DISS. ETH NO. 25621

EVALUATING SYMMETRY AND ORDER IN URBAN DESIGN

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zürich)

presented by

ARTEM CHIRKIN
MSc., University of Amsterdam
MSc., ITMO University, St.Petersburg

born on 16.01.1990
citizen of Russian Federation

accepted on the recommendation of

Prof. Dr. G. Schmitt
Prof. Dr. Y. Liu

2019

ABSTRACT

Symmetry and order are among the fundamental principles in architecture and urban design. Although symmetry and order are often associated with each other, the ways we think about the two are different. Symmetry has a precise mathematical definition; thus, in principle, it is possible to compute it. Order, in contrast, is a very vague and subjective notion. The goal of this work is to establish a link between computable symmetry and perception of order in the context of urban design.

The full complexity of interaction between symmetry, order, and perception in real-world urban design is beyond the scope of the presented thesis. Therefore, I have put my best efforts to narrow down the scope of the research topic and to rule out insignificant or controversial details. As a result, the presented work consists of four parts.

First, I limit the analysis of urban design to two-dimensional top-down views. I present Quick Urban Analysis Kit (qua-kit) – a tool that has been used as an online urban design exercise. It has allowed me to collect design data in a machine readable format specific to the needs of the study.

Second, I develop an efficient symmetry detection approach that processes edges in a design drawing to find potential symmetry patterns and measure their significance. This part of the work serves as a computational foundation of the analysis.

Third, I conduct an online survey experiment to asses the perception of order in design drawings. Using the results of the experiment, I create a statistical model of the perceived order measure.

Last, I build a regression model connecting computed symmetry and perceived order and conclude the work by analyzing the results. I show that the model agrees with the perception of order in the collected set of design drawings. I estimate the uncertainty of choices in the conducted experiment and assess the accuracy of the model predictions. Finally, I provide the evidence explaining the structure of dependence between different types of symmetry and the perceived order measure.

ZUSAMMENFASSUNG

Symmetrie und Ordnung sind fundamentale Prinzipien in Architektur und Stadtplanung. Und obwohl Symmetrie und Ordnung oft miteinander in Verbindung gebracht werden, ist die Art wie wir über beide denken unterschiedlich. Symmetrie hat eine präzise mathematische Definition; dadurch ist es, im Prinzip, möglich sie zu berechnen. Ordnung, im Gegensatz dazu, ist ein sehr vager und subjektiver Begriff. Das Ziel dieser Arbeit ist es, eine Verbindung zwischen berechenbarer Symmetrie und der wahrgenommenen Ordnung im Kontext der Stadtplanung herzustellen.

Die gesamte Komplexität der Verbindungen zwischen Symmetrie, Ordnung und Wahrnehmung in einer angewandten Stadtplanung übersteigt den Rahmen der hier vorgelegten Arbeit. Deshalb habe ich mein bestes versucht den Rahmen des Forschungsthemas einzuschränken und unwesentliche oder umstrittene Details auszuklammern. Folglich gliedert sich die vorgelegte Arbeit in vier Abschnitte.

Zunächst begrenze ich die Analyse von Stadtplanung auf zweidimensionale Grundrisse. Ich präsentiere hierzu das Quick Urban Analysis Kit (qua-kit) welches als Werkzeug für eine über das Internet betriebene Stadtplanungsübung gedient hat. Es hat mir ermöglicht die für die Studie notwendigen spezifischen Entwurfsdaten in einem maschinenlesbaren Format zu sammeln.

Im zweiten Abschnitt entwickle ich eine effiziente Symmetrieerkennungsmethode welche Kanten in einer Zeichnung analysiert um potentielle Symmetriemuster zu finden und deren Signifikanz zu messen. Dieser Teil der Arbeit dient als rechnerische Grundlage der Analyse.

Im darauf folgenden Kapitel führe ich eine Onlinebefragung durch um die Wahrnehmung von Ordnung in Entwurfszeichnungen zu bewerten. Die gewonnenen Resultate werden dann weiterverwendet um ein statistisches Modell über das Mass der wahrgenommenen Ordnung zu entwickeln.

Abschliessend entwickle ich ein Regressionsmodell welches die berechnete Symmetrie und die wahrgenommene Ordnung miteinander verbindet und beende die Arbeit mit der Analyse dieser Resultate. Ich zeige, dass das Rechenmodell mit der wahrgenommenen Ordnung im Datensatz der gesammelten Entwurfszeichnungen übereinstimmt. Ich schätze die Unsicherheit der Wahlmöglichkeiten im durchgeführten Experiment und bewerte die Genauigkeit der Vorhersagen. Schliesslich bringe ich den Beweis der die Struktur von Abhängigkeiten zwischen verschiedenen Typen von Symmetrie und ihrer wahrgenommenen Ordnung erklärt.

ACKNOWLEDGEMENTS

This work would not have been possible without the advice, continuous support, and encouragement of my colleagues, friends, and relatives.

First of all, I would like to express my warmest gratitude to Prof. Dr. Gerhard Schmitt for the exciting opportunity to be a part of our unique research group. I deeply appreciate his guidance, understanding, and the freedom he allowed me in pursuing my own ideas while helping to shape them into the present form.

I would like to thank Jun.-Prof. Dr. Reinhard König for his guidance in the first steps of my doctoral studies. He helped me to define the initial scope of my work and to develop research proposals.

I would like to thank Prof. Dr. Y. Liu for teaching the *Computational Regularity* course at ETH Zürich. The topic of this course evoked one of the core ideas of the thesis – using symmetry detection to analyze urban geometry.

I want to thank my colleagues at the chair of Information Architecture in Zürich and Future Cities Laboratory in Singapore for actively participating in development, testing, and promoting of qua-kit. Without them, this software would never be ready for public use. And I am grateful to the team of the "Future Cities" series of online courses. This enthralling series attracted thousands of students; some of them completed qua-kit design exercises, thus providing me with the data necessary for my research.

Above all, I would like to thank my family for their patience and support. I am deeply grateful to my parents, because they always believed in me. And I am deeply grateful to my wife for her love and support, as well as for the help with statistics and machine learning and for proofreading parts of this thesis.

I want to thank Swiss National Science Foundation for their financial support (Scientific & Technological Cooperation Program Switzerland-Russia 2015, Project IZLRZ1_164056).

Special thanks to Dr. Matthias Standfest for the German version of the abstract.

CONTENTS

INTRODUCTION

> *Symmetry, as wide or as narrow as you may define its meaning, is one idea by which man through the ages has tried to comprehend and create order, beauty, and perfection.*
> — Hermann Weyl (1952)

It is hard to overestimate the role of symmetry in science, art, and culture. On the one hand, people have been pursuing symmetry as a way to organize the world and our understanding of it. On the other hand, nature itself is full of symmetry stemming from the laws of physics.

Symmetry and order has fascinated humanity since ancient times. Aristotle's Metaphysics mentions that symmetry, order, and definiteness are the primary forms of beauty (Metaphysics: 1078a-1078b). Vitruvius in his *de architectura* claims that the symmetry, order, and arrangement are among the fundamental principles in architecture. Use of symmetry makes construction, design, and planning simpler, thus symmetric elements can be found everywhere in the history of architecture and urban planning. Even though some contemporary approaches in architecture promote the idea of breaking symmetry, I believe it still serves as a reference point in the creation process.

In science, symmetry is a synonym for invariance. A commonly known definition of symmetry is a property of an object (or its attribute) being invariant under a group of transformations. Symmetry is one of the fundamental principles in our understanding of nature. Munowitz (2006) notes, "for every symmetry, there comes a constraint". Indeed, symmetry constrains the complexity of various objects and phenomena at all levels of existence, from the wave function in quantum systems to biological organisms and to celestial bodies.

Often symmetry is associated with order. Both of these terms imply some structure in an observed object or phenomenon. The difference between the two notions becomes clear when one wants to find formal definitions for them. Independently of the domain of science, symmetry is always defined as a property of being invariant. Hence, it is always possible to test whether symmetry is present or not. For spatial symmetry of finite objects, it is also possible to measure the size of an object part being symmetric, which can serve as a starting point for measuring symmetry. On the contrary, the usage of the term *order* is very different across the domains of science. Usually, the idea of order is related to the degree of complexity or homogeneity of an object. Yet, in my opinion, none of the formal definitions are directly applicable to the orderliness as it is seen in architecture or urban design. At the same time, the orderliness as a subjective quality is rather intuitive; a person can easily decide if one group of objects looks more ordered than another. Therefore, analysis of perception might be the key to evaluating order.

In the presented work, I investigate the ways to evaluate symmetry and order in urban design. Although one can define the measure of symmetry as the measure of an object

part being symmetric (e. g. area or volume), it is not clear whether such a measure would agree with the individual's perception of symmetry. Another problem is the existence of imperfect symmetry: a small change of shape makes an object not symmetric in the strict mathematical sense, but a human observer may tolerate or ignore the imperfection. Thus, there must be a continuum of intermediate values in the symmetry measure indicating varying levels of symmetry significance. A few definitions for measuring imperfect symmetry have been proposed (e. g. Kutateladze, 1976; Zabrodsky, Peleg, and Avnir, 1992), but these works usually focus on the mathematical correctness rather than consistency with symmetry perception.

RESEARCH OBJECT The main objects of analysis in the presented work are symmetry patterns in design and human perception of these patterns. I assess the perception in terms of an individual's response to presence or absence of symmetry in an observed design. Independently of perception, I evaluate symmetry patterns by processing design geometry. The rich complexity of a real-world design makes it very hard to distinguish a response to a single aspect of the design from an overall impression. There are simply too many details that can affect the perception of symmetry or order. Therefore, I narrow down the scope of the research to the simplest possible form: two-dimensional figure-ground drawings of a small-scale urban design.

A figure-ground drawing is a simple and convenient form for conveying design ideas. It is a good representation for the symmetry analysis: it omits unnecessary details but preserves significant symmetry patterns. Furthermore, I restrict the drawings to polygonal geometry: no color, no other primitives such as isolated points and lines; curves are approximated by chains of line segments. This helps to rule out many undesirable effects on the perception and to simplify the symmetry analysis.

