al., 2002). The evaluation of the placement logic is carried out by implementing known similarity algorithms, depending on distance functions, since the initial step of the placement logic contains a similarity analysis based on feature distance calculation.

The selected similarity algorithms for this stage are the Hausdorff distance, Minkowski distance, Bottleneck Distance, and Turning functions distance. These selected similarity measures rely on the area of computational geometry (Veltkamp, 2001).

The first measure, the Hausdorff distance, is a measure that is quicker to calculate and relies on the information of the points (Biasotti, 2015). In spite of that, it is exceedingly susceptible to the outliers (Zhao et al., 2005). Its formula can be seen below (Alt, 2009):

$$\delta H(A, B) = \max(\delta H(A, B), \delta H(B, A))$$

Another point-based measure is the Minkowski distance, which is also called L_p . For 2 points as x and y , it is calculated as (Veltkamp, 2001):

$$L_p(x, y) = (\sum_{i=1}^n |x_i - y_i|^p)^{1/p}$$

The Bottleneck distance is the minimum value of calculated one-to-one distances of sets of points (Veltkamp, 2001). The last measure is the Turning Function Distance, also known as the cumulative angle function, which is the distance between the angles of the edges that are before and after sequential vertices. Graphical representation of this function for an instance can be seen in the following figure (Andreou & Sgouros, 2005; Veltkamp, 2001).

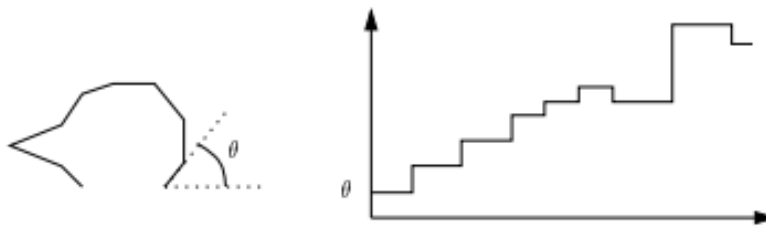


Figure 5.12 Polygonal Curve and Its Turning Function (Veltkamp, 2001)

5.5.2 Placement with Similarity Measures

In order to assess the matching algorithm, the mentioned similarity measures, which are Hausdorff distance, Minkowski distance, Bottleneck distance, and Turning Functions distance, are used to match pieces to the contours of the patterns. The matched pieces are placed in the corresponding pattern from their centroids. This process is applied to all of the 116 predictions that are gathered for 20 batches. The examples that are used for the placement algorithm are also depicted by using similarity measures in the figures below to compare.



Figure 5.13 Example from Batch 0 - Random Rubble - Hausdorff Distance



Figure 5.14 Example from Batch 0 - Random Rubble - Minkowski Distance

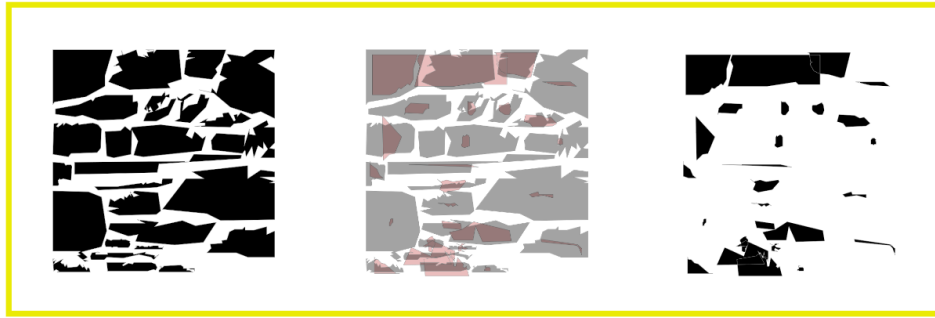


Figure 5.15 Example from Batch 0 - Random Rubble - Minkowski Distance



Figure 5.16 Example from Batch 0 - Random Rubble - Turning Functions Distance

The figures of examples of random rubble from batch 0, that have matching with similarity measures, show that almost all of them have overlaps at different levels. Moreover, some of the pieces have too large a size difference from their matched contour. When examining visually, the Hausdorff distance gives the best matching since it appears to have more proportional matches.

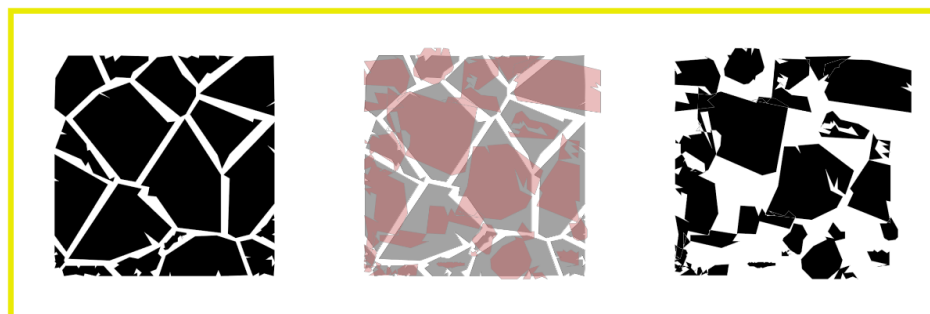


Figure 5.17 Example from Batch 5 - Polygonal Rubble - Hausdorff Distance



Figure 5.18 Example from Batch 5 - Polygonal Rubble - Minkowski Distance



Figure 5.19 Example from Batch 5 - Polygonal Rubble - Bottleneck Distance

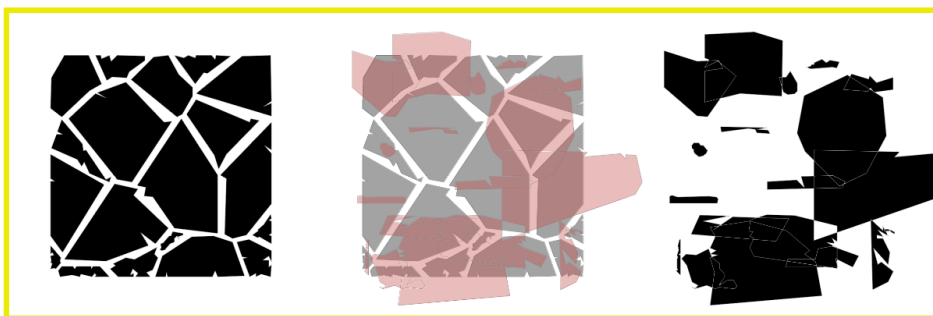


Figure 5.20 Example from Batch 5 - Polygonal Rubble - Turning Functions Distance

For the matches of polygonal rubble, there are also overlaps and proportional differences. Even though the Bottleneck distance provides more proportional matches, the Hausdorff distance seems to have fewer overlaps and still has proportional matches.

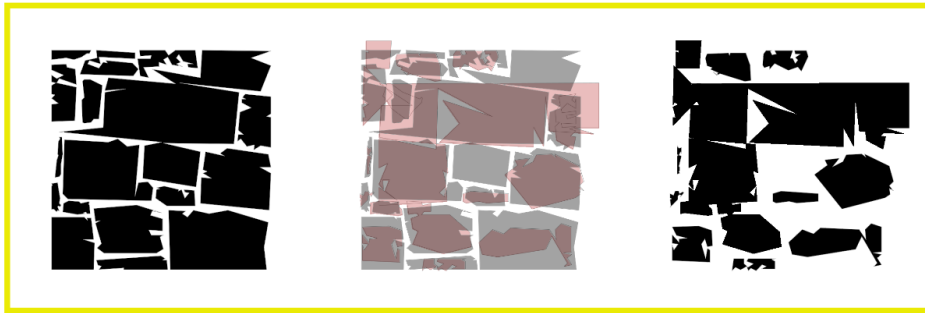


Figure 5.21 Example from Batch 10 - Square Rubble - Hausdorff Distance



Figure 5.22 Example from Batch 10 - Square Rubble - Minkowski Distance

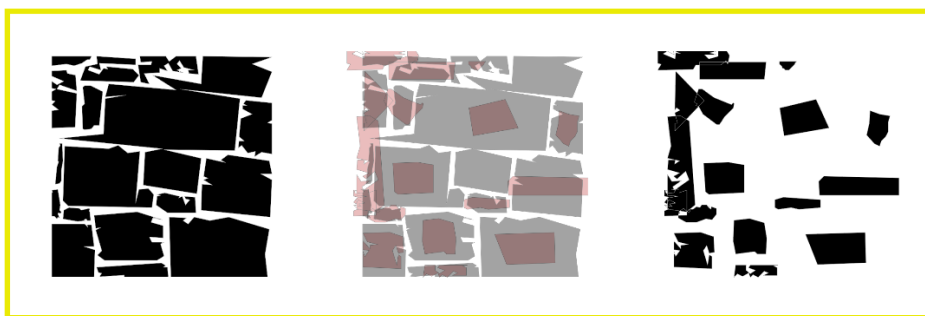


Figure 5.23 Example from Batch 10 - Square Rubble - Bottleneck Distance



Figure 5.24 Example from Batch 10 - Square Rubble - Turning Functions Distance

For the example of square rubble, the similarity measures failed considerably. Like the previous attempts, the Hausdorff distance ensures the most suitable matching among them. Further, the feature distances, the difference of the coverage ratios and overlap ratios of the placement of our approach, and matching with the similarity measures are calculated to compare and evaluate the proposed method. The predictions for batch 0 are used for this process, which contains 8 patterns, 3 of which are predicted as square rubble, and 5 of them are predicted as random rubble with the contour number of 25.

Table 5.1 Feature Distances

Filename	Our Approach	Hausdorff	Minkowski	Bottleneck	Turning Functions
batch0_output_placed_499	1491.931522	770.4620971	1434.876554	1226.301846	1262.915673
batch0_output_placed_139	1276.277856	712.3260985	1058.197394	1037.063924	2771.034378
batch0_output_placed_245	1914.906324	821.820221	2157.401245	1436.950694	2203.804364
batch0_output_placed_1025	1474.572222	772.5159762	1647.851088	1903.645981	2021.467763
batch0_output_placed_226	1707.685866	639.4580143	1557.702745	1374.292396	2590.868827
batch0_output_placed_228	1307.187593	792.7092746	1017.509219	1298.971014	2734.207309
batch0_output_placed_1038	917.8518531	1087.689476	1638.359828	1878.758257	3317.204436
batch0_output_placed_897	933.4656051	1066.949739	1392.81721	1280.895294	3258.941263

Feature distances of the patterns and their matched contours are calculated by employing the features of contour length, edge number, area, and aspect ratio. As a result of this computation, Hausdorff distance performs mostly better results while the Turning Functions distance causes larger distances. Our approach stands on average values that could be caused by the logic of moving to the next best piece to prevent overlaps, which increases the feature distance.