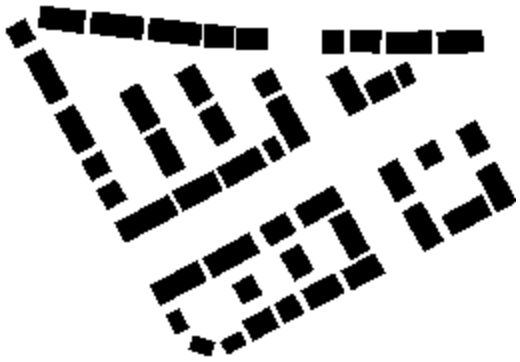


FIGURE 0.1: An example of a figure-ground drawing. In this work, it is often referred as a 2D polygonal design drawing to acknowledge its representation in the discussed algorithms.

It has been argued that the ability to notice symmetry is justified by the evolutionary necessity to process visual information more efficiently (Attneave, 1954). Experimental studies show that the human brain is particularly good at recognizing vertical reflection symmetry, probably due to its abundance in nature (Goldmeier, 1937; Rock and Leaman, 1963). Normally, such experiments are conducted in the form of surveys. A participant is asked either to tell if a single object is symmetric or to choose a more symmetric object from a set; typical objects are images of human faces, dots, or geometric shapes (Giannouli, 2013).

Psychological studies of symmetry perception are mainly interested in understanding neurological processes responsible for detection of symmetry. At the same time, computer vision studies attempt to quantify symmetry in a way that would conform human perception. For instance, Welch and Kobourov (2017) have shown in their experiment that a few previously proposed symmetry metrics on graphs tend to disagree with each other and often fail to predict the human assessment. In this work, I analyze symmetry perception

from the computer vision point of view, as an unknown function of an observed design drawing; and a human response in this model is taken as the ground-truth output value of the function.

MOTIVATION One research area that can benefit from symmetry analysis is generative design. It develops algorithms for exploring the space of possible design configurations. A generative algorithm does not need to operate on the same 2D representation as the symmetry analysis. Yet, even if one narrows down the problem to 2D space and allows only a limited set of shapes to be placed in the design space, there are endless possibilities for arranging them. For example, if the task is to arrange n rigid building blocks on a plane, the corresponding configuration space has at least $3n$ degrees of freedom (two position coordinates and the rotation angle for every block) and a complex set of non-linear constraints for allowed configurations (e. g. non-overlapping blocks). In this many-dimensional continuous space, only a small fraction of configurations corresponds to *meaningful* arrangements – such that would resemble a figure-ground drawing of a real-world design. The chance of hitting a meaningful configuration by randomly selecting a point in the configuration space is close to zero. Thus, a generative algorithm would need an extra guidance to search for viable configurations only.

Most modern studies in generative design define explicit rules to constrain the configuration space to correspond to meaningful designs only. For example, bottom-up approaches like shape grammars define sets of possible patterns (e. g. Gong et al., 2018); top-down approaches use spatial structures that cannot represent some (likely undesirable) configurations (e. g. Miao et al., 2018). These approaches can generate an optimal configuration or closely replicate a human-made solution; their output is often indistinguishable from a design emerged in the real world. However, every explicit constraint cuts the "creativity" of an algorithm: by forbidding a large range of (possibly undesirable) configurations, a *hard* constraint reduces the ability of an algorithm to produce unique, unpredictable designs.

Symmetry analysis could be a solution to the problem of hard constraints. A generative algorithm could use a measure of symmetry as a parameter of optimization and emerging symmetry patterns as *soft* constraints. Such a system of constraints would not forbid any design configuration explicitly but penalize it for deviating too much from a desirable state. That is, symmetry as a constraint would steer the exploration of the design space preserving the sense of order.

In the face of the flourishing artificial intelligence (AI) research, one could extend the idea of using symmetry as a constraint. Symmetry and order have been the basis of the architectural approach in past centuries. Even intentional asymmetry can be seen as deviation from symmetry. Hence, it is safe to assume that the same basis can be used by AI in future. In other words, a machine must understand symmetry to be able to generate designs. Thus, a symmetry detection study is a small step towards the AI-generated urban design.

Aside from optimization and machine learning applications, the presented study should help human designers understand the effects of symmetry better. Intuitively, the relation between symmetry and perceived order seems obvious: the more symmetry is present, the more ordered a design appears to a viewer. However, to the best of my knowledge, there is no practical way to evaluate this relation quantitatively. I believe, the quantification would

improve the understanding of a design. Even if the orderliness is not among desirable characteristics of a design, symmetry analysis still can be used as a reference point in a design process.

GOAL The goal of this work is to develop a computational approach for analysis of symmetry and order in urban design. This development consists of three components: the computational evaluation of symmetry, the model of order perception, and the analysis of the relation between the two. More formally, I aim to fulfill the following objectives:

1. develop an approach to detect symmetry patterns and measure symmetry in two-dimensional figure-ground drawings;

2. develop the measure of order in the drawings – a metric that would closely match the human perception of order;

3. investigate whether the detectable symmetry explains the perception of order.

The first objective is to be achieved on the grounds of symmetry detection research, a branch of computer vision.

The second objective implies a statistical experiment: I need to ask participants to evaluate their perception of order in design drawings; then I can build a model that predicts the perceived order score based on the previous participants' responses.

The last objective despite its closed form states an open-ended question. When the model is built, what can it tell about the dependence between the featured symmetry and the perceived order? How well does the model agree with the feedback of the experiment participants? Do different symmetry types play the same role in perception of order? Can the model be generalized beyond the designs in the conducted experiment?

The bottom line of the presented thesis is to prove a hypothesis: *computable symmetry reflects perception of order*. I do it by constructing a model of perceived order. Other questions stated in thesis are answered by means of observing the model performance.

OVERVIEW My approach to the problem is inspired by the streetscore model[1] developed by Naik et al. (2014). The streetscore algorithm evaluates the perceived streetscape safety based on images from Google Street View. Naik et al. created an online survey asking participants to assess a series of image pairs; for each pair, participants were asked to choose a neighborhood that looked safer. Then, Naik et al. used the Microsoft TrueSkill system to model neighborhood safety scores. At the same time, they used a comprehensive set of computer vision methods to extract some quantitative descriptors from the images. In the end, they used a machine learning algorithm to create a model of the safety score depending on the extracted descriptors. They trained the model with the output of the TrueSkill system and used it on other street views from the same cities.

Similarly to the work of Naik et al., the presented model consists of several components. However, while Naik et al. used the components of their model as they were provided by third parties, I have to adapt or extend every component of my model. As a result, the

1 `http://streetscore.media.mit.edu/`, last accessed on 25.09.2018

presented research covers several disciplines: I use computer vision methods to analyze symmetry in design drawings; then, I use statistical models in a psychological experiment to assess perception of order; finally, I use a range of machine learning techniques to investigate the relation between computable symmetry and perceived order. Therefore, I split the review of the state of the art methods into the respective chapters rather than keeping it in a dedicated chapter. I believe, this helps the reader to follow the logic of the thesis.

In the first place, I need to acquire data in an appropriate form for my study. Chapter 1 presents the work I have done for that. I introduce qua-kit – a software that I have developed to collect data and to let others share their experience in urban design. Qua-kit has been used as an exercise platform for a series of online courses. As a result I have obtained a unique data set of design submissions in a convenient machine-readable format. Most importantly, the submissions are made for a small group of well-defined exercise scenarios via the tool with a strictly limited functionality; this ensures that the designs are comparable, and many unknown factors that can affect the analysis are ruled out.

Chapter 2 describes a computational method for analysis of symmetry in figure-ground design drawings. The core of this method is an algorithm, which evaluates all edges in a drawing and assigns a score to every point in the parametric space of symmetry transforms. I have found that the existing symmetry detection methods do not perform well enough given such a specific geometry data as the design drawings. The proposed algorithm is an adaptation of the method proposed by Loy and Eklundh (2006).

In Chapter 3, I evaluate the perceived order score of design submissions by conducting a paired comparison experiment, similarly to the streetscore experiment of Naik et al. (2014). In this experiment, participants are asked to compare submissions and tell which one of them features more order. I adapt the Bradley-Terry model (Bradley and Terry, 1952) to rate submissions based on the experiment results. Thus, with the help of the experiment participants I create a crowdsourced order measure.

Chapter 4 brings the findings of Chapters 2 and 3 together. I build a regression model using the computed symmetry score as the input variable and the crowdsourced order measure as the output variable. After the model is trained with the experiment data, it can predict the perception of order in a design submission using solely its geometry. In the end, I discuss the conclusions drawn from the analysis of the developed model and the acquired data. That is, Chapter 4 describes the final component of the presented work and answers the questions stated in the thesis.

1

QUA-KIT AND DATA COLLECTION

In this chapter, I give an overview of Quick Urban Analysis Kit (qua-kit) – a system I developed to collect the data for my research and to test the concepts of Citizen Design Science. I enumerate the software and tools used for the development of the system and describe the instruments it provides. Then, I present the exercise grading system I developed for qua-kit. Next, I describe the qua-kit subsystem allowing to perform the analysis of an urban design using third-party software as computational services. Lastly, I list the case studies conducted by means of qua-kit and the collected data and discuss possible use cases for the qua-kit and the collected data.
The software and the data presented in this chapter serve as a basis and a frame for the research presented in the following chapters.

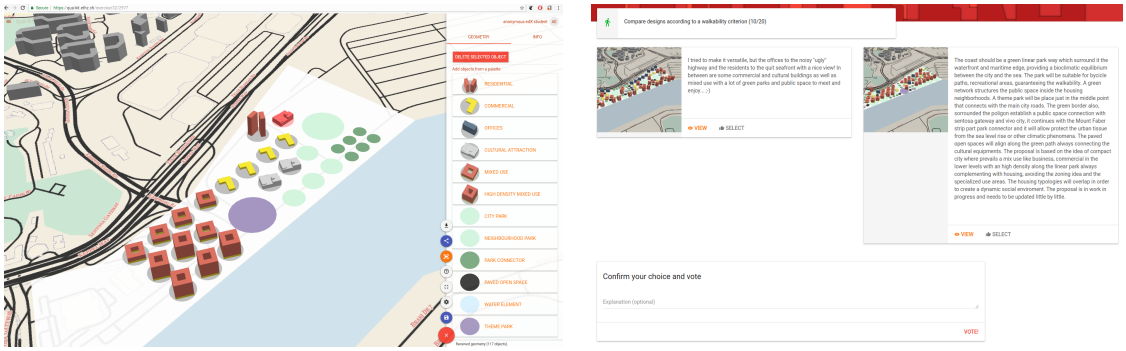
The term *order* has many definitions and is highly subjective when explained in terms of perception. An individual may understand and perceive order differently dependent on a context. And order may be perceived differently for different types of objects. Therefore, I narrow the scope of the analysis to a single aspect of an urban design. I have explored the ways to obtain a set of designs that have had as many similar or equal properties as possible, while varying significantly with respect to the human perception of order. In addition, having a sample of designs with similar semantic structure would have been beneficial for the quantitative analysis. To obtain a sufficiently large data set satisfying all these requirements, I have developed qua-kit – an online tool that is described in this chapter.