Table 5.2 Overlaps

Filename	Our Approach	Hausdorff	Minkowski	Bottleneck	Turning Functions
batch0_output_placed_499	0	0.040211205	0.289669597	0.058374806	0.124099977
batch0_output_placed_139	0	0.041647749	0.055656454	0.025005346	0.172316726
batch0_output_placed_245	0	0.0750346	0.1464806	0.137118304	0.176793395
batch0_output_placed_1025	0	0.013415257	0.003517973	0.089362796	0.160084159
batch0_output_placed_226	0	0.057842578	0.134087883	0.119296495	0.145029753
batch0_output_placed_228	0	0.082414797	0.082688648	0.040171948	0.140498819
batch0_output_placed_1038	0	0.087995	0.072051881	0.239431703	0.274960033
batch0_output_placed_897	0	0.057808781	0.026928904	0.139101239	0.267232251

For the overlap calculation, our approach gives the value of 0 since it has a logic of eliminating them. Apart from that, the lowest value is changing through the measures, yet the Turning Functions distance doesn't have any of them.

Table 5.3 Coverage Distances

Filename	Our Approach	Hausdorff	Minkowski	Bottleneck	Turning Functions
batch0_output_placed_499	0.14458495	0.035389828	0.26544301	0.079493968	-0.087791474
batch0_output_placed_139	0.142266616	0.117234783	0.187956662	0.239369375	-0.073583367
batch0_output_placed_245	0.222910454	0.010202824	0.187956662	0.239369375	-0.073583367
batch0_output_placed_1025	0.272947365	0.161235229	0.543713072	0.010008924	0.021002585
batch0_output_placed_226	0.120453221	0.038920886	0.331159068	0.13856699	-0.363372115
batch0_output_placed_228	0.044746803	0.007864022	0.192302601	0.449845445	-0.294955346
batch0_output_placed_1038	0.250729255	0.130977403	0.421542133	-0.300250221	-0.442160373
batch0_output_placed_897	0.146773485	0.196923818	0.421527066	-0.122325227	-0.617480357

Coverage distance is calculated by subtracting the ratio of filled space of the created pattern from the original pattern. It can be both positive and negative depending on the sizes of the placed pieces compared to the existing ones. This distance is changing through the methods, so that there is no prominent one that performs the best; still, the Hausdorff distance seems to be the best one. Furthermore, the results of our approach are lower than 0.3, while the remaining measures have several large values. As a result of these metrics, the Turning Functions distance is the worst similarity measure for this case, when our approach, that based on feature distances, can be considered as the best solution.

5.5.3 Applicability of the Placement with Tetris

Tetris blocks are selected as the test case of the applicability of the method of this study because of their variety of pieces and configuration options. The Tetris configuration that is initially used for this study is shown in Figure 5.22, which couldn't be composed with the proposed method due to the conflict between the logic of Tetris and the proposed method.

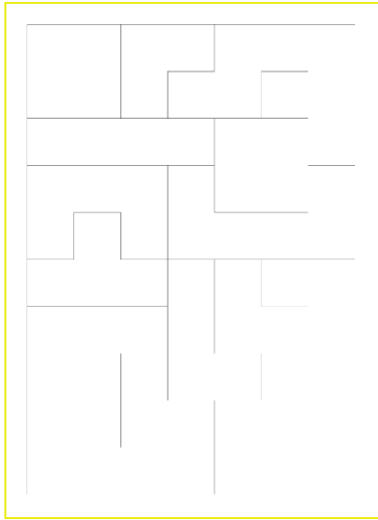


Figure 5.25 Pattern with Tetris Blocks

There are two major conflicts between the logic of Tetris and our approach. The first one is that the edge-to-edge interlocking of the Tetris blocks is considered as overlaps during the placement algorithm. As a result of this conflict, it is represented as the figure below.

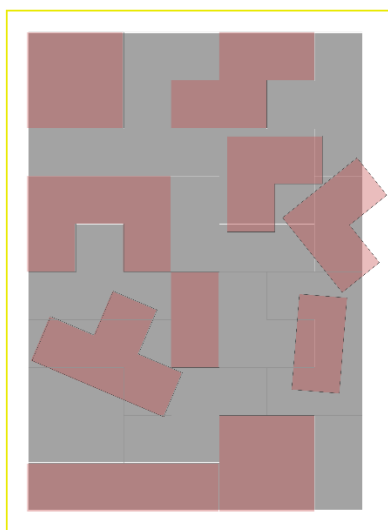


Figure 5.26 Conflicted Tetris Placement

In order to solve this conflict, an offset is applied to the pieces; even so, the placement wasn't flawless because of the second conflict. The second conflict is caused by the convex hull method that is employed to handle irregular pieces. The Tetris pieces are covering a grid; thus, there is more than one piece that could be covered by the same convex hull. This situation causes overlaps when the algorithm tries to put another piece that has the same convex hull as the intended piece. The pieces that can cause this problem can be seen in the figure below.

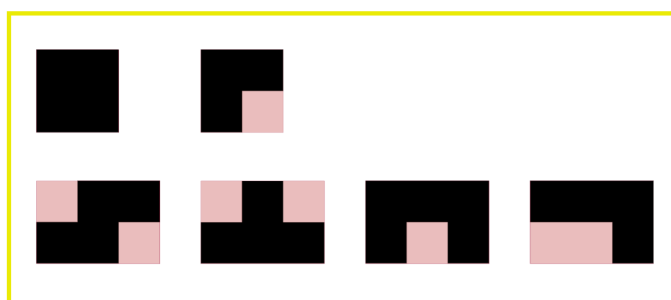


Figure 5.27 Tetris Blocks that Belong to 2 Different Convex Hulls

After realizing these conflicts. A simple Tetris configuration (figure 5.25) with narrow gaps and pieces that don't have the same convex hull is tested with the placement algorithm. The algorithm manages to put the correct pieces in their position.

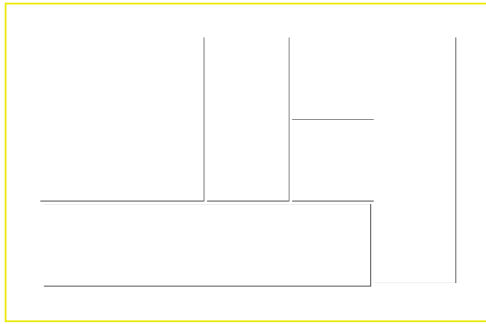


Figure 5.28 Simple Tetris Configuration with Distinct Blocks

5.5.4 Applicability of the Placement with the City Plan of Barcelona

The practicability of the proposed method is planned to be tested with a city plan, as it can be used for recomposition of damaged parts of the cities using their own pattern. For accomplishing that, the city of Barcelona is selected due to its high reachability. The city is divided into two parts from the middle of its grid system, to test the prospective methodology as towards the mountainside and towards the seaside (figure 5.26). After that, different portions of the city are used as the base patterns, as well as separating individual pieces that represent building blocks.

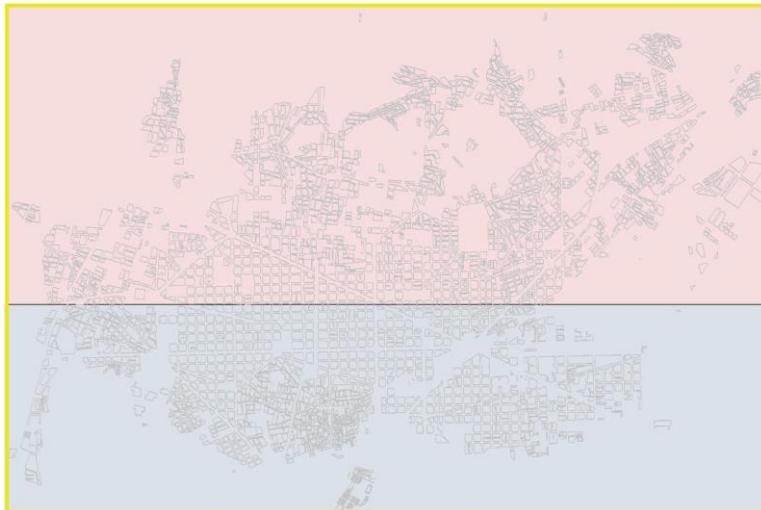


Figure 5.29 Division of the City of Barcelona for Testing the Applicability

The blocks of mountainside are used for placing the patterns of seaside and vice versa. The examples from the first attempt that use the blocks of mountainside can be seen in Figure 5.27. Since most of the blocks on that part are not following the grid system, the shape of the placed blocks differs.

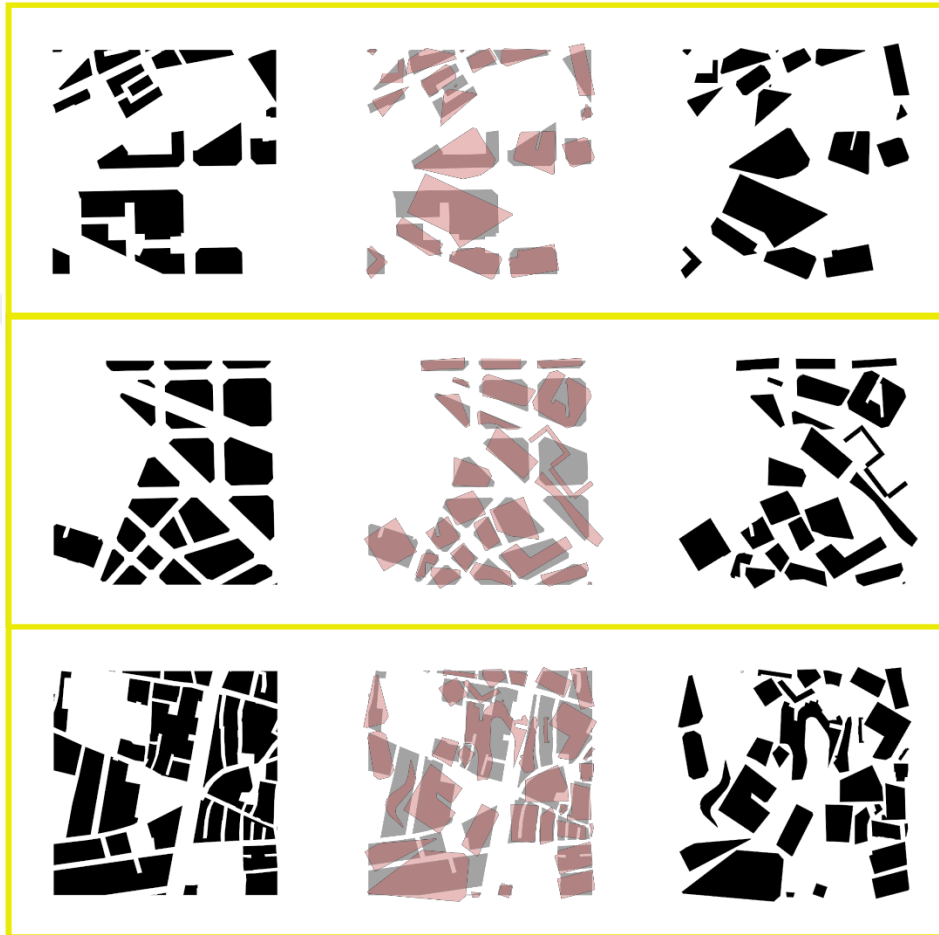


Figure 5.30 Placement of Blocks of Mountainside to Patterns of Seaside

Figure 5.28 shows the placement of seaside blocks to patterns of the parts of the mountainside that broadly respect the grid. Consequently, more feasible placements are gathered.