1.1 OVERVIEW

Quick Urban Analysis Kit (qua-kit) is a web platform for simple urban design editing, sharing and discussion[2]. The first version of the platform was released in summer 2016; since then, it received a few updates and yet has a list of features to be implemented. Qua-kit is integrated into a series of massive open online courses (MOOCs) called *Future Cities*, which is developed by the chair of Information Architecture (iA) at ETH Zürich. Qua-kit is used there in the form of two exercises.

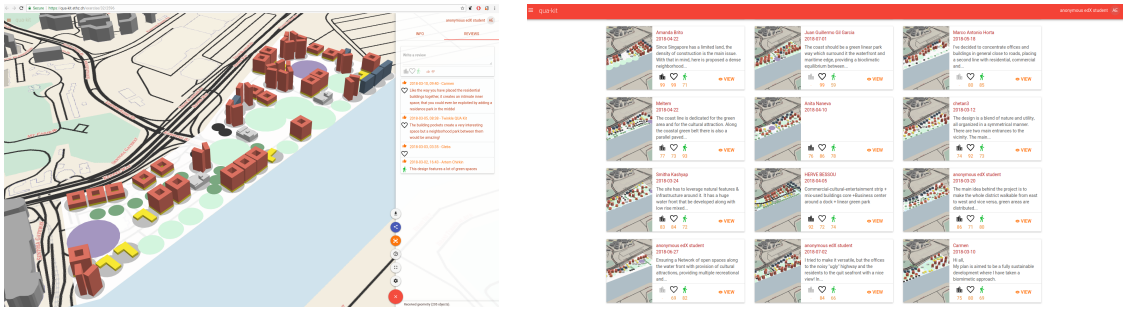
DESIGN EXERCISE In the first qua-kit exercise a student is asked to work on a predefined design scenario. Figure 1.1a presents qua-kit user interface for this exercise. The tool uses WebGL to manipulate 3D geometry in browser. The student can move, delete, or create (from templates) individual objects. After the work is finished, the student submits

2 `https://qua-kit.ethz.ch`, last accessed on 30.09.2018



(a) Design exercise (b) Compare exercise

FIGURE 1.1: The qua-kit web interface for student exercises.



(a) Submission view (b) Qua-kit gallery

FIGURE 1.2: The qua-kit web interface for viewing submissions.

the design with optional textual explanation of their ideas. At any moment, the student can come back to the site and update their submission.

COMPARE EXERCISE In the second exercise, given a series of twenty randomly selected submission pairs, a student is asked to select (vote for) one design in a pair that performs better according to a given design criterion. Figure 1.1b shows the interface for comparing two designs in a pair. At the top of the page, the student sees the name and and the icon of a design criterion under consideration. The student sees previews and descriptions of two randomly chosen designs. Then, the student has to select a design by clicking on a corresponding "Select" button. Optionally, a student can add a textual explanation for their choice.

To make the exercises more entertaining and to foster the discussion among students, qua-kit features a design gallery (Figure 1.2b). Students can browse the gallery to explore submissions of their peers. They also can open any submission and write reviews (Figure 1.2a, a panel on the right). Qua-kit has a ranking system that uses the votes from the *compare* exercise to calculate ratings of student designs, adding a competition element to the exercises.

Qua-kit keeps records of all student actions throughout their work on the exercises: it stores commentaries and geometry of submitted designs, as well as individual modifications (e. g. move a block from one place to another), votes, and other activity information. As a result, qua-kit database contains the full picture of students designs and interactions.

Most importantly, it features hundreds of unique submissions for every exercise scenario, which makes the statistical analysis possible.

Aimed initially to aid the research, qua-kit has developed beyond a data collection tool. Qua-kit implements the principles of Citizen Design Science – a union of citizen science, citizen design, and design science (Mueller et al., 2018). It supports a dual process of sharing knowledge in urban design and communicating citizen ideas to professionals and scientists. The two exercises establish a feedback loop. On the one hand, they enable students to develop their designs individually or as a community. On the other hand, qua-kit allows moderators to steer the community with expert knowledge.

1.2 DEVELOPMENT

Qua-kit is implemented using Haskell programming language. The site consists of two loosely dependent projects: qua-server and qua-view.

QUA-SERVER Qua-server is a program behind the site; it is based on yesod web framework[3] and PostgreSQL[4] database. Qua-server is responsible for managing the user data, presenting the site pages online and running most of the business logic. The user interface of qua-kit is based on a Daemonite's Material UI[5].

QUA-VIEW Qua-view is a client-side WebGL program. Qua-view is written in Haskell and compiled to JavaScript – a language that is executed by web browsers. The compiled qua-view code is transfered to a client by qua-server to start the design exercise. The logic of the program is implemented using Reflex[6] – a functional reactive programming (FRP) framework. Qua-view features an interactive 3D geometry editor; 3D geometry rendering is implemented using WebGL, which is a technology that allows using capabilities of a graphics processing units (GPU) to accelerate animation in a browser.

Qua-view handles the geometry in the augmented GeoJSON[7] format. GeoJSON is a widely-used format for encoding a variety of geographic data structures. GeoJSON represent every object as a *feature* that has a set of properties and a geometry. Qua-view explicitly defines special properties to control visual appearance and behavior of individual objects. In addition, it restricts the form of special objects, such as a scene view position (camera), template objects, etc. While the qua-view variant expands the GeoJSON format, it does not introduce any new entities; thus, third-party tools can open and manipulate it as usual. The full list of special properties and objects can be found on the project's web page[8].

3 `https://www.yesodweb.com/`, last accessed on 15.08.2018
4 `https://www.postgresql.org/`, last accessed on 15.08.2018
5 `https://daemonite.github.io/material/`, last accessed on 15.08.2018
6 `https://reflex-frp.org/`, last accessed on 15.08.2018
7 `https://tools.ietf.org/html/rfc7946`, last accessed on 22.10.2018
8 `https://github.com/achirkin/qua-view`, last accessed on 22.10.2018

1.3 EXERCISES MANAGEMENT AND EDX INTEGRATION

The first step of creating a design exercise in qua-kit is tailoring a GeoJSON scenario file containing geometry and context information. One can use a 3D design tool such as Blender to create a file with geometry and modify special properties and objects using a text editor. The second step is creating an exercise record in qua-kit via an administrator interface; this involves uploading the scenario file and specifying an exercise description, a preview image, and exercise rules. An administrator chooses whether to allow editing object properties or adding new and deleting existing objects. Additionally, an administrator need to define a few design criteria and link them to the exercise. A design criterion is represented in qua-kit as a text description and an example image. An exercise should have at least one criterion to make the grading possible. Once an exercise is created in qua-kit, it can be linked with an external course platform or used standalone.

STANDALONE EXERCISE The exercise editor interface for administrators displays special invite and enroll URL links for available exercises. An administrator gives one of these links to a group of participants; a participant can register and start an exercise by opening the link. Optionally, participants can register with their emails to login and continue the work on an exercise later.

INTEGRATED EXERCISE EdX provides an API for integration of third-party educational components. The API is an implementation of the Learning Tools Interoperability (LTI) specification[9]. LTI defines secure routines for authorizing students and transferring grades.

- When a student clicks on a special link inside edX site, a signed authorization request is sent to an LTI provider (qua-kit). The LTI provider identifies the student and authorizes them to start an exercise.
- After a student submits an exercise solution, LTI provider assigns a grade. Later, this grade can be transfered back to the LTI consumer (edX).

Qua-kit keeps a list of courses, exercise units, and users obtained from edX via the LTI. To set up an exercise for edX, one needs to go through three steps as follows:

1. specify an LTI public key and an LTI secret in qua-kit;
2. use the specified credentials to make an LTI passport string and register the string in an edX course properties;
3. add an LTI component to an edX course unit and specify exercise parameters, such as qua-kit address, exercise ID and type (“design” or “compare”).

This process is explained in detail in the edX documentation[10].

9 `https://www.imsglobal.org/activity/learning-tools-interoperability`, last accessed on 22.10.2018
10 `http://edx.readthedocs.io/projects/edx-partner-course-staff/en/latest/exercises_tools/lti_component.html`, last accessed on 15.08.2018

1.4 GRADING EXERCISE SUBMISSIONS

Having an experiment online potentially results in far greater number of submissions compared to an offline experiment. This is great for the analysis but raises another problem in the exercise setting: student submissions must be graded. Generally, there are two approaches to grade larger volumes of submissions: either calculate grades automatically or design a peer-review process. The former seems to be impossible in the context of Urban Design, hence I have implemented the latter.

The design exercise requires students to design their submissions keeping in mind a predefined set of design criteria. The *compare* exercise asks students to vote for designs according to these criteria. Thus, the votes submitted in the *compare* exercise can be used for grading the *design* exercise. The remaining part of the problem is grading the *compare* exercise. To solve this, I implemented a consensus model: the majority determines the standard. That is, a student gets a higher grade for the *compare* exercise if their votes agree with majority.