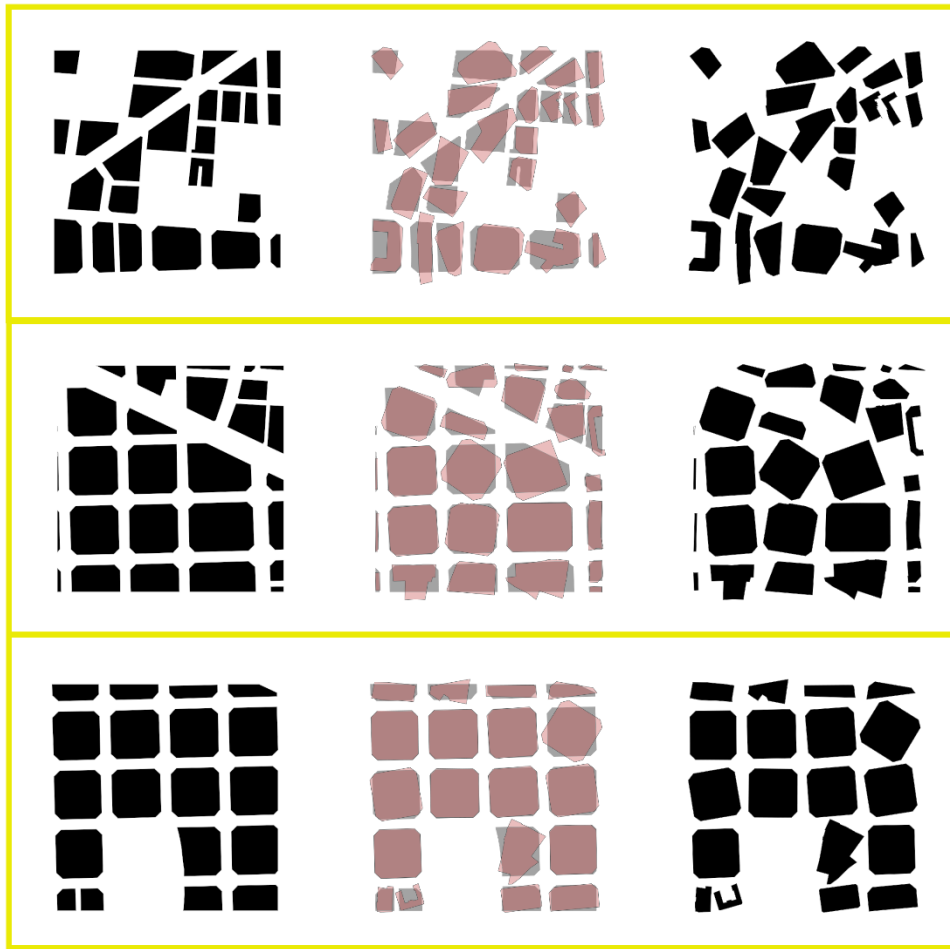


Figure 5.31 Placement of Blocks of Seaside to Patterns of Mountainside

The metrics that are used for evaluating the similarity measures are also employed for this testing. The feature distances for the first attempt are much larger than the second attempt, while the distances of the coverage are low in both of them (Table 5.4).

Table 5.4 Evaluation of the Placements Based on the Barcelona City Plan

	Filename	Feature Distance	Coverage Distance
Attempt 1	1	131.32310	-0.01876
	2	157.37721	0.06107
	3	821.19668	0.07570
Attempt 2	1	89.09638	0.02613
	2	50.07654	0.01517
	3	31.29689	0.00814

As can be observed from the table, the 3rd trial of the second attempt gives the best result, which can also be noticed from the visual continuity in the image.

CHAPTER 6

CONCLUSION

6.1 Discussion

Masonry, stacking, and modularity are among the key concepts in architectural structures that can be constituted by integrating diverse materials and forms, including irregular/nonconventional ones. These concepts present the technique of composing a structure by using smaller pieces. The irregularity/non-conventionality in the forms of these pieces reveals the precision problem that emerges from the connection among them. As another aspect, this study proposed to use salvage materials, which mostly have irregular shapes, as a potential source, since it addresses the circularity. The mentioned precision/stacking problem is a standing topic in the related research area contains different approaches to solve it. At this point, this study attempted to solve this problem by integrating a stereotomy into the methodology that contains machine learning, mathematical/geometrical operations, and robot simulation to reduce the required time, skill, and labour that are the challenges of conventional methods.

The first objective of this study, which is *"To find a way of combining pre-existing structural elements to have an architectural component."*, is addressed by utilizing the logic of masonry and stacking. In order to understand and apply these concepts, the related literature is reviewed. Correspondingly, the contemporary approaches are examined through broadening the vision of this study. These methods are also used while creating the main dataset of this study. The second objective, which is *"To understand the concept and application of stereotomy in masonry structures by reviewing the literature and integrating it into the methodology,"* is addressed by reviewing the works about both conventional and re-emphasized stereotomy called Digital Stereotomy. These studies show that the stereotomy is a long-established

technique and mostly done through cutting and carving the material to fit them edge to edge, yet the possibility of putting the materials together without any alteration is explored thoroughly within the scope of this study.

The third objective, *"To propose a way of reusing the salvage materials as the structural elements from urban depositories in order to uphold circularity,"* reveals itself as reclaiming the salvage materials that are represented by the individual pieces that are extracted from the masonry wall patterns that comprise the dataset. Besides, the circularity is also assured by avoiding the waste as mentioned earlier. The fourth objective is the most important one, which is *"To use machine learning methods for learning from existing masonry wall patterns to generate the combination of elements."*, that establishes the backbone of the study. Multiple ML models are trained to reach the maximum capability of masonry wall structures. The new configurations are established upon the outputs of these models, the decision tree in particular. This study enabled geometrical/mathematical operations as well as adding to these machine learning outputs, with regard to the original configurations. These operations start with the matching and continue with overlap avoidance to reach the maximum potential of the pieces and the patterns.

The last objective that is *"To decrease the need for labor via the utilization of robotics during the fabrication of the proposed composition"* is initiated through the simulation environment that has the potential to be applied in real real-life environment. The placement information, the matching, and the alignment are transferred to the RoboDK environment by its Python API. The robot arm is able to detect the piece and place it with the required orientation and configuration. The next step is to carry out the matching algorithm with the shapes gathered by a computer vision application through information from the camera that will be installed on the robot arm.

6.2 Conclusion

To sum up, this study succeeded in reaching the specified objectives with verifiable results. The comparisons and application of the method in different fields support that it has validity. It can be derived from that new configurations of irregular/non-conventional forms can be created by learning from masonry wall patterns by machine learning implemented methodology. Besides, the created configurations are feasible to assemble by a robot arm. So that the followed methodology could be considered as a successful application since it fulfills the required objectives, yet there is more to improve it, which is explained below.

6.3 Future Remarks

In terms of future remarks, this study is open to new kinds of contributions. The prior step is to carry out the robotic assembly into the real-world environment, which requires advancements in the Python implementation, like integrating computer vision into the process to be able to visually detect the objects. The robot arm that is used in the simulation, which is Fanuc CRX, is available in real life to perform the assembly. The RoboDK simulation files are transferable to the format that connects with Fanuc robots' processor.

Another one is to create a dataset from real salvage materials to perform the proposed methodology. These materials can be windows, doors, concrete blocks, and bricks, since the proposed method solves planar configurations. Further, the method can be adapted to different types of problems other than a structural problem. It can be applied to complete the damaged parts of cities, as the initial trials are done with the city plan of Barcelona. A distinct application area can be the formation of a tissue from cells, to be able to achieve that, a dataset from different tissue patterns should be collected.

Another prospective step, which could be the most complex task within the scope of the stated problem, is extending the scope of this study into a 3D environment. In

this stage, it solves the problem in planar mode, which reflects the basics of the configuration problem. The conversion to 3D requires more detailed data, more features, and more geometrical complexity to solve. The 3D data can be collected by scanning with LIDAR or photogrammetry applications. After that, the process can be carried out with meshes or voxels. Depending on that, the feature extraction may also differ since the geometrical information of meshes and voxels is different from each other.

6.4 Limitations

This study proposes an innovative approach for creating structural configurations of materials with irregular/non-conventional forms. In spite of that, there are certain limitations for the implementation stage. The first limitation is to solve the 3D configuration in this stage, since the collected data represents planar information. The collection of 3D data that is sufficient for training machine learning models requires an abundant amount of time, which was not available for this study. The second limitation is the gathered contour lines, which are extracted by computer vision applications. Its optimization consumes time, but deep learning models provide more robust and accurate results. Although the reason for utilizing the current method is its compatibility with the ezdx library, hence acquiring the contours in vector format.

Another limitation is the gap amount between the pieces in the final configurations. The algorithm tends to pick smaller pieces while it tries to eliminate overlaps, along with the fact that the existing patterns also have gaps at a certain level. Accordingly, the output configurations may require mortar to be able to stand. Another limitation of this study is the amount of data that is scraped and processed. 393 images are sufficient to enable the proposed approach; still, a wider dataset could provide more enhanced results. Another limitation is the lack of real salvage material data. Because of this deficiency, the extracted individual units are used as representative data for the implementation part, relying on their shape variety.

REFERENCES

- Andreou, I., & Sgouros, N. M. (2005). Computing, explaining and visualizing shape similarity in content-based image retrieval. *Information Processing & Management*, 41(5), 1121–1139. doi:10.1016/j.ipm.2004.08.008
- Alt, H. (2009). The computational geometry of comparing shapes. *Lecture Notes in Computer Science*, 235–248. doi:10.1007/978-3-642-03456-5_16
- Barun, I., & Kranjec-Markešić, A. (2015). Divovske rađevine, Nova Akropola, za boljeg čovjeka i bolji svijet. www.nova-akropola.com
- Belousov, B., Wibranek, B., Schneider, J., Schneider, T., Chalvatzaki, G., Peters, J., & Tessmann, O. (2022). Robotic Architectural Assembly with tactile skills: Simulation and optimization. *Automation in Construction*, 133, 104006. <https://doi.org/10.1016/j.autcon.2021.104006>
- Bolhassani, M., & Wisniewski, S. P. (2022). Design and fabrication of a post-tensioned funicular masonry structure. *In Proceedings of the IASS 2022 Symposium affiliated with APCS 2022 conference Innovation-Sustainability-Legacy. Beijing.*
- Biasotti, S., Cerri, A., Falcidieno, B., & Spagnuolo, M. (2015). 3D artifacts similarity based on the concurrent evaluation of heterogeneous properties. *Journal on Computing and Cultural Heritage*, 8(4), 1–19. doi:10.1145/2747882
- Brütting, J., Senatore, G., & Fivet, C. (2021). Design and fabrication of a reusable kit of parts for diverse structures. *Automation in Construction*, 125, 103614. <https://doi.org/10.1016/j.autcon.2021.103614>
- Chiang, Y.-C., Bier, H., & Mostafavi, S. (2018). Design to robotic assembly: An exploration in stacking. *Frontiers in Digital Humanities*, 5. <https://doi.org/10.3389/fdigh.2018.00023>
- Certain Measures. (n.d.). <https://www.certainmeasures.com/projects/mine-the-scrap>

- Clifford, B., & McGee, W. (2018). Cyclopean cannibalism. A method for recycling rubble. *ACADIA Proceedings*.
<https://doi.org/10.52842/conf.acadia.2018.404>
- Colella, M. (2018). Stereotomy 2.0 and Digital Construction Tools. *Nexus Network Journal*, 20(3), 769–788. doi:10.1007/s00004-018-0407-7
- Cousin, T., Marshall, D., Pearl, N., Alkhayat, L., & Mueller, C. (2023). Integrating irregular inventories: Accessible technologies to design and build with nonstandard materials in architecture. *Journal of Physics: Conference Series*, 2600(19), 192004. <https://doi.org/10.1088/1742-6596/2600/19/192004>
- Diles, J. (2018). Lightweight stereotomy with glass-fiber reinforced plastic. *Nexus Network Journal*, 20(3), 645–669. <https://doi.org/10.1007/s00004-018-0376-x>
- Douglas, D. H., & Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2), 112–122. <https://doi.org/10.3138/fm57-6770-u75u-7727>
- Fallacara, G., & Barberio, M. (2018). An unfinished manifesto for stereotomy 2.0. *Nexus Network Journal*, 20(3), 519–543. <https://doi.org/10.1007/s00004-018-0390-z>
- Fallacara, G., & Gadaleta, R. (2021). Stereotomy: Architecture and mathematics. *Handbook of the Mathematics of the Arts and Sciences*, 1325–1344. https://doi.org/10.1007/978-3-319-57072-3_85
- Fallacara, G., & Stigliano, M. (2012). *Stereotomy: modern stone architecture and its historical legacy* (1st ed.). November 2024, <https://www.ateliers-romeo.com/wp-content/uploads/2014/09/Stereotomy-Modern-Stone-Architecture-and-Its-Historical-Legacy.pdf>

- Fernando, S. (2019). Collaborative crafting of interlocking structures in stereotomic practice. *Blucher Design Proceedings*. https://doi.org/10.5151/proceedings-ecaadesigradi2019_116
- Flynn, B. (2015). *The Complete Guide to building with Rocks & Stone: Stonework projects and techniques explained simply*. Atlantic Publishing Group, Inc.
- Foraboschi, P. (2019). Masonry does not limit itself to only one structural material: Interlocked masonry versus cohesive masonry. *Journal of Building Engineering*, 26, 100831. <https://doi.org/10.1016/j.jobbe.2019.100831>
- Furrer, F., Wermelinger, M., Yoshida, H., Gramazio, F., Kohler, M., Siegwart, R., & Hutter, M. (2017). Autonomous Robotic Stone stacking with online next Best Object Target Pose Planning. *2017 IEEE International Conference on Robotics and Automation (ICRA)*. <https://doi.org/10.1109/icra.2017.7989272>
- Gadaleta, R. (2018). New stereotomic bond for the dome in stone architecture. *Nexus Network Journal*, 20(3), 707–722. <https://doi.org/10.1007/s00004-018-0379-7>
- Gorgolewski, M. (2018). Resource salvation: The Architecture of Reuse. *John Wiley & Sons, Inc.*
- Grangeot, M, Fivet, C., & Parascho, S. (2023). From concrete waste to walls: An investigation of reclamation and digital technologies for new load-bearing structures. *Journal of Physics: Conference Series*, 2600(19), 192019. doi:10.1088/1742-6596/2600/19/192019
- Grangeot, M., Paracscho, S., & Fivet, C. (2024). Concrete rubble as a new construction material: panorama of applications to known structural typologies. *Proceedings of the IASS 2024 Symposium*.
- Graur, A.-M. (2020). Geometry as Assembly in Architecture. Semi-regular Polyhedrons. *Journal of Industrial Design and Engineering Graphics*, 15(2), 17–20.

- Guillén, J. F., Chao, Y.-H., Reich, S., & Buchmann, C. (2025). A computational workflow for crafting large-scale coral inspired modules through additive manufacturing. *Construction Robotics*, 9(1). doi:10.1007/s41693-025-00150-4
- Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, 28(1), 75–105.
- Huang, Y., Alkhayat, L., De Wolf, C., & Mueller, C. (2021). Algorithmic circular design with reused structural elements: Method and tool. *Fib Proceedings*. <https://doi.org/10.35789/fib.proc.0055.2021.cdsymp.p056>
- Hunkar, E., & Lee, D. (2023). ACADIA 2023. In *Habits of the Anthropocene* (pp. 208–213). Denver, Colorado: ACADIA.
- Ivaniuk, E., Friedrich Eichenauer, M., Tošić, Z., Müller, S., Lordick, D., & Mechtcherine, V. (2022). 3D printing and assembling of frame modules using printable strain-hardening cement-based composites (SHCC). *Materials & Design*, 219, 110757. <https://doi.org/10.1016/j.matdes.2022.110757>
- Jhons, R. L. (2023). *Autonomous Dry Stone Mobile Robotic Construction with Naturally Nonstandard Materials* (dissertation). ETH Zurich, Zurich.
- Kang, S., Dy, J. G., & Kane, M. B. (2023). Stone masonry design automation via reinforcement learning. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 37. <https://doi.org/10.1017/s0890060423000100>
- Knippers, J., Solly, J., Koslowski, V., & Fildhult, T. (2016). Modular Coreless Filament Winding For Lightweight Systems in Architecture.
- Ko, M.-C., & Hsieh, S.-J. (2023). Design and evaluation of algorithms for stacking irregular 3D objects using an automated material handling system. *The International Journal of Advanced Manufacturing Technology*, 126(5–6), 1951–1964. <https://doi.org/10.1007/s00170-023-11248-w>

Larsson, M., Kaiser, A., & Girhammar, U. A. (2012). World Conference on Timber Engineering.

Liu, J., Chen, X., & Yu, S. (2024). From junk to genius: Robotic arms and AI crafting creative designs from scraps. *Buildings*, 14(12), 4076. doi:10.3390/buildings14124076

Liu, Y. (2023). *Assembly Planning for Robotic Construction with In-Situ Materials* (dissertation).

Liu, Y., & Napp, N. (2023). Stability-based sequence planning for robotic dry-stacking of natural stones. *IEEE Robotics and Automation Letters*, 8(9), 5894–5901. <https://doi.org/10.1109/lra.2023.3301275>

Liu, Y., Saboia, M., Thangavelu, V., & Napp, N. (2019). Approximate stability analysis for drystack structures. *2019 International Conference on Robotics and Automation (ICRA)*. <https://doi.org/10.1109/icra.2019.8794158>

Liu, Y., Shamsi, S. M., Fang, L., Chen, C., & Napp, N. (2018). Deep Q-learning for dry stacking irregular objects. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. <https://doi.org/10.1109/iros.2018.8593619>

Lynn, G. (2009). Blobwall - Greg Lynn form. *Architectural Design*, 79(2), 96–99. <https://doi.org/10.1002/ad.861>

M Rucker, I. (2006). Calculus-based form: An interview with Greg Lynn. *Architectural Design*, 76(4), 88–95. <https://doi.org/10.1002/ad.298>

MacQueen, J. (1967) Some methods for classification and analysis of multivariate observations, *Proceedings of the Fifth Berkeley Symposium On Mathematical Statistics and Probabilities*, 1, 281-296.

- Menezes, A., Vicente, P., Bernardino, A., & Ventura, R. (2021). From rocks to walls: A model-free reinforcement learning approach to dry stacking with irregular rocks. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. <https://doi.org/10.1109/cvprw53098.2021.00234>
- Mostafavi, S., Kemper, B. N., Kretzer, M., Etemadi, A., Mahmoud, H. A., Yaseen, A., ... Chatterjee, S. (2023). Cobotic matters – collaborative robots and Discrete Assembly Design: From stacking to self-interlocking of reciprocal components. *CAADRIA Proceedings*, 2, 241–250. doi:10.52842/conf.caadria.2023.2.241
- Müller-Zitzke, P. (2024). Rethinking Modularity Designing a Prototype for Checkered Stacking (dissertation).
- Paris, V., Ruscica, G., Olivieri, C., & Mirabella Roberti, G. (2023). A systemic approach to simulate the construction process of self-supporting masonry structures. *Sustainability*, 15(12), 9596. <https://doi.org/10.3390/su15129596>
- Pham, Q.-H., Nguyen, T., Hua, B.-S., Roig, G., & Yeung, S.-K. (2019). JSIS3D: Joint semantic-instance segmentation of 3D point clouds with multi-task pointwise networks and multi-value conditional random fields. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. <https://doi.org/10.1109/cvpr.2019.00903>
- Protzen, J.-P. (1985). Inca quarrying and stonecutting. *Journal of the Society of Architectural Historians*, 44(2), 161–182. <https://doi.org/10.2307/990027>
- Qiaoping, Z., Deren, L., & Jianya, G. (2002). Shape similarity measures of linear entities. *Geo-Spatial Information Science*, 5(2), 62–67. doi:10.1007/bf02833888
- Rajabzadeh, S., & Sassone, M. (2016). Brick patterning on free-form surfaces. *Nexus Network Journal*, 19(1), 5–25. doi:10.1007/s00004-016-0305-9

- Rojas, C. (2016, December 28). *European Council and Council of the European Union / Samyn and Partners*. ArchDaily. <https://www.archdaily.com/802379/european-council-and-council-of-the-european-union-samyn-and-partners>
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). Imagenet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3), 211–252. <https://doi.org/10.1007/s11263-015-0816-y>
- Sanabria, S. L. (1989). From Gothic to Renaissance stereotomy: The design methods of Philibert de l'Orme and Alonso de Vandelvira. *Technology and Culture*, 30(2), 266. <https://doi.org/10.2307/3105105>
- Šaravanja, K., Oreč, F., & Kopilaš, V. (2023). Giant Stone Buildings (III) - cyclopean and other buildings of more developed architecture. *E-Zbornik, Elektronički Zbornik Radova Građevinskog Fakulteta*, 13(26), 76–99. <https://doi.org/10.47960/2232-9080.2023.26.13.76>
- Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR International Conference on Learning Representations* 2015. <https://doi.org/https://doi.org/10.48550/arXiv.1409.1556>
- Squier, E. G. (1877). *Peru: Incidents of travel and exploration in the land of the Incas*. Harper & Bros.
- Szabó, S., Funari, M. F., & Lourenço, P. B. (2023). Masonry patterns' influence on the damage assessment of urm walls: Current and future trends. *Developments in the Built Environment*, 13, 100119. <https://doi.org/10.1016/j.dibe.2023.100119>
- Thamboo, J. A., Zahra, T., & Dhanasekar, R. (2020). Development of Design Methodology for mortarless masonry system: Case study – A resettlement housing colony. *Journal of Building Engineering*, 27, 100973. <https://doi.org/10.1016/j.jobbe.2019.100973>