1.4.1 RATING MODEL

To grade submissions of the *design* exercise, the system needs to rate the designs based on the student votes obtained in the *compare* exercise. The task of rating the designs can be formulated as a problem of ranking given the paired comparison tests. One of the early paired comparison models was developed by Zermelo (1929); this model became widely known as the Bradley-Terry model presented by Bradley and Terry (1952). Later, Elo (1961) proposed a simple rating system for ranking the performance of chess players. The Elo rating system assumes the performance of a player is a normally distributed random variable and can be inferred from wins, loses, or draws. An important advantage of the Elo rating system is that players' scores can be updated over time without the need of recomputing the model using all previous games. Glickman (1998) developed the Glicko rating system as an improvement of the Elo rating system addressing a shortcoming of the latter: they added the "ratings deviation" (RD) measure to measure how reliable is the rating of a player based on how many games they played recently. A few more complex rating systems were developed recently due to rising demand in the area of online multiplayer games. Herbrich, Minka, and Graepel (2006) developed the TrueSkill rating system based on approximated Bayesian inference. The TrueSkill system allowed modeling team competitions as well as two-player games and featured a fair compromise between accuracy and evaluation cost to be practical for many online games. Some improvements to the TrueSkill system have been done in the last decade, for example, Minka, Cleven, and Zaykov (2018) and Nikolenko and Sirotkin (2010).

Grading the voters in the compare exercise can be viewed as a psychometrics problem of student performance assessment for a choice-based test. One of the most widely known models representing this problem is the Rasch model developed by Rasch (1960). The Item Response Theory (IRT) is a modern approach for design and analysis of experiments for testing individuals' skills or attitudes (Embretson and Reise, 2013).

Although, the problem of ranking items in the pairwise comparison tests and the problem of a person skill assessment are well studied, there is no public research on the combination

of the two to the best of my knowledge. Therefore, I have implemented a new model for qua-kit grading.

Define n – the total number of design submissions, including updates. n_i – the number of updates of a design by a single user; then $\sum_i n_i = n$. Define m as the total number of comparisons, and m_{i_t} as the number of comparisons involving design i_t, where i_t is the t-th update of a design i. Since every comparison involves two designs, the following is true: $\sum_i \sum_{t=1}^{n_i} m_{i_t} = 2m$. Finally, define the ratings:

- $\pi^{\mathrm{d}}_{i_t}$ – the design rating of a submission i, update t;
- π^{c}_k – the compare rating of a student k.

The ratings are not identifiable in an absolute scale due to the design of the *compare* exercise: the exercise asks students to tell which design is better rather than how good is a single design. Thus, I put additional constraints on the distribution of the scores:

$$\begin{aligned} \mathrm{E}[\pi^{\mathrm{d}}] &= 0, \qquad \mathrm{Var}[\pi^{\mathrm{d}}] &&= 1, \\ \mathrm{E}[\pi^{\mathrm{c}}] &= 0, \qquad \mathrm{Var}[\pi^{\mathrm{c}}] &&= 1. \end{aligned} \tag{1.1}$$

RATING EVIDENCE The rating evidence quantifies the information available to the model for assigning a rating to a design or a voter. Every time a design submission participates in a comparison test, its evidence value increases. But, when a user updates their submission, the evidence drops: a submission has changed and previous votes may not represent the current quality of the updated submission. The evidence value is similar to the inverse of the ratings deviation (RD) values introduced by Glickman (1998). In qua-kit, the evidence is updated on every design submission and on every vote as follows:

$$\begin{aligned} w^{\mathrm{d}}_{i_t} &= (1-\alpha) w^{\mathrm{d}}_{i_{t-1}} + \alpha m_{i_t}, \qquad w^{\mathrm{d}}_{i_1} = 1, \\ w^{\mathrm{c}}_k &= \sum_{\substack{\text{All pairs } (i_t, j_s) \\ \text{compared by user } k}} (w^{\mathrm{d}}_{i_t} + w^{\mathrm{d}}_{j_s}). \end{aligned} \tag{1.2}$$

Here $w^{\mathrm{d}}_{i_t}$ is the evidence value of design submission i, update t; w^{c}_k is the compare evidence value of a student k; $\alpha \in (0, 1]$ is a constant model parameter that defines how much previous submissions of the same author affect the current rating. For example, $\alpha = 1$ means the new submission is not related to the previous at all; $\alpha = 0$ means the design rating does not change on submission updates.

UPDATE ON SUBMISSION The rating of a submission is modeled as follows:

$$\pi^{\mathrm{d}}_{i_t} = \sqrt{1-\alpha}\,\pi^{\mathrm{d}}_{i_{t-1}} + \sqrt{\alpha}\,\xi, \qquad \xi \sim \mathcal{N}(0, 1). \tag{1.3}$$

Here ξ is a standard normal random variable. When a user updates their design submission, it may change significantly, invalidating previous comparison votes. Thus, ξ represents the lack of knowledge about the rating of the new version of the design. The system does not change the design rating at the moment the design is updated, but it changes the evidence ($w^{\mathrm{d}}_{i_{t+1}} \leftarrow (1-\alpha) w^{\mathrm{d}}_{i_t}$) and resets the comparison counter ($m_{i_{t+1}} \leftarrow 0$).

UPDATE ON VOTING The action of the pairwise comparison affects three parties: the voter and the two designs being compared. Given the designs i_t and j_s, define the decision of the voter as follows:

$$\delta_{i_t j_s k} = \begin{cases} 1 & \text{if } i_t \text{ is preferred over } j_s, \\ -1 & \text{otherwise.} \end{cases} \tag{1.4}$$

Additionally, define latent variables:

$$\begin{aligned} y_{i_t j_s k} &= \delta_{i_t j_s k}(\pi^{\mathrm{d}}_{i_t} - \pi^{\mathrm{d}}_{j_s}), \\ z_{i_t j_s k} &= \delta_{i_t j_s k} \mathrm{e}^{\pi^{\mathrm{c}}_k - y_{i_t j_s k}}. \end{aligned} \tag{1.5}$$

Variable $z_{i_t j_s k}$ is used for convenience in the following equations. Variable $y_{i_t j_s k}$ reflects the agreement of the voter's decisions with the majority: if the difference of design ratings $(\pi^{\mathrm{d}}_{i_t} - \pi^{\mathrm{d}}_{j_s})$ has the same sign as the decision $\delta_{i_t j_s k}$ then the user agrees with the current design ratings (opinion of the majority). The model can adjust the ratings but cannot change the user decisions $\delta_{i_t j_s k}$. Hence, the expected value of $y_{i_t j_s k}$ may serve as the quality of the model; the higher Ey is, the better the scores agree with the votes.

To maintain the ratings up to date with students votes and to avoid recalculating all ratings at once, the system updates the ratings and counters after every comparison trial. Let $\delta_{i_t j_s k}$ be a new comparison outcome; then, update the counters:

$$\begin{aligned} m_{i_t} &\leftarrow m_{i_t} + 1, & m_{j_s} &\leftarrow m_{j_s} + 1, \\ w^{\mathrm{d}}_{i_t} &\leftarrow w^{\mathrm{d}}_{i_t} + k, & w^{\mathrm{d}}_{j_s} &\leftarrow w^{\mathrm{d}}_{j_s} + k, \\ w^{\mathrm{c}}_k &\leftarrow w^{\mathrm{c}}_k + w^{\mathrm{d}}_{i_t} + w^{\mathrm{d}}_{j_s}. \end{aligned} \tag{1.6}$$

At the same time, update the ratings:

$$\begin{aligned} \pi^{\mathrm{c}}_k &\leftarrow \frac{\sqrt{w^{\mathrm{c}}_k}\pi^{\mathrm{c}}_k + \frac{y_{i_t j_s k}}{\sigma_y(\sqrt{\rho^2_{vy} w^{\mathrm{c}}_k + w^{\mathrm{d}}_{i_t} + w^{\mathrm{d}}_{j_s}} + \rho_{vy}\sqrt{w^{\mathrm{c}}_k})}}{\sqrt{w^{\mathrm{c}}_k + w^{\mathrm{d}}_{i_t} + w^{\mathrm{d}}_{j_s}}}, \\ \pi^{\mathrm{d}}_{i_t} &\leftarrow \sqrt{\frac{w^{\mathrm{d}}_{i_t}}{w^{\mathrm{d}}_{i_t}+1}}\pi^{\mathrm{d}}_{i_t} + \frac{z_{i_t j_s k}}{\sigma_z\sqrt{w^{\mathrm{d}}_{i_t}+1}\left(\sqrt{w^{\mathrm{d}}_{i_t}\rho^2_{zr_+}+1} + \sqrt{w^{\mathrm{d}}_{i_t}}\rho_{zr_+}\right)}, \\ \pi^{\mathrm{d}}_{j_s} &\leftarrow \sqrt{\frac{w^{\mathrm{d}}_{j_s}}{w^{\mathrm{d}}_{j_s}+1}}\pi^{\mathrm{d}}_{j_s} - \frac{z_{i_t j_s k}}{\sigma_z\sqrt{w^{\mathrm{d}}_{j_s}+1}\left(\sqrt{w^{\mathrm{d}}_{j_s}\rho^2_{zr_+}+1} + \sqrt{w^{\mathrm{d}}_{j_s}}\rho_{zr_+}\right)}, \\ &\text{where} \qquad \rho_{zr_+} = \frac{\rho_{z\max(\pi^{\mathrm{d}}_{i_t},\pi^{\mathrm{d}}_{j_s})} - \rho_{z\min(\pi^{\mathrm{d}}_{i_t},\pi^{\mathrm{d}}_{j_s})}}{2}. \end{aligned} \tag{1.7}$$

Here, σ_y and σ_z are standard deviations of variables y and z respectively; ρ_{vy}, $\rho_{z\max(\pi^{\mathrm{d}}_{i_t},\pi^{\mathrm{d}}_{j_s})}$, and $\rho_{z\min(\pi^{\mathrm{d}}_{i_t},\pi^{\mathrm{d}}_{j_s})}$ denote correlations of the random variables. These four values to some extent characterize the case study and the rating criteria. I have estimated them empirically using the data from the first exercise by minimizing the model error rate. Together with model parameters, these represent the set of constants for the qua-kit rating model:

$$k \leftarrow 0.7, \qquad \sigma_y \leftarrow \sqrt{0.46}, \qquad \sigma_z \leftarrow \sqrt{0.2}, \qquad \rho_{vy} \leftarrow 0.33, \qquad \rho_{zr_+} \leftarrow 0.45. \tag{1.8}$$

Note, since both designs in a comparison are chosen at random from the same distribution, and the distribution has the zero mean, distributions of $\max(\pi^{\mathrm{d}}_{i_t}, \pi^{\mathrm{d}}_{j_s})$ and $\min(\pi^{\mathrm{d}}_{i_t}, \pi^{\mathrm{d}}_{j_s})$ must be equal. This results in the following property:

$$\rho_{z\,\max(\pi^{\mathrm{d}}_{i_t}, \pi^{\mathrm{d}}_{j_s})} = -\rho_{z\,\min(\pi^{\mathrm{d}}_{i_t}, \pi^{\mathrm{d}}_{j_s})} = \rho_{zr_+}. \tag{1.9}$$

Exploiting this property, I define parameter ρ_{zr_+} in Equation 1.7 to make the numeric estimation more robust.