- Thangavelu, V., Liu, Y., Saboia, M., & Napp, N. (2018). Dry stacking for automated construction with irregular objects. *2018 IEEE International Conference on Robotics and Automation (ICRA)*. <https://doi.org/10.1109/icra.2018.8460562>
- Varela, P. A., & Sousa, J. P. (2016). Revising stereotomy through digital technology. *eCAADe Proceedings*, 2, 427–434. <https://doi.org/10.52842/conf.ecaade.2016.2.427>
- Varela, P. A., & Sousa, J. P. (2019). Digital expansion of stereotomy a semantic classification. *Blucher Design Proceedings*. https://doi.org/10.5151/proceedings-ecaadesigradi2019_340
- Veltkamp, R. C. (2001). Shape matching: Similarity measures and Algorithms. *Proceedings International Conference on Shape Modeling and Applications*, 188–197. doi:10.1109/sma.2001.923389
- Wang, Q., Pantoja-Rosero, B. G., Santos, K. R. M., & Beyer, K. (2024). An image convolution-based method for the irregular stone packing problem in masonry wall construction. *European Journal of Operational Research*, 316(2), 733–753. doi:10.1016/j.ejor.2024.01.037
- Wermelinger, M., Furrer, F., Yoshida, H., Gramazio, F., Kohler, M., Siegwart, R., & Hutter, M. (2018). Greedy stone tower creations with a robotic arm. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. <https://doi.org/10.24963/ijcai.2018/760>
- Willmann, J., Augugliaro, F., Cadalbert, T., D'Andrea, R., Gramazio, F., & Kohler, M. (2012). Aerial robotic construction towards a new field of Architectural Research. *International Journal of Architectural Computing*, 10(3), 439–459. <https://doi.org/10.1260/1478-0771.10.3.439>
- Wilton, O., Barnett Howland, M., & Parker, T. (2025). Nested Stone Pasts and Futures: Stone reuse prototyping at St Leonard's hill. *Architecture, Structures and Construction*, 5(1). doi:10.1007/s44150-025-00149-z

- Witt, A. (2022). Through a machine lens: Case studies of Computer Vision and machine learning in design methodology. *Loci Communes*, 1(2), 1–16. <https://doi.org/10.31261/lc.2022.02.06>
- Wu, J. (2017). Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China.*
- Xue, T., Wu, M., Lu, L., Wang, H., Dong, H., & Chen, B. (2023). Learning gradient fields for scalable and generalizable irregular packing. *SIGGRAPH Asia 2023 Conference Papers*. <https://doi.org/10.1145/3610548.3618235>
- Yu, B., & Fingrut, A. (2022). Sustainable building design (SBD) with reclaimed wood library constructed in collaboration with 3D Scanning Technology in the UK. *Resources, Conservation and Recycling*, 186, 106566. doi:10.1016/j.resconrec.2022.106566
- Zandavali, B. A., & García, M. J. (2019). Automated Brick Pattern Generator for robotic assembly using machine learning and images. *Blucher Design Proceedings*. https://doi.org/10.5151/proceedings-ecaadesigradi2019_605
- Zhang, Y., Tatarintseva, L., Clewlow, T., Clark, E., Botsford, G., & Shea, K. (2023). A template design and automated parametric model for sustainable corbel dwellings with interlocking blocks. *Developments in the Built Environment*, 14, 100148. <https://doi.org/10.1016/j.dibe.2023.100148>
- Zhao, C., Shi, W., & Deng, Y. (2005). A new Hausdorff distance for image matching. *Pattern Recognition Letters*, 26(5), 581–586. doi:10.1016/j.patrec.2004.09.022
- Zhou, X., Zhao, G., Xing, Y., Wu, J., & Xiong, Z. (2023). Robotic stacking of irregular objects with load position identification and compensation. *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*. <https://doi.org/10.1109/case56687.2023.10260654>

APPENDICES

A. CSV File

Image No	Type	Mortar/Dry	Total Contour Number	Mean of Number of Edges	Std of Number of Edges	Mean of Contour Length	Std of Contour Length	Total Neighbourhood Number	Close Packed
1.jpg	Random Rubble	Mortar	17	9.058	3.114	229.124	68.898	60	No
2.jpg	Random Rubble	Mortar	17	12.764	3.718	245.691	108.736	68	No
3.jpg	Random Rubble	Mortar	14	13.142	4.223	252.662	71.236	52	No
4.jpg	Random Rubble	Mortar	15	10.933	3.837	249.117	51.904	54	No
5.jpg	Random Rubble	Dry	16	23.375	7.606	317.534	246.104	48	No
6.jpg	Random Rubble	Dry	4	23.75	4.493	544.628	89.155		No
7.jpg	Random Rubble	Dry	4	25.25	2.772	616.525	287.636	6	No
8.jpg	Square Rubble	Dry	8	21	6.11	350.39	285.764	20	Yes
10.jpg	Random Rubble	Mortar	17	20.588	4.538	244.18	83.696	68	No
11.jpg	Random Rubble	Mortar	7	19.714	4.62	428.822	184.373	18	No
14.jpg	Square Rubble	Dry	3	30.666	4.109	1082.066	847.258		Yes
16.jpg	Random Rubble	Dry	11	27	7.627	693.284	294.139	32	Yes
17.jpg	Random Rubble	Mortar	18	21.944	5.243	258.114	155.323	66	No
20.jpg	Random Rubble	Dry	21	22.904	7.302	313.031	219.714	66	Yes
22.jpg	Square Rubble	Mortar	23	8.13	3.138	203.635	75.462	98	No
27.jpg	Random Rubble	Dry	9	25.777	5.94	431.181	280.11	24	Yes
31.jpg	Random Rubble	Mortar	15	22.6	5.547	304.11	211.833	52	Yes
35.jpg	Random Rubble	Dry	40	19.725	3.84	178.694	92.427	126	Yes
38.jpg	Square Rubble	Dry	18	14.277	6.269	256.37	283.344	52	Yes
43.jpg	Random Rubble	Dry	43	19.441	3.774	183.765	87.449	168	Yes
45.jpg	Square Rubble	Dry	3	18.666	3.091	971.984	480.491	4	Yes
48.jpg	Random Rubble	Dry	7	10	4.105	335.638	168.318	20	No
49.jpg	Polygonal Rubble	Dry	21	16.666	3.563	239.104	169.899	58	Yes
50.jpg	Polygonal Rubble	Mortar	193	9.212	2.074	66.09	44.18		Yes
52.jpg	Random Rubble	Dry	5	15.2	3.487	480.865	166.624	12	No
53.jpg	Random Rubble	Dry	13	14	4.367	318.637	142.029	46	Yes
67.jpg	Random Rubble	Dry	6	14.666	4.384	433.895	89.457	18	No
68.jpg	Polygonal Rubble	Dry	4	22.5	5.408	621.766	445.553	10	Yes
71.jpg	Polygonal Rubble	Dry	7	21	4.956	463.47	241.102	16	Yes

76.jpg	Polygonal Rubble	Mortar	9	22.444	4.901	398.399	315.367	26	Yes
81.jpg	Random Rubble	Mortar	26	15.923	4.066	237.605	137.783	92	No
88.jpg	Polygonal Rubble	Mortar	37	14.216	4.387	163.708	203.147	122	No
90.jpg	Polygonal Rubble	Mortar	14	10.642	4.6844	262.241	99.93	46	No
91.jpg	Random Rubble	Dry	35	12.885	4.399	186.977	159.268	102	Yes
92.jpg	Random Rubble	Mortar	85	11.505	2.384	98.103	57.842	324	No
94.jpg	Square Rubble	Mortar	36	20.055	4.061	208.312	133.462	136	Yes
95.jpg	Polygonal Rubble	Mortar	13	11.692	3.624	314.533	128.044	46	Yes
96.jpg	Polygonal Rubble	Mortar	20	16.75	4.121	241.754	118.339	68	No
97.jpg	Random Rubble	Dry	45	8.444	2.155	151.785	106.111	134	Yes
100.jpg	Square Rubble	Dry	13	8.666	2.82	284.053	131.081	44	No
104.jpg	Random Rubble	Dry	26	11.884	3.423	198.423	166.458	78	No
106.jpg	Random Rubble	Mortar	30	21.666	4.57	228.067	144.889	120	No
108.jpg	Polygonal Rubble	Mortar	22	18.681	3.559	261.002	191.926	72	No
111.jpg	Polygonal Rubble	Mortar	9	12.777	3.644	490.426	263.955	28	Yes
114.jpg	Polygonal Rubble	Mortar	9	17.666	6.765	387.45	157.451	22	Yes
118.jpg	Random Rubble	Dry	23	14.739	4.988	238.106	149.547	72	Yes
120.jpg	Square Rubble	Mortar	59	23.033	5.0922	154.036	111.778	224	No
125.jpg	Polygonal Rubble	Mortar	56	17.964	3.489	154.68	92.17	202	Yes
128.jpg	Polygonal Rubble	Mortar	36	25.666	4.725	222.789	158.048	126	No
131.jpg	Random Rubble	Dry	94	24.265	5.128	114.417	93.428		No
134.jpg	Square Rubble	Dry	13	18.307	7.02	384.111	388.798	34	Yes
136.jpg	Square Rubble	Dry	6	14.333	5.153	685.329	483.172	10	Yes
137.jpg	Random Rubble	Dry	8	21.5	2.236	471.664	289.832	24	Yes
139.jpg	Random Rubble	Mortar	25	16.88	4.062	214.692	121.38	84	No
145.jpg	Random Rubble	Dry	17	18.647	5.8	322.543	178.273	54	No
146.jpg	Polygonal Rubble	Mortar	12	16.25	4.584	360.477	184.208	38	No
150.jpg	Square Rubble	Dry	7	15.857	3.043	504.474	409.444	18	No
151.jpg	Square Rubble	Dry	14	18.214	4.959	324.896	263.226	40	Yes
152.jpg	Square Rubble	Mortar	8	12.25	3.112	357.316	108.885	26	No
157.jpg	Random Rubble	Dry	24	14.833	3.555	215.981	149.976	76	No
168.jpg	Random Rubble	Mortar	35	23.142	3.757	241.453	164.666	122	No
177.jpg	Square Rubble	Mortar	22	32.272	5.94	282.076	200.317	78	Yes
178.jpg	Random Rubble	Dry	13	19.692	4.339	391.388	192.032	38	Yes
180.jpg	Random Rubble	Dry	11	23.181	3.432	441.794	387.793	32	No

181.jpg	Random Rubble	Dry	13	22.153	4.221	376.474	357.49	32	No
184.jpg	Random Rubble	Dry	6	22.5	3.149	438.084	384.642	16	No
185.jpg	Random Rubble	Dry	7	21.142	1.551	484.75	320.157	20	No
186.jpg	Random Rubble	Dry	16	17.933	4.249	261.716	289.488	40	No
189.jpg	Random Rubble	Dry	5	26.2	3.867	604.73	232.023	12	No
190.jpg	Random Rubble	Dry	8	18.125	3.55	422.17	164.002	24	No
191.jpg	Square Rubble	Dry	9	17.555	7.917	465.308	339.864	26	No
192.jpg	Square Rubble	Dry	10	22.2	5.946	391.309	207.433	30	Yes
193.jpg	Square Rubble	Dry	10	27.7	7.043	428.908	411.098	28	No
194.jpg	Random Rubble	Mortar	12	21.666	3.399	312.039	101.81	42	No
195.jpg	Random Rubble	Mortar	17	24.235	3.54	293.686	105.15	60	No
196.jpg	Random Rubble	Mortar	21	24.333	3.98	223.182	58.591	82	No
198.jpg	Random Rubble	Mortar	12	15.166	3.89	311.013	101.745	44	No
199.jpg	Random Rubble	Mortar	8	12	4.924	353.434	148.848	26	No
215.jpg	Random Rubble	Mortar	20	23.36	5.985	202.208	113.227	98	No
216.jpg	Random Rubble	Mortar	16	19.187	5.736	280.667	138.137	52	No
217.jpg	Random Rubble	Mortar	21	17.238	4.363	235.822	91.116	82	No
218.jpg	Random Rubble	Mortar	13	23.461	5.583	336.482	194.744	40	No
219.jpg	Random Rubble	Mortar	18	16.333	2.848	255.244	124.453	58	No
220.jpg	Random Rubble	Mortar	23	14.478	3.005	202.74	125.392	80	No
221.jpg	Random Rubble	Mortar	23	26.913	5.182	217.931	98.496	98	No
226.jpg	Random Rubble	Dry	25	20.16	6.85	252.4	234.85	70	Yes
227.jpg	Random Rubble	Dry	23	18.391	6.329	271.272	175.354	88	Yes
228.jpg	Random Rubble	Dry	25	18.64	5.705	230.89	96.713	86	Yes
238.jpg	Random Rubble	Dry	12	25.583	6.304	376.87	226.576	36	Yes
240.jpg	Random Rubble	Dry	7	23.714	5.146	422.17	224.363	20	Yes
244.jpg	Random Rubble	Mortar	19	19.631	5.686	262.279	133.312	72	Yes
245.jpg	Random Rubble	Mortar	25	20.2	5.238	213.036	140.472	86	Yes
246.jpg	Random Rubble	Mortar	21	21.952	5.979	256.689	164.841	62	Yes
247.jpg	Random Rubble	Mortar	17	20.823	6.308	268.872	139.69	60	Yes
248.jpg	Random Rubble	Mortar	21	15.142	5.54	221.522	123.664	74	Yes
258.jpg	Random Rubble	Dry	37	19.324	3.464	195.713	134.733	118	Yes
259.jpg	Random Rubble	Dry	37	19.783	4.412	197.517	126.785	128	Yes
260.jpg	Random Rubble	Dry	33	20.939	3.773	216.222	140.517	118	Yes
264.jpg	Square Rubble	Dry	19	15.368	4.544	245.639	153.008	58	No
265.jpg	Square Rubble	Dry	19	16.105	3.522	249.253	160.914	72	No
266.jpg	Square Rubble	Dry	17	15.823	5.436	284.736	237.284	54	No
269.jpg	Random Rubble	Dry	43	18.093	4.481	178.473	130.14	160	No
270.jpg	Random Rubble	Dry	47	19.382	4.045	170.587	106.819	176	No
271.jpg	Random Rubble	Dry	45	19.755	4.492	173.903	132.896	158	No
273.jpg	Square Rubble	Dry	5	20.6	6.311	588.591	229.506	12	Yes
278.jpg	Square Rubble	Dry	5	23.4	4.758	657.794	277.064	14	Yes

281.jpg	Square Rubble	Dry	4	23.25	7.119	828.664	293.763	6	Yes
282.jpg	Square Rubble	Dry	4	24	8.185	691.463	389.247	6	Yes
283.jpg	Square Rubble	Dry	5	23.8	4.534	665.466	257.07	14	Yes
294.jpg	Random Rubble	Mortar	7	13.71	4.712	452.487	260.708	18	No
295.jpg	Random Rubble	Mortar	7	10	2.138	387.97	112.625	20	No
296.jpg	Random Rubble	Mortar	7	15.285	5.443	413.172	131.951	22	No
297.jpg	Polygonal Rubble	Dry	16	23.062	4.264	327.936	248.276	40	Yes
300.jpg	Polygonal Rubble	Dry	21	22.142	3.313	253.953	218.183	64	Yes
301.jpg	Polygonal Rubble	Dry	19	21.894	3.998	287.651	207.419	68	Yes
315.jpg	Random Rubble	Dry	3	12.666	1.247	747.497	370.613	6	Yes
316.jpg	Random Rubble	Dry	5	19.2	8.231	565.197	339.028	14	No
317.jpg	Random Rubble	Dry	4	15.75	3.031	681.864	315.516	10	No
318.jpg	Random Rubble	Dry	13	15.153	4.4	318.272	185.985	32	Yes
319.jpg	Random Rubble	Dry	14	14.571	4.03	317.404	223.817	42	Yes
320.jpg	Random Rubble	Dry	12	14.416	3.925	379.112	159.156	40	Yes
391.jpg	Random Rubble	Dry	7	15.571	5.038	401.957	70.623	22	No
392.jpg	Polygonal Rubble	Dry	5	19.4	2.244	643.771	134.314	12	Yes
393.jpg	Polygonal Rubble	Dry	5	19.8	2.638	570.472	347.333	10	Yes
404.jpg	Polygonal Rubble	Dry	5	21.8	3.249	613.376	482.461	10	Yes
405.jpg	Polygonal Rubble	Dry	8	18.625	3.772	477.324	279.141	24	Yes
417.jpg	Polygonal Rubble	Mortar	18	21.666	6.173	303.671	151.663	56	No
428.jpg	Random Rubble	Mortar	14	16.714	3.368	393.449	215.393	40	No
429.jpg	Random Rubble	Mortar	35	16.542	4.052	170.288	122.63	116	No
433.jpg	Random Rubble	Mortar	19	16.105	5.802	282.578	146.056	62	No
434.jpg	Random Rubble	Mortar	21	16.952	4.088	266.474	165.307	72	No
435.jpg	Random Rubble	Mortar	19	16.315	5.068	306.499	159.684	62	No
438.jpg	Random Rubble	Mortar	22	16.727	3.99	254.366	116.335	74	No
459.jpg	Polygonal Rubble	Mortar	32	14.625	4.255	181.42	200.071	112	No
460.jpg	Polygonal Rubble	Mortar	27	15.888	3.744	226.901	244.16	88	No
461.jpg	Polygonal Rubble	Mortar	29	13.551	4.832	177.767	218.184	94	No
465.jpg	Polygonal Rubble	Mortar	11	9.181	2.917	268.032	152.279	36	No
466.jpg	Polygonal Rubble	Mortar	12	10.083	4.838	281.927	97.652	40	No
467.jpg	Polygonal Rubble	Mortar	12	8.083	4.071	276.134	106.509	38	No

468.jpg	Random Rubble	Mortar	92	10.836	2.671	84.886	53.463		No
469.jpg	Random Rubble	Mortar	88	11.068	2.521	91.21	76.74		No
471.jpg	Random Rubble	Mortar	101	11.138	2.782	82.199	53.845		No
472.jpg	Random Rubble	Mortar	80	10.287	2.734	92.645	69.488		No
474.jpg	Square Rubble	Mortar	43	20.046	3.549	187.775	91.79	180	Yes
475.jpg	Square Rubble	Mortar	45	19.822	3.591	182.783	114.507	174	Yes
476.jpg	Polygonal Rubble	Mortar	16	10	3.872	238.473	84.435	56	No
477.jpg	Polygonal Rubble	Mortar	14	10.857	2.799	285.324	124.031	46	No
478.jpg	Polygonal Rubble	Mortar	17	9.529	3.256	246.864	95.865	60	No
479.jpg	Polygonal Rubble	Mortar	13	12.153	4.417	310.654	182.973	42	No
480.jpg	Polygonal Rubble	Mortar	14	11.928	3.534	304.253	216.007	48	No
481.jpg	Random Rubble	Dry	45	8.444	2.049	140.882	84.572	156	No
482.jpg	Random Rubble	Dry	41	9.243	2.303	163.964	115.65	144	No
483.jpg	Random Rubble	Dry	50	9.28	2.097	147.464	84.29	176	No
486.jpg	Square Rubble	Dry	15	9.066	3.275	270.398	152.477	52	No
487.jpg	Square Rubble	Dry	11	9.272	2.219	363.982	182.815	26	No
488.jpg	Square Rubble	Dry	11	7.818	1.991	351.426	185.327	34	No
493.jpg	Square Rubble	Dry	11	8.09	2.678	363.22	207.993	30	No
494.jpg	Square Rubble	Dry	10	8.3	1.615	346.442	240.086	30	No
495.jpg	Square Rubble	Dry	14	8.357	2.286	286.707	185.44	42	No
496.jpg	Square Rubble	Dry	13	8	2.287	300.051	169.205	38	No
497.jpg	Square Rubble	Dry	15	8.666	2.699	295.819	94.089	50	No
499.jpg	Random Rubble	Dry	25	13.72	4.005	206.735	190.448	90	No
500.jpg	Random Rubble	Dry	30	12.533	4.216	178.752	130.5	104	No
501.jpg	Random Rubble	Dry	22	13.409	4.417	231.76	140.031	64	No
502.jpg	Random Rubble	Dry	23	14.304	4.946	194.891	103.242	86	No
503.jpg	Random Rubble	Mortar	36	19.388	3.506	178.186	120.985	148	No
504.jpg	Random Rubble	Mortar	32	21.812	4.713	206.974	142.519	108	No
506.jpg	Random Rubble	Mortar	35	20.942	3.92	193.208	105.097	142	No
507.jpg	Random Rubble	Mortar	40	20.425	4.247	166.678	55.1	172	No
508.jpg	Random Rubble	Mortar	35	22.085	4.901	189.558	121.911	128	No
509.jpg	Polygonal Rubble	Mortar	10	12.1	3.78	460.98	218.973	32	Yes
510.jpg	Polygonal Rubble	Mortar	11	14.727	5.0288	406.802	175.367	36	Yes
511.jpg	Polygonal Rubble	Mortar	11	11.181	3.588	365.54	225.474	30	Yes
513.jpg	Polygonal Rubble	Mortar	12	12.166	2.91	328.586	142.527	40	Yes
520.jpg	Random Rubble	Mortar	47	21.765	5.419	167.984	176.762	164	No