INITIAL ASSUMPTIONS AND FAIRNESS When a user submits their design for the first time, their rating is set to zero ($\pi^{\mathrm{d}}_{i_1} = 0$) and evidence value is set to the minimum possible value ($w^{\mathrm{d}}_{i_1} = 1$). A user that has not voted yet has the zero voter rating ($\pi^{\mathrm{c}}_k = 0$) and the minimum possible evidence value ($w^{\mathrm{c}}_k = 1$). As seen from Equations 1.2 and 1.6, the more active a student is, the more rating evidence the system has. At the same time, the compare rating changes depending on how well the student votes agree with the opinion of the majority. This model has a caveat: the continuous nature of the experiment and the lack of votes at the beginning of the experiment suggest that ratings may be biased towards the opinions of the first voters. This supposition requires further investigation and lies outside of the scope of the presented work.

1.4.2 RATINGS AND QUA-KIT

Rating values as described in this section have zero mean and unit variance. However, qua-kit presents the ratings on the integer scale from 0 to 99, and edX accepts the grades on the scale from 0 to 1. Thus, the system uses a custom transforms to present ratings in qua-kit and grades in edX:

$$\begin{aligned} f_{\mathrm{qua-kit}}(\pi^{\mathrm{d}}) &= \max\left(0, \min\left(99, 50 + \frac{100}{2.1}\pi^{\mathrm{d}}\right)\right) \\ f_{\mathrm{edX:design}}(\pi^{\mathrm{d}}) &= \max\left(0.6, \min\left(1, 0.8 + 0.2\pi^{\mathrm{d}}\right)\right) \\ f_{\mathrm{edX:compare}}(\pi^{\mathrm{c}}) &= \max\left(0.6, \min\left(1, 0.6 + 0.4\pi^{\mathrm{c}}\right)\right) \end{aligned} \tag{1.10}$$

The constants for the transformations are chosen empirically.

PRESENTING RATINGS Evaluating the rating of a submission takes time: at the moment of design submission, there are no votes to use for the estimation; then, students start to vote and slowly develop the evidence. When the evidence level for the design rating reaches a predefined minimum value, qua-kit makes the rating visible. The minimum evidence value currently defined in qua-kit corresponds to having at least three votes. Figure 1.3 shows the preview card of a submission in the qua-kit gallery. The example submission has enough evidence to show ratings for two out of four available criteria.

GRADING Qua-kit cannot evaluate the grade for a design at the moment it is submitted. Thus, our teaching team have decided to assign a minimum acceptable grade automatically upon submission and update the grade when qua-kit gets enough votes. The qua-kit subsystem for communicating edX grades acts as follows:

- when a user finishes the exercise, the minimum grade of 0.6 is saved and sent to edX;

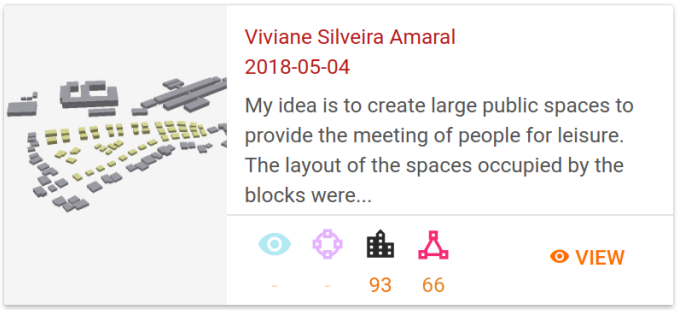


FIGURE 1.3: Preview of a student submission in qua-kit. As soon as the submission score evidence for a criterion reaches a predefined minimum value, the score appears in the qua-kit gallery.

- once a day, the grade is recalculated and resent if the design has received new votes.

The final grade is computed as the average of available ratings and transformed as specified in Equation 1.10.

MODEL PERFORMANCE The most straightforward way to assess the correctness and the performance of the presented rating model is to calculate its error rate on the real exercise data. I define the error rate as the fraction of user votes that disagree with the ratings of designs being compared:

$$\mathrm{Err}(\pi^{\mathrm{d}}) = \frac{1}{m} \sum_{\delta_{i_t j_s k}} \mathbb{1}(y_{i_t j_s k} < 0) = \frac{1}{m} \sum_{\delta_{i_t j_s k}} \mathbb{1}(\delta_{i_t j_s k}(\pi^{\mathrm{d}}_{i_t} - \pi^{\mathrm{d}}_{j_s}) < 0). \tag{1.11}$$

In all of the qua-kit exercises and criteria, the error rate $\mathrm{Err}(\pi^{\mathrm{d}})$ varies between 0.25 and 0.3. Is that a good model? This question is difficult to answer without an alternative to compare. On the one hand, assigning random scores to the designs would result in the error rate $\mathrm{Err}(\pi^{\mathrm{d}})$ being close to 0.5, which is only twice as large as the error rate of the presented model. On the other hand, the error rate being close to zero is not possible due to the way qua-kit chooses the design pairs: chosen at random, the submissions may be very similar or not comparable.

Another interesting approach for analyzing the performance of the model is the histogram of ratings. Figure 1.4 presents the histograms of "winning" (red) and "losing" (blue) submissions. Every design in these histograms is counted the number of times it was (or wasn't) chosen by a voter. Although the histograms largely overlap, their peaks are clearly separated suggesting that the system has some prediction power.

1.5 COMPUTATIONAL DESIGN ANALYSIS

Besides geometry editing, qua-kit is meant to be a simple interface for analysis tools. In contrast to arbitrary design criteria used in the *compare* exercise (Section 1.4), such analysis tools may compute a very limited set of essential design properties. In other words, they provide various quantitative metrics to assist the design process.

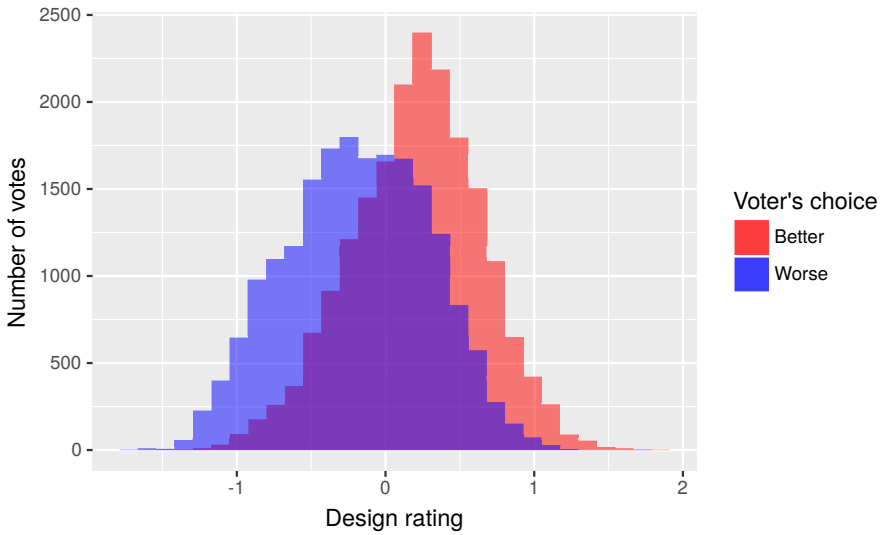


FIGURE 1.4: Rating histograms of all designs appeared in the compare exercise. Although the system selects both designs independently at random, users introduce the bias between the sample of "winning" (red) and "losing" (blue) submissions.

Figure 1.5 presents the components of a system behind qua-kit that provides analysis tools to its users. Depending on an execution mode, qua-view allows clients to start analysis services remotely. The start request is forwarded by qua-server to a middleware component, which, in turn, executes the analysis request and returns the results back.

1.5.1 LUCI MIDDLEWARE

Lightweight Urban Computation Interchange (LUCI) is a specialized middleware developed by Treyer et al. (2016). The core of LUCI is a message transfer protocol and a task scheduler.

A LUCI message contains a JSON header and a series of binary attachments. The header specifies attachments descriptions and task execution parameters such as a service name or a requested form of a result. The attachments contain data to be passed to analysis services. The full description of the message protocol is available online on the qua-kit project page[11].

The task scheduler is a TCP/IP server that implements all its functionality via services as shown in Figure 1.5. Every client connected to the scheduler may register any number of services it provides and several clients may register the same service. The scheduler keeps a pool of registered services and associates every service record with a list of clients providing the service. The scheduler balances clients' requests to a service among the clients that provide it. That is, every client can register and provide or request any services.

In the scheme presented in Figure 1.5, qua-kit is a client of the LUCI server. Qua-view executed by a client browser connects to qua-server using the WebSockets protocol; qua-

11 `https://github.com/achirkin/qua-kit/raw/reflex/doc/bin/specification.pdf`, last accessed on 16.08.2018

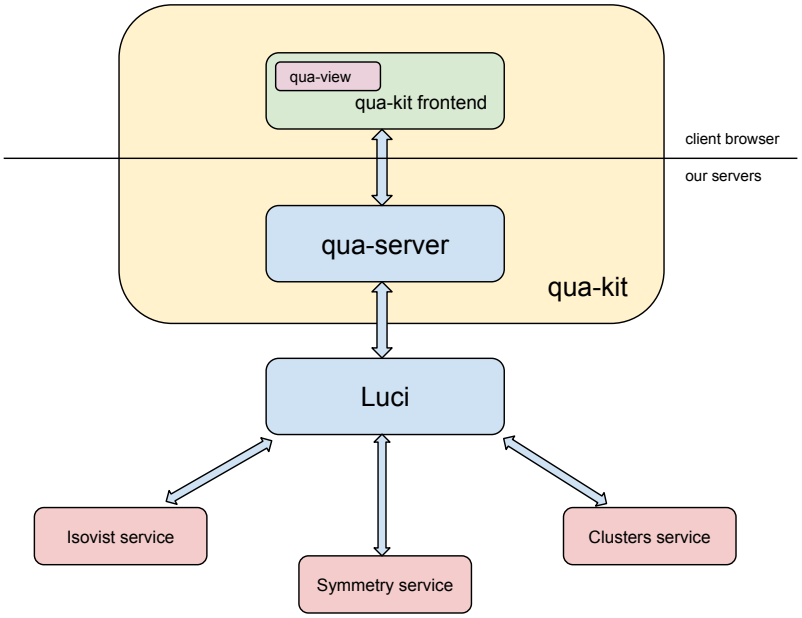


FIGURE 1.5: Components of the qua-kit-luci system. Luci is a middleware that provides access to analysis services; Qua-server plays the role of a proxy between clients (qua-view in a browser) and Luci.

server authorizes the client and forwards the content of WebSockets messages to LUCI via TCP/IP.

1.5.2 SCENARIO MANAGEMENT

One of the key services in LUCI is the scenario management service, which introduces the concept of the *scenario*. The scenario is a collection of geometry, geographical context, and other information stored on a server. The implementation of the qua-kit scenario services is called siren[12]. Siren uses PostgreSQL database with PostGIS extension to store scenarios and all scenario updates in a GIS-compatible format. Siren provides a set of services to list, get, create, and update a scenario or a part of it.

A scenario is associated with a unique identifier. Using this identifier, a client can refer to a scenario to synchronize all changes with the siren database or to run analysis services on it. A service obtains a scenario from siren using an identifier specified in a service run request. This scheme avoids sending the scenario back and forth between a client and a service and enables a collaborative mode of work on a scenario.

1.5.3 ANALYSIS SERVICES

The qua-kit analysis service is a LUCI service that can be invoked using the qua-view graphical interface. This implies a few additional requirements on the contents of the service request and response. The protocol specification on the qua-kit project page[11] defines the form of parameters and execution modes. Qua-view allows a user to specify

12 `https://github.com/achirkin/qua-kit/tree/reflex/services/siren`, last accessed on 16.08.2018

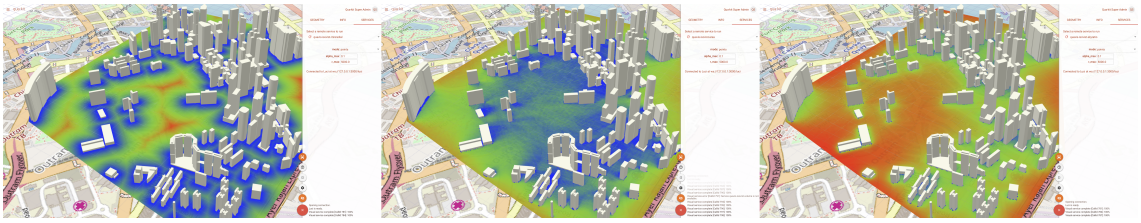


FIGURE 1.6: Visibility-based analysis services executed via qua-kit; qua-view user interface.

parameters using GUI control elements using number, boolean, enumeration, or string inputs.

All qua-kit analysis services require the scenario identifier as a parameter. The scenario identifier gives information about the geometry and the context, but more parameters are required depending on an execution mode. Qua-kit supports four execution modes:

1. `mode:points` means the client (qua-view) sends a grid of points and receives an array of values; qua-view visualizes the values as a heat map;

2. `mode:objects` means the service sends a single value for every object in a scenario; qua-view visualizes the values by coloring the objects (buildings and blocks);

3. `mode:scenario` means the service returns a single value, either `.png` image or a string; qua-view shows the result on a dedicated placeholder;

4. `mode:new` means the service is allowed to create new or update existing geometry; qua-view notifies a user when the service finishes its execution while keeping track of scenario updates.

The most common mode of execution is `points`. The result of execution in this mode appears to a user as a heat map beneath the scenario geometry. As an example, Chirkin, Pishniy, and Sender (2018) developed a series of services evaluating various functions of the space visible from each point on a grid. Figure 1.6 presents screenshots of three different services running on the same scenario.

If the heat map is not sufficient instrument to represent results of the analysis, a service can return an arbitrary picture using the `scenario` mode. Sender et al. (2018) used this approach to show an optimal arrangement of street lights and corresponding light levels. Alternatively, one can use the `new` mode to view results in the geometric form, as it has been proposed by Mouromtsev et al. (2017).

1.6 CASE STUDIES

The chair of Information Architecture (iA) at ETH Zurich created a series of four massive open online courses (MOOCs) and published it on edX platform[13]. Since the release of the

13 `https://www.edx.org/xseries/future-cities-0`, last accessed on 20.08.2018

first version of qua-kit, more than 1500 users world-wide created a design using the website at least once. Qua-kit has been used in two online courses and in several workshops:

1. Smart Cities – edX course `FC-03x-1`;
2. Responsive Cities – edX course `FC-04x-2`;
3. Smart Cities – edX course `FC-03x-2`;
4. Responsive Cities – edX course `FC-04x-1`;
5. Dashilar design exhibition week – public experiment (Beijing, China);
6. `ideasfortanjongpagar.com` – public workshop with three scenarios (Singapore).

1.6.1 QUA-KIT IN ONLINE COURSES

Qua-kit has been developed primarily to be used in conjugation with edX platform. Hence, edX is the main source of participants for the study. Our MOOC team has selected case studies emerged from the research of our partner teams.

EMPOWER SHACK Empower Shack[14] is an interdisciplinary research and development project conducted by the Urban Think-Tank (U-TT) team at ETH Zürich and their partners. The project aims at redeveloping informal settlements in suburbs of Cape Town, South Africa, where a housing crisis threatens millions of people. The U-TT team works with local communities to offer sustainable and affordable construction upgrades, fair distribution of public space, and delivery of public services. Some of their results and proposals are presented in "SLUM Lab 9: Made In Africa" (Klumpner and Brillembourg, 2014).

One of the long-term goals of the Empower Shack project is to develop a methodology and instruments that can be used by local communities with no involvement of foreign professionals. Miao et al. (2017) have made an effort to automatize the process of redevelopment. They have implemented a design generation tool using Rhino Grasshopper software and have connected it to qua-kit via LUCI.

14 `http://u-tt.com/project/empower-shack/`, last accessed on 20.08.2018

The Empower Shack project appeared as an exercise in the "Smart Cities" course at edX[15]. The course ran two times, and we chose two adjacent districts in Cape Town as the exercise scenarios. Both districts were rather small, approximately 0.5 hectare. We imported and adapted the context geometry, such as surrounding roads and building, from OpenStreetMap[16]. Then, we placed several dozens of primitive blocks at random positions in the center of the scenario area, the blocks represented small one-family shacks. The students had to redevelop the district considering a given list of design criteria. Figure 1.7 shows an example of a student submission. The dark blocks in the figure represent static buildings (environment) and the light-yellow blocks are movable objects. In this exercise, students could not add or delete blocks; they could only move or rotate them. The intended result of this simplicity was that the student behavior would be easy to analyze due to the small action space.

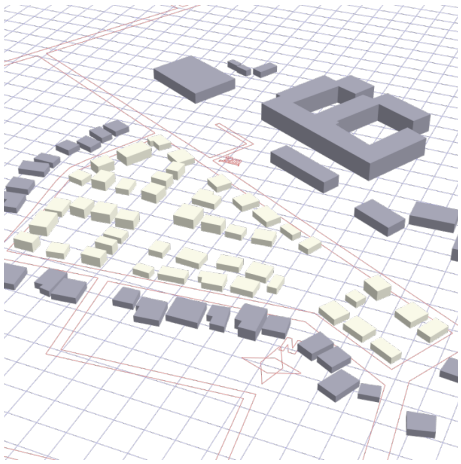


FIGURE 1.7: A submission for the Empower Shack exercise (Khayelitsha, Cape Town, South Africa).

TANJONG PAGAR Tanjong Pagar is a historic district in Singapore, which is planned to be upgraded by the Urban Redevelopment Authority (URA) of Singapore. In particular, the Tanjong Pagar container terminal is supposed to be moved away from the Tanjong Pagar district, which would free a huge seashore area for other uses. The district was in the focus of research at the Future Cities Laboratory (FCL), at the Singapore-ETH Center[17]. Thus, I suggested the FCL team to use it as a scenario for our fourth edx course "Responsive Cities"[18]. The FCL team kindly prepared two scenarios taking advantage of new features implemented in the new qua-kit version.

Figure 1.8 presents a student submission for the Tanjong Pagar exercise. Compared to the Empower Shack exercise, this scenario covers a larger area and operates in terms of large buildings instead of small blocks. By the time the exercise was designed, qua-kit had matured as a geometry editor. This allowed us to enrich the context of the exercise: we added a map layer underneath the buildings, colored buildings and building parts according to their function (e. g. residential or commercial), presented preview images and extra information about individual buildings, enabled creation (from templates) and deletion of objects.

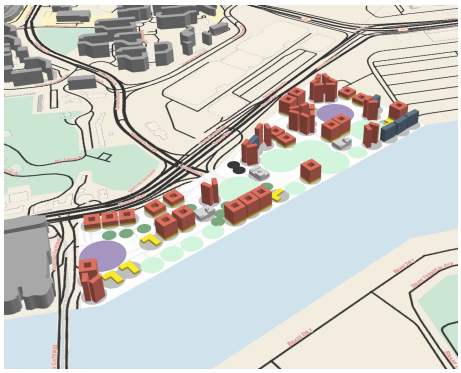


FIGURE 1.8: A submission for the Tanjong Pagar exercise (Singapore).

15 `https://www.edx.org/course/smart-cities-ethx-ethx-fc-03x-1`, last accessed on 20.08.2018
16 `https://www.openstreetmap.org`, last accessed on 20.08.2018
17 `http://www.fcl.ethz.ch/`, last accessed on 23.10.2018
18 `https://www.edx.org/course/responsive-cities-1`, last accessed on 23.10.2018

The increased complexity of the scenario geometry, the rich context, and the new types of allowed actions had twofold consequences for the exercise results. On the one hand, the produced submission data were harder to process and analyze. On the other hand, the students had better understanding of the model and were able to produce more diverse results.