521.jpg	Square Rubble	Mortar	49	22.632	5.829	170.241	132.886	168	No
522.jpg	Square Rubble	Mortar	47	21.957	4.819	162.717	135.614	170	No
523.jpg	Polygonal Rubble	Mortar	52	16.692	4.116	151.272	112.074	196	Yes
524.jpg	Polygonal Rubble	Mortar	56	18.107	4.173	155.566	101.822	202	Yes
528.jpg	Random Rubble	Dry	107	23.626	5.53	107.319	80.705		No
529.jpg	Random Rubble	Dry	105	23.885	4.848	106.101	80.323		No
531.jpg	Square Rubble	Dry	14	19.214	5.492	409.847	405.241	42	Yes
534.jpg	Square Rubble	Dry	11	16.909	2.874	435.073	236.736	30	Yes
536.jpg	Square Rubble	Dry	11	15.09	6.855	434.223	374.988	14	Yes
537.jpg	Square Rubble	Dry	9	14.111	3.899	485.364	289.727	26	Yes
538.jpg	Square Rubble	Dry	12	14.25	3.654	351.381	271.446	26	Yes
539.jpg	Square Rubble	Dry	8	12.25	3.381	534.39	217.848	22	Yes
540.jpg	Square Rubble	Dry	12	17.75	7.119	416.206	362.69	40	Yes
541.jpg	Square Rubble	Dry	11	12.727	3.62	426.265	331.608	32	Yes
542.jpg	Square Rubble	Dry	11	13.454	3.798	413.88	346.727	20	Yes
543.jpg	Square Rubble	Dry	10	14.9	5.889	443.639	420.798	30	Yes
544.jpg	Square Rubble	Dry	12	13	5.196	426.665	240.796	36	Yes
547.jpg	Random Rubble	Dry	9	22.555	4.03	463.623	177.818	30	Yes
548.jpg	Random Rubble	Dry	11	22.545	4.519	408.441	231.218	28	Yes
549.jpg	Random Rubble	Dry	13	20.769	3.703	307.31	185.039	46	Yes
550.jpg	Random Rubble	Dry	14	20.785	4.211	306.026	187.389	46	Yes
551.jpg	Random Rubble	Dry	18	21.333	3.726	243.373	175.171	54	Yes
552.jpg	Random Rubble	Dry	14	21.571	3.83	290.829	179.406	42	Yes
553.jpg	Random Rubble	Mortar	30	17.033	4.094	209.511	101.057	116	No
554.jpg	Random Rubble	Mortar	21	19	3.251	249.198	150.86	76	No
555.jpg	Random Rubble	Mortar	24	18.541	3.915	227.606	133.972	86	No
577.jpg	Random Rubble	Dry	16	20.75	3.75	351.168	177.817	56	No
578.jpg	Random Rubble	Dry	14	19.285	3.594	374.663	212.401	44	No
580.jpg	Polygonal Rubble	Mortar	11	18.727	2.339	428.209	216.324	34	No
582.jpg	Polygonal Rubble	Mortar	16	18.062	5.436	331.384	162.897	56	No
586.jpg	Square Rubble	Dry	9	15.666	4.807	398.6	301.234	26	No
587.jpg	Square Rubble	Dry	7	16.142	3.09	374.315	264.929	20	No
588.jpg	Square Rubble	Dry	7	15.285	3.41	534.683	319.578	20	No
589.jpg	Square Rubble	Dry	7	12	2.927	532.438	331.103	18	No
590.jpg	Square Rubble	Dry	12	15.5	3.64	279.504	293.506	30	No
591.jpg	Square Rubble	Dry	10	15.1	4.369	330.47	287.43	24	Yes
592.jpg	Square Rubble	Dry	11	15.909	4.925	404.576	284.557	34	No
593.jpg	Square Rubble	Dry	14	16.214	4.857	298.408	253.675	40	Yes
594.jpg	Square Rubble	Dry	12	15.75	3.744	381.37	279.668	36	No

595.jpg	Square Rubble	Dry	9	15.111	4.556	425.496	272.448	24	Yes
598.jpg	Square Rubble	Dry	8	18	2.061	468.03	287.205	22	No
600.jpg	Square Rubble	Dry	4	13	5.477	782.812	508.417	12	Yes
601.jpg	Square Rubble	Dry	10	15.1	3.419	405.129	268.835	32	No
602.jpg	Square Rubble	Dry	7	16.142	2.641	428.729	322.745	14	No
603.jpg	Square Rubble	Dry	6	17.5	3.304	540.374	193.341	18	No
607.jpg	Square Rubble	Mortar	7	12	2.672	464.653	178.253	22	No
608.jpg	Square Rubble	Mortar	6	11.666	3.543	495.348	297.931	18	No
609.jpg	Square Rubble	Mortar	6	13.428	3.499	426.644	210.543	22	No
610.jpg	Square Rubble	Mortar	8	11.75	3.897	339.788	74.285	28	No
612.jpg	Square Rubble	Mortar	8	12.5	1.936	315	116.699	26	No
613.jpg	Square Rubble	Mortar	9	12.222	3.292	292.492	106.278	32	No
614.jpg	Square Rubble	Mortar	11	12	5.393	281.306	105.542	34	No
615.jpg	Square Rubble	Mortar	8	14	2.915	363.276	103.376	24	No
616.jpg	Square Rubble	Mortar	9	13	2.905	304.592	135.607	28	No
617.jpg	Square Rubble	Mortar	10	13.3	2.758	296.605	117.462	32	No
618.jpg	Square Rubble	Mortar	10	13.2	3.682	329.558	149.414	32	No
619.jpg	Square Rubble	Mortar	9	13	2.943	356.401	145.535	28	No
644.jpg	Random Rubble	Dry	11	12.818	3.809	361.018	286.864	38	No
645.jpg	Random Rubble	Dry	14	15.071	3.058	299.326	177.196	54	No
689.jpg	Random Rubble	Mortar	37	22.891	4.164	228.056	145.8	136	No
690.jpg	Random Rubble	Mortar	35	22.571	4.883	233.397	180.131	140	No
691.jpg	Random Rubble	Mortar	40	21.825	4.732	207.222	147.125	148	No
713.jpg	Square Rubble	Mortar	20	31.7	4.807	278.022	153.393	90	Yes
714.jpg	Square Rubble	Mortar	24	28.958	4.41	237.193	143.282	86	Yes
715.jpg	Square Rubble	Mortar	21	31.523	5.611	262.92	357.578	68	Yes
716.jpg	Polygonal Rubble	Mortar	11	12.09	6.487	323.671	103.439	36	No
717.jpg	Polygonal Rubble	Mortar	12	15.166	6.817	324.292	153.917	44	No
718.jpg	Polygonal Rubble	Mortar	11	13.181	5.556	323.863	86.961	36	No
719.jpg	Polygonal Rubble	Mortar	8	10.5	5.766	347.452	107.672	26	No
720.jpg	Polygonal Rubble	Mortar	10	14.3	5	354.647	173.633	34	No
721.jpg	Polygonal Rubble	Mortar	9	8.666	3.829	317.331	94.801	32	No
722.jpg	Polygonal Rubble	Mortar	9	12.777	5.39	337.724	85.509	30	No
723.jpg	Polygonal Rubble	Mortar	12	12.166	5.382	308.745	124.912	44	No
724.jpg	Polygonal Rubble	Mortar	10	8.3	1.486	293.724	162.971	32	No

725.jpg	Polygonal Rubble	Mortar	7	11.857	4.356	353.615	167.366	20	Yes
726.jpg	Polygonal Rubble	Mortar	7	16.285	5.521	431.738	137.918	18	Yes
727.jpg	Polygonal Rubble	Mortar	7	13.428	5.368	370.742	168.151	20	Yes
728.jpg	Polygonal Rubble	Mortar	6	8.666	3.858	400.043	166.359	18	Yes
729.jpg	Polygonal Rubble	Mortar	60	18.683	5.875	151.603	64.42	280	Yes
730.jpg	Polygonal Rubble	Mortar	62	18.919	5.798	137.391	57.935	274	Yes
731.jpg	Polygonal Rubble	Mortar	65	19.984	6.207	145.029	67.284	278	Yes
732.jpg	Polygonal Rubble	Mortar	69	17.434	5.92	124.998	37.353	312	Yes
733.jpg	Polygonal Rubble	Mortar	50	18.38	5.141	170.71	104.216	224	Yes
734.jpg	Polygonal Rubble	Mortar	44	13.681	3.812	154.821	54.157	182	Yes
735.jpg	Polygonal Rubble	Mortar	53	15.075	4.496	140.583	65.277	230	Yes
736.jpg	Polygonal Rubble	Mortar	52	16.173	4.933	144.128	84.423	222	Yes
737.jpg	Polygonal Rubble	Mortar	58	15.568	4.909	136.264	79.579	228	Yes
738.jpg	Polygonal Rubble	Mortar	51	15.803	4.238	148.894	102.715	204	Yes
739.jpg	Polygonal Rubble	Mortar	41	15.024	4.105	173.066	153.695	170	Yes
740.jpg	Polygonal Rubble	Mortar	51	16.745	5.666	151.551	114.058	206	Yes
741.jpg	Polygonal Rubble	Mortar	59	16.372	4.46	130.487	61.719	244	Yes
742.jpg	Polygonal Rubble	Mortar	57	15.842	5.053	130.43	46.25	250	Yes
743.jpg	Polygonal Rubble	Mortar	55	15.781	4.72	145.718	88.356	220	Yes
744.jpg	Polygonal Rubble	Mortar	43	15.976	4.985	166.01	103.141	180	Yes
745.jpg	Polygonal Rubble	Mortar	45	15.577	4.019	161.272	163.284	172	Yes
746.jpg	Polygonal Rubble	Mortar	58	15.086	5.276	134.172	59.507	242	Yes
747.jpg	Polygonal Rubble	Mortar	48	15.416	4.563	156.319	84.274	206	Yes
748.jpg	Polygonal Rubble	Mortar	47	16.085	4.694	167.178	152.076	184	Yes
749.jpg	Polygonal Rubble	Mortar	47	17.297	5.206	171.347	141.539	184	Yes