1.6.2 QUA-KIT WORKSHOPS

Up to the date of writing this thesis, qua-kit has been used in a few private workshops and in two public experiments. In these studies, qua-kit has been hosted on external servers; thus, the corresponding collected data summary is not included in Section 1.7.

The first public case study was conducted during the ten-day Beijing Design Week starting on September 25, 2017. The description and the result analysis were presented on CAADRIA'2018 conference by Lu et al. (2018).

The FCL team in Singapore used qua-kit in another study called "Ideas for Tanjong Pagar"[19]. The experimental side of the study consisted of three design exercises on different scales in Tanjong Pagar container terminal area of Singapore. Theoretical principles of the study were published in an article "Citizen Design Science: A strategy for crowd-creative urban design" (Mueller et al., 2018).

1.7 DATA COLLECTION RESULTS

The live version of qua-kit has been hosted on ETH servers at `qua-kit.ethz.ch` for two years. In this time, the iA team used it for the MOOC exercises as well as closed tests and workshops. A part of the data related to the non-public use of qua-kit was deleted. A few student submissions that were clearly broken due to technical faults were also deleted from the database. All these manual changes to the database introduce small inconsistencies to the collected data. Thus, the statistics presented in this section is not exact. Though, the total discrepancy must be less than a few percent.

At the moment of writing this thesis, the qua-kit database counts five exercises: two runs of "Smart Cities" course, two runs of "Responsive Cities" course, and a private workshop conducted on 06.10.2017. The five exercises amount to 1558 unique submissions. In total, eight design criteria are linked to at least one exercise (one to four per exercise). Table 1.1 shows the statistics for every exercise. "Submission updates" column depicts how many times the students submitted their designs including updates. "Reviews" column reflects the number of voluntary comments made by students; it may be interpreted as a degree of engagement, because reviews are not a part of the exercises. "Compare votes" is the number of comparisons a student has done for an exercise. The minimum number of votes required to finish the compare exercise is 20, but a student is free to continue comparing after they finish the exercise.

19 `http://ideasfortanjongpagar.com`, last accessed on 20.08.2018

Exercise name	Criteria	Unique submissions	Submission updates	Reviews	Compare votes
Smart Cities [EdX FC-03x-1]	4	569	1296	855	23007
Smart Cities [EdX FC-03x-2]	4	686	1185	213	20049
Workshop 06 Oct 2017	1	11	11	6	0
Responsive Cities [EdX FC-04x-1]	4	183	362	131	7090
Responsive Cities [EdX FC-04x-2]	3	109	186	24	2704
Total	8	1558	3040	1229	52850

TABLE 1.1: Exercise statistics

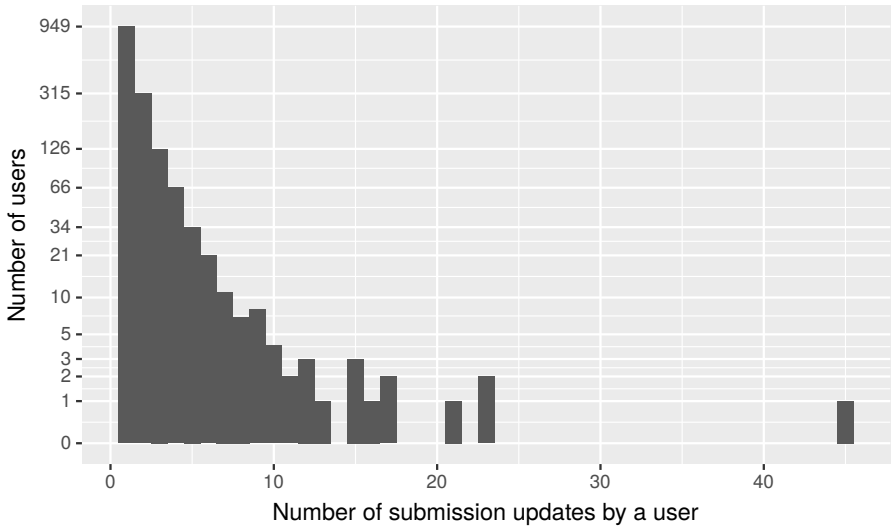


FIGURE 1.9: Number of submission updates done by qua-kit users (Note the logarithmic scale of the vertical axis). Most commonly, a student submits their design once, but a large fraction of students come back to update their submissions. One student submitted a design 45 times.

STUDENT ACTIVITY One of the goals of the qua-kit exercises is fostering public participation. Therefore, I have added some interaction elements to the site, such as a personalized view of comments and votes, to convince students to come back to the site and continue the work on their designs. Some indication of the outcome can be deduced from the "Reviews" and "Submission updates" columns of Table 1.1. Another way to assess the engagement is to test how many times students update their submissions. Figure 1.9 presents a histogram of submission updates. 949 out of 1558 students submitted their design only once; the rest 609 students submitted their designs at least twice. 294 students did more than two submissions, and one student submitted their work as many as 45 times.

QUA-VIEW ACTION LOGGING As soon as the first version of qua-kit was set up live for public use, the system started to record user actions in qua-view. The goal of this recording was to be able to reproduce the process of design development. This would allow me and other researches to study the behavior of students and designers when they used qua-view. Qua-kit logged actions of all users: students of the edX courses and workshops, as well

Question	yes / total
Did you look at other design submissions in the gallery?	40 / 43
Were the four design criteria useful for you?	39 / 41
Were the design criteria descriptions useful and understandable?	32 / 41
Did you search for an additional information about the case study?	27 / 42
Did you change your design after any feedback from other students?	7 / 36
Do you agree with your final design rating (compared to others)?	36 / 38
Do you feel the design rating is fair?	31 / 33

TABLE 1.2: A student survey after the first round of "Smart Cities". The closed-form questions and the fraction of positive answers.

as local and anonymous users running analysis services via LUCI. Hence, the number of action records in the database is slightly higher than it would be counted for the edX students only.

In the beginning, qua-view allowed only to move and rotate objects. Thus, I created a database table, the sole purpose of which was to store transformation matrices together with user identifiers and timestamps. The type of transformation matrices I used was a 4x4 matrix representing affine transformations in homogeneous coordinates. By March 2018 the logging table had 498897 records. This is the number of individual object motions and rotations done by all qua-kit users in the period from September 2016 to March 2018.

In late 2017 qua-kit received a major update that enabled new editor features. The original logging table was no longer able to represent all possible qua-view actions. Therefore, I added a new table to store arbitrary actions in a JSON-encoded format. The new format included more types of records: deleting and creating objects, updating object properties, updating camera (view point) position, and others. The site switched to the new version in March 2018; the new logging system was used since then. At the moment of writing these lines, there are 226208 records in the logging table, among which 60099 records are object location updates.

STUDENT SURVEY After the first run of the "Smart Cities" course had finished, I published a qua-kit survey to find out the ways to improve the site for the next rounds of exercises. The survey consisted of a few closed-form and open-form questions asking if a user had any technical problems, if they understood the design criteria, whether there was enough context information, and others.

The iA MOOC team sent a link to the survey to all students of the course using the edX mailing system. In total, around forty students responded; it was not possible to determine the exact number of students, because all questions were optional and the survey was anonymous. Table 1.2 presents some of the questions in the survey. The survey helped me to understand some flaws in the system and to choose the directions for further development. For instance, based on the survey feedback, we updated the design criteria list and descriptions for the next rounds of exercises. I got a crude estimate of students' opinions on the rating system and improved the graphical user interface. I have not attempted to run the survey one more time yet.

1.8 DISCUSSION

Qua-kit started as a simple prototype in 2016 and evolved into a fully-functional Citizen Design Science tool by 2018. It was used by the members of the iA and FCL teams for workshops and research as well as in the MOOCs. However, there are still more features we want to implement to improve the user experience and the editor capabilities for the creativity. The computational design analysis component of the system was not tested in public yet due to security and performance considerations. The main priority for further development would be to make the computational design analysis tools accessible for public users, because this would support the educational role of the system. The source code of qua-kit is published online under the MIT license[20]; thus, it can be used by other research and education groups in future.

In the scope of the presented research, the most important outcome of the qua-kit development and studies is the designs submitted by the students. The designs constitute a unique data set:

- Each exercise scenario has a sufficient amount of submissions to be analyzed statistically (see Table 1.1).
- The design geometry is represented in a widely known and easy-to-process GeoJSON format.
- The controlled set of allowed actions ensures preserving the design structure for an exercise. Thus, the geometry of scenario objects (buildings or blocks) may be abstracted away during the analysis phase if necessary.
- The voting data can be used to some extent for the automatic assessment of submissions. This adds more quantitative metrics to the geometry data.

The data set will be published online based on the ETH Zürich open access policy.

In the following chapters, I use the submission data of the first run of the "Smart Cities" course (edX FC-03x-1) if not stated otherwise. I have chosen this exercise, because it features more submissions and reviews and a simple geometry.

20 `https://github.com/achirkin/qua-kit`, last accessed on 18.08.2018

2

ANALYSIS OF SYMMETRY IN URBAN GEOMETRY

In this chapter, I present a novel approach for detecting symmetry patterns in two-dimensional polygon geometry that is provided by the qua-kit design exercises. Common symmetry detection algorithms rely on detecting and using outstanding points of an object – local features; they match pairs of local features to determine location and other parameters of a symmetry pattern. The proposed algorithm employs spatially extended edge features – walls of building blocks. Thus, it has a potential to provide much more complete information about symmetry patterns than point-based algorithms.
First, I give an overview of existing studies on symmetry detection, evaluation, and perception. Then, I describe the challenges of symmetry detection in qua-kit designs and advocate the choice of the method. Then, I explain the proposed method in detail, step-by-step: symmetry voting, search of best pattern parameters, grouping objects into patterns, and unification of similar patterns. Last, I evaluate the method performance on the data provided by qua-kit exercises and discuss the algorithm limitations and the directions for future research.