750.jpg	Polygonal Rubble	Mortar	61	17.54	4.853	135.34	61.201	256	Yes
751.jpg	Polygonal Rubble	Mortar	50	16.04	3.789	153.196	102.988	206	Yes
752.jpg	Polygonal Rubble	Mortar	53	14.792	3.803	140.431	81.607	222	Yes
753.jpg	Polygonal Rubble	Mortar	42	15.023	4.328	172.318	127.63	172	Yes
763.jpg	Random Rubble	Mortar	5	18.6	2.416	555.636	276.484	10	Yes
767.jpg	Polygonal Rubble	Mortar	20	15.95	4.283	272.29	233.406	74	Yes
768.jpg	Polygonal Rubble	Mortar	31	13.225	4.903	181.933	68.187	118	Yes
769.jpg	Polygonal Rubble	Mortar	35	14.2	4.438	176.391	74.915	146	Yes
770.jpg	Polygonal Rubble	Mortar	38	13.578	5.441	155.647	75.408	148	Yes
771.jpg	Random Rubble	Mortar	186	14.672	3.506	63.375	43.425		No
775.jpg	Random Rubble	Mortar	9	17.555	4.003	367.11	140.905	28	Yes
776.jpg	Random Rubble	Mortar	11	19.181	4.238	308.855	125.93	38	Yes
778.jpg	Random Rubble	Mortar	5	13.6	3.878	543.436	215.064	14	Yes
781.jpg	Polygonal Rubble	Mortar	5	14.8	1.166	616.446	246.075	14	No
791.jpg	Polygonal Rubble	Mortar	5	10.4	3.261	510.216	283.457	12	No
794.jpg	Polygonal Rubble	Mortar	6	11	2.38	469.331	220.03	16	No
796.jpg	Polygonal Rubble	Mortar	5	12.8	2.638	542.02	287.291	10	No
797.jpg	Polygonal Rubble	Mortar	7	10.571	4.169	374.693	275.693	16	No
804.jpg	Polygonal Rubble	Mortar	5	10.2	1.166	527.088	203.477	12	No
823.jpg	Random Rubble	Mortar	14	16.857	5.012	289.773	200.173	42	Yes
824.jpg	Random Rubble	Mortar	19	14.578	5.071	251.748	144.042	66	Yes
825.jpg	Random Rubble	Mortar	15	22.466	4.544	303.182	242.49	44	Yes
826.jpg	Random Rubble	Mortar	18	20.944	3.851	264.558	153.682	54	Yes
827.jpg	Random Rubble	Mortar	9	20.555	3.435	412.618	229.674	30	Yes
828.jpg	Random Rubble	Mortar	16	20.75	3.976	271.201	147.914	54	Yes
829.jpg	Random Rubble	Mortar	12	21.166	6.555	345.982	150.698	40	Yes
830.jpg	Random Rubble	Mortar	17	21.47	6.25	292.566	209.638	54	Yes
861.jpg	Random Rubble	Mortar	3	7.333	1.247	570.213	63.944	6	No
862.jpg	Random Rubble	Mortar	5	8.8	2.481	428.997	156.978	12	No
863.jpg	Random Rubble	Mortar	2	6.5	1.5	727.181	192.769	2	No
864.jpg	Random Rubble	Mortar	2	9.5	4.5	759.545	227.662	2	No
865.jpg	Random Rubble	Mortar	8	11.25	7.758	368.953	125.393	24	No
866.jpg	Random Rubble	Mortar	5	12	2.607	472.614	86.414	14	No

867.jpg	Random Rubble	Mortar	5	10.4	4.223	501.103	293.787	14	No
868.jpg	Random Rubble	Mortar	6	13.166	4.487	427.226	159.045	18	No
869.jpg	Random Rubble	Mortar	8	8	2.645	309.438	155.929	24	No
870.jpg	Random Rubble	Mortar	4	11.5	5.545	543.864	95.438	10	No
872.jpg	Random Rubble	Mortar	5	10.4	1.854	385.948	262.074	14	No
873.jpg	Random Rubble	Mortar	4	9	1.732	517.524	305.357	14	No
874.jpg	Random Rubble	Mortar	3	9.666	2.624	527.177	283.364	4	No
875.jpg	Random Rubble	Mortar	4	9.25	4.602	527.196	214.507	10	No
877.jpg	Polygonal Rubble	Mortar	78	16.397	3.494	127.588	85.016	346	Yes
884.jpg	Square Rubble	Mortar	32	13.062	5.08	213.994	99.639	114	Yes
885.jpg	Square Rubble	Mortar	33	14.757	5.624	211.571	95.67	124	Yes
886.jpg	Square Rubble	Dry	34	21.147	4.722	239.533	185.84	112	Yes
888.jpg	Square Rubble	Mortar	15	14.8	2.902	318.379	152.352	52	No
897.jpg	Square Rubble	Mortar	25	8.8	3.464	206.967	61.677	92	No
908.jpg	Square Rubble	Mortar	5	5	1.264	456.799	149.697	12	Yes
913.jpg	Square Rubble	Mortar	16	15.812	4.049	275.347	116.892	62	No
935.jpg	Random Rubble	Mortar	5	13.2	0.748	526.673	103.528	14	No
937.jpg	Polygonal Rubble	Mortar	7	10.142	2.948	386.223	201.539	18	No
938.jpg	Polygonal Rubble	Mortar	6	8.5	2.986	442.053	99.894	18	No
939.jpg	Polygonal Rubble	Mortar	7	10.571	3.155	391.119	248.979	20	No
940.jpg	Polygonal Rubble	Mortar	7	9.714	2.49	323.115	184.782	22	No
941.jpg	Square Rubble	Mortar	40	9.2	2.984	145.116	47.536	176	Yes
944.jpg	Square Rubble	Dry	18	14.222	4.13	264.648	141.164	64	Yes
945.jpg	Square Rubble	Dry	20	13.3	4.196	244.56	96.003	80	Yes
946.jpg	Square Rubble	Dry	16	12.25	3.881	287.068	151.57	56	Yes
947.jpg	Square Rubble	Dry	21	12.095	3.435	244.007	79.329	74	Yes
948.jpg	Random Rubble	Mortar	11	13.818	3.712	300.315	115.335	40	No
949.jpg	Random Rubble	Mortar	12	12.416	3.569	264.209	120.945	38	No
951.jpg	Random Rubble	Mortar	9	9	2.449	301.438	94.3	28	No
952.jpg	Polygonal Rubble	Mortar	19	11.894	4.076	227.918	89.029	62	Yes
953.jpg	Polygonal Rubble	Mortar	18	12.388	4.436	248.33	98.143	58	Yes
954.jpg	Polygonal Rubble	Mortar	19	14.684	4.542	260.982	119.984	70	Yes
955.jpg	Polygonal Rubble	Mortar	20	15.4	5.994	235.671	121.323	68	Yes
956.jpg	Random Rubble	Mortar	12	10.5	3.013	350.645	217.511	34	Yes
957.jpg	Random Rubble	Mortar	13	11.307	3.428	290.239	205.278	44	Yes
958.jpg	Random Rubble	Mortar	9	13	2.581	413.968	233.707	26	Yes

959.jpg	Random Rubble	Mortar	8	11.75	4.71	309.31	248.225	24	Yes
961.jpg	Random Rubble	Mortar	7	13.714	2.813	505.786	393.263	18	Yes
963.jpg	Polygonal Rubble	Dry	7	11.428	3.155	425.781	201.727	18	Yes
964.jpg	Polygonal Rubble	Dry	8	10	3.316	415.356	87.935	26	Yes
965.jpg	Polygonal Rubble	Dry	5	11.4	1.743	561.698	348.959	14	Yes
966.jpg	Polygonal Rubble	Dry	6	9.166	1.343	455.841	179.575	18	Yes
967.jpg	Square Rubble	Mortar	50	8.16	2.737	130.397	38.753	216	Yes
968.jpg	Polygonal Rubble	Mortar	107	12.635	3.07	94.28	37.133		Yes
969.jpg	Random Rubble	Mortar	13	12	3.088	284.284	65.179	50	No
970.jpg	Random Rubble	Mortar	16	11.562	3.081	236.667	85.635	64	No
971.jpg	Random Rubble	Mortar	13	9.23	3.744	262.557	75.617	44	No
972.jpg	Random Rubble	Mortar	14	9.428	2.993	237.322	101.974	54	No
973.jpg	Polygonal Rubble	Mortar	116	8.543	2.32	81.017	25.446		Yes
974.jpg	Polygonal Rubble	Mortar	23	12.913	3.832	227.092	68.087	96	Yes
975.jpg	Square Rubble	Dry	18	11.111	3.725	313.793	162.621	50	Yes
976.jpg	Square Rubble	Dry	18	12	3.431	307.521	195.061	58	Yes
977.jpg	Square Rubble	Dry	11	13.181	4.195	498.572	246.192	30	Yes
978.jpg	Square Rubble	Dry	14	12.071	2.987	320.045	244.158	42	Yes
991.jpg	Square Rubble	Dry	7	10.857	4.189	502.792	294.999	20	Yes
992.jpg	Square Rubble	Dry	9	12.333	4.216	418.091	245.558	30	Yes
993.jpg	Square Rubble	Dry	8	12	3.391	433.787	424.586	22	Yes
994.jpg	Square Rubble	Dry	6	10.833	2.544	514.161	209.255	12	Yes
995.jpg	Square Rubble	Dry	7	10.428	2.969	508.164	220.317	18	Yes
1010.jpg	Square Rubble	Dry	13	15.23	3.64	339.849	183.442	40	Yes
1016.jpg	Square Rubble	Mortar	21	16.238	3.702	338.529	159.405	62	Yes
1018.jpg	Square Rubble	Mortar	18	17	3	367.731	196.672	54	Yes
1019.jpg	Square Rubble	Mortar	21	16.428	313.933	138.994	138.994	78	Yes
1022.jpg	Square Rubble	Dry	24	10.791	3.593	290.742	208.231	74	Yes
1024.jpg	Square Rubble	Mortar	24	9.833	4.149	206.388	66.67	100	Yes
1025.jpg	Square Rubble	Mortar	25	10.56	3.909	202.979	87.88	104	Yes
1026.jpg	Square Rubble	Mortar	19	10	3.684	233.364	95.643	74	Yes
1027.jpg	Square Rubble	Dry	52	13.269	4.678	180.508	101.801	186	Yes
1028.jpg	Square Rubble	Dry	50	12.64	4.013	177.073	105.166	188	Yes
1029.jpg	Square Rubble	Dry	46	12.934	4.244	187.984	117.28	166	Yes
1030.jpg	Square Rubble	Dry	53	12.773	4.436	179.774	100.136	186	Yes
1031.jpg	Square Rubble	Dry	47	13.063	4.483	192.053	103.099	168	Yes
1032.jpg	Square Rubble	Dry	48	13.5	4.65	189.169	124.143	162	Yes

1033.jpg	Square Rubble	Dry	53	13.132	4.817	176.414	113.37	202	Yes
1034.jpg	Square Rubble	Dry	48	13.541	4.898	187.315	107.523	178	Yes
1035.jpg	Square Rubble	Mortar	27	11.481	5.028	220.741	101.164	102	Yes
1036.jpg	Square Rubble	Mortar	23	7.695	3.367	215.903	81.326	80	Yes
1037.jpg	Square Rubble	Mortar	16	7.187	3.263	258.185	153.979	48	Yes
1038.jpg	Square Rubble	Mortar	25	8.72	3.292	220.797	116.436	98	Yes
1050.jpg	Polygonal Rubble	Mortar	14	12.642	4.789	283.862	202.978	50	Yes
1051.jpg	Polygonal Rubble	Mortar	16	10.625	2.781	227.457	81.718	54	Yes
1052.jpg	Polygonal Rubble	Mortar	15	9.666	2.086	245.618	150.013	52	Yes
1053.jpg	Polygonal Rubble	Mortar	15	9.133	2.526	249.898	138.005	48	Yes
1054.jpg	Polygonal Rubble	Mortar	37	8.081	2.018	154.642	39.613	164	Yes
1055.jpg	Polygonal Rubble	Mortar	34	8.558	2.403	164.182	52.804	144	Yes
1056.jpg	Polygonal Rubble	Mortar	41	8.536	2.47	144.397	52.206		Yes

B. Images

Images are available at: <https://github.com/emire-nr/Masonry-Wall-Images-and-Edge-Detections/tree/main>