In this work, I regard an urban design as a two-dimensional geometry of its top-down view. This representation is indeed very limited, yet it is convenient for the presented analysis for two reasons. First, symmetry is much easier to analyze in two dimensions than in three dimensions. Second, given the low level of detail of qua-kit designs, symmetry patterns are essentially possible in the horizontal plane only.

I study symmetric patterns formed by the projections of buildings and other objects. In this context, symmetry is defined as a property of a region in a design, that the top-down projection of the region is invariant to a given two-dimensional transformation. Objects in the region form a symmetry pattern. Therefore, the symmetry analysis amounts to the problem of detecting symmetry patterns and the problem of measuring the *amount* or the *significance* of symmetry featured in a design.

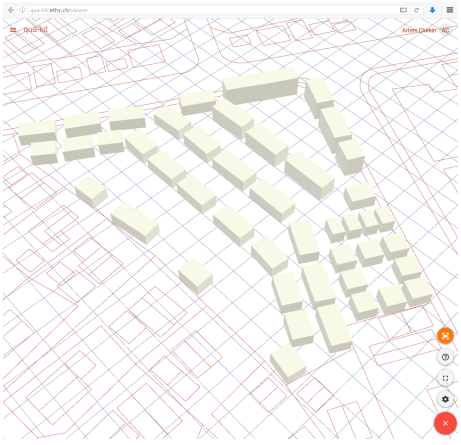


FIGURE 2.1: An example of a student design submission.

The design submission data described in Chapter 1 largely determines the approach I take to evaluate symmetry of a design. I have access to the design

Findings of this chapter are partially based on the content of my course project for the course 263-5903-00 *Computational Regularity* taught by Prof. Dr. Y. Liu at ETH Zürich in the Fall semester 2016.

geometry in a vector (polygonal) format and I have to cope with problems arising due to limitations and specificities of the environment used to produce the submission. The goal of the work presented in this chapter is to process raw polygonal geometry data of student submissions to augment it with quantitative information about structure and symmetry of the submitted geometry. Then, this information can be used for further design analysis, similarity tests, and computational design optimization.

2.1 STATE OF THE ART

Attneave (1954) argued that the evolution had tailored the biological vision to perceive highly redundant scenes in terms of information quantity. They justified that visual cues explained by the presence of symmetry and the Gestalt grouping laws allowed a brain to process the visual information more efficiently.

The evolutionary ability to notice symmetry patterns has been studied experimentally in psychology since mid-twentieth century. For example, Goldmeier (1937) tested if humans were better at detecting bilateral symmetry when the symmetry axis was vertical. Rock and Leaman (1963) and Corballis and Roldan (1975) argued that the perceived vertical axis was not necessarily forced by the gravitational coordinates and that the brain could compensate some misalignment with an eye retina as well. A comprehensive review of symmetry perception studies was published by Giannouli (2013).

For thousands of years, symmetry has been used for aesthetics reasons in architecture and arts. However, the effect of symmetry on a person's affection for a perceived object is difficult to study with scientific methods. Dresp-Langley (2016) presented a psychophysical experimental study asking participants to assess the perceived beauty of fractal trees with complete or broken bilateral symmetry. They claimed that a small break of symmetry in a presented pattern significantly reduced its perceived attractiveness. Referring to psychological and biological studies, they asserted that the subconscious affinity to symmetry in complex visual patterns had evolutionary roots and was inherent to humans irrespective of their professional background. Welch and Kobourov (2017) in their experiment compared several mathematically-defined symmetry measures on graphs to human perception of symmetry.

DETECTING SYMMETRY To the best of my knowledge, psychological studies agree that symmetry, being so easily identified by the human vision, plays an important role in object recognition. This phenomenon has attracted attention of researchers in the computer vision discipline (Chen et al., 2017; Davis, 1977; Loy and Eklundh, 2006; Marola, 1989; Yip, 2000). Symmetry detection has found its applications in computer vision attentional mechanisms (Reisfeld, Wolfson, and Yeshurun, 1995; Zhang and Lin, 2013), 3D object reconstruction (Cohen et al., 2012; Wu, Frahm, and Pollefeys, 2010), building facade detection and generalization (Musialski et al., 2013; Wang et al., 2015; Wu, Frahm, and Pollefeys, 2010), and other research areas. The two branches of symmetry detection research, two-dimensional image space symmetry detection and three-dimensional space symmetry detection, employ similar methods, but vary in areas of application and thus face slightly different challenges. Dealing with 3D meshes, volumes, or point clouds, the

latter has to cope with larger number of symmetry transform parameters; whereas the former focuses more on tolerance for partial occlusions, distortion, and noise in images. A comprehensive overview of symmetry analysis in images was published by Liu et al. (2010); symmetry detection methods in 3D geometry were discussed by Mitra et al. (2013b).

According to Liu et al. (2010) and Mitra et al. (2013b), symmetry detection methods can be divided into two groups: global and local methods. The global detection methods assume that a whole object is symmetric. This assumption greatly simplifies the problem, because it helps determining the parameters of a symmetry transform. For example, the presence of bilateral symmetry implies that the reflection axis (or the plane in 3D) passes through the centroid point of an object. The local detection methods search for parts of an object being symmetric. This is a more general and much more challenging task. However, Liu et al. (2010) point out that the global detection methods can be coupled with image segmentation algorithms to compete with the local methods. At the same time, some global detection methods allow a more detailed analysis of a symmetry pattern, such as detecting the type of a symmetry group.

Many modern symmetry detection methods involve local feature detection or construction algorithms (Lee and Liu, 2010; Loy and Eklundh, 2006; Mitra, Guibas, and Pauly, 2006). A local feature is a descriptor characterizing a located part of an image or an object in 3D space. The most common feature type is a point feature describing a small patch of an image or an object surface around a point. Though, other feature types, such as line features, are possible (Bokeloh et al., 2009). Normally, a feature descriptor is represented as a numeric vector; besides the location, it may represent the normal vector of a surface near the feature point, the curvature, the color gradient of an image, or something else. The purpose of a descriptor is to identify parts of an object that can be matched by a symmetry transform. A naive symmetry detection algorithm would search for all possible pairs (or triples in 3D) of matching features and associated symmetry transforms. Then, it would select such transforms that match the highest number of feature pairs (triples). Assuming the total number N of detected features may be high, the time complexity of the algorithm $O(N^2)$ ($O(N^3)$) would make the execution time too long. Therefore, to reduce the execution time, it is important to keep the number of matching features low by increasing the dimensionality of the feature descriptor. Loy and Eklundh (2006) have been particularly successful in this regard choosing 128-dimensional SIFT image descriptors (Lowe, 1999) for their symmetry detection algorithm. Their algorithm has been found being among the most efficient image-based algorithms in terms of detection rate and speed (Park et al., 2008).

Matching the pairs (triples) of local features is an essential step of a feature-based detection algorithm. A pair or a triple of matching features is associated with a configuration of a corresponding symmetry transform. The configuration representation depends on a symmetry type. For example, to describe the axis of reflection symmetry, Loy and Eklundh (2006) and Ogawa (1991) used the Hough transform determining the position of the axis line by the slope and the distance from the image center. For N-fold rotational symmetry, Loy and Eklundh (2006) used the original image coordinates determining the center of rotation. In both cases, following the logic of the Hough line detection algorithm, Loy and Eklundh (2006) discretized the parameter space and let every feature pair *vote* for corresponding symmetry transform parameter values. Thus, the result of feature matching

in this case was a grid of symmetry scores. In the case of 3D geometry, Mitra, Guibas, and Pauly (2006) used a more general 7D rigid transformation space, and Shi et al. (2016) used the logarithm mapping of the 7D transform. In their setting, the discretization using a dense grid would not be feasible due to enormous memory requirements. Hence, they represented individual matches as 7D points and used clustering methods to find plausible transformation configurations. The common aspect of all feature-based methods is searching for a transform parameter value that is shared (or voted) by a significant fraction of feature matches. Therefore, further in this thesis, I use the term *symmetry voting space* for any parameter space used to describe symmetry.

Once the candidate symmetry parameters are found, symmetry detection algorithms perform further processing steps according to the problem context, the application area, and the goal. Typically, a grouping of similar symmetries is a necessary step along with other refinement techniques. Some researchers apply global detection methods to identify symmetry groups corresponding to the detected symmetries (Lee and Liu, 2010). Others locate the object parts being part of a symmetry pattern (Mitra, Guibas, and Pauly, 2006) or build symmetry constellations for the structural analysis of the object shape (Chen et al., 2017; Mitra et al., 2013a).

MEASURING SYMMETRY Even though symmetry as a property of an object can be clearly defined, there is no consensus on how to compare symmetries seen in different objects. There is no common method to measure the amount of symmetry in an object as well.

One can say that the symmetry measure defines how close is the transformed version of an object matches the original version for a given symmetry transform. Thus, the symmetry measure should be defined for a pair object-transform, and not for the object alone. On the other hand, the object-transform symmetry measure can be generalized to the object-symmetry-type measure by taking the maximum value among all possible symmetry transforms of a given type (e. g. bilateral or rotational). Whether comparing symmetries of different types makes sense or not is debatable, because they are perceived differently for biological reasons (Goldmeier, 1937). Though, it is clear that the objective aspect of symmetry perception may be evaluated as the similarity between an original object and its transformed counterpart.

The need to measure symmetry arises naturally in symmetry detection algorithms when selecting the best candidate symmetry transforms. For instance, a direct approach to test if an image features reflection symmetry is to flip it along the candidate symmetry axis and check if all pixel values in the transformed image are the same as in the original image. However, this test is very sensitive to noise and partial occlusions. To overcome this problem, one can calculate the symmetry error measure as the average difference of the pixel values and compare it to a predefined tolerance level (Vasiliev, 1984). A slight generalization of this approach would be to get the maximum of the image autocorrelation, which is often used for detecting rotational symmetry (Krahe, 1986). For binary images and other types of input data, one could simply compute the overlapping area or volume for the original and transformed objects. One may suggest to use the overlapping area as the symmetry measure, but Liu et al. (2010) emphasizes that the overlapping area does not reflect the perception of symmetry well. Figure 2.2 illustrates this: the blue blocks seem to form a reflection symmetry pattern with a vertical axis; however, the reflected yellow blocks have a very small overlapping area with the blue blocks.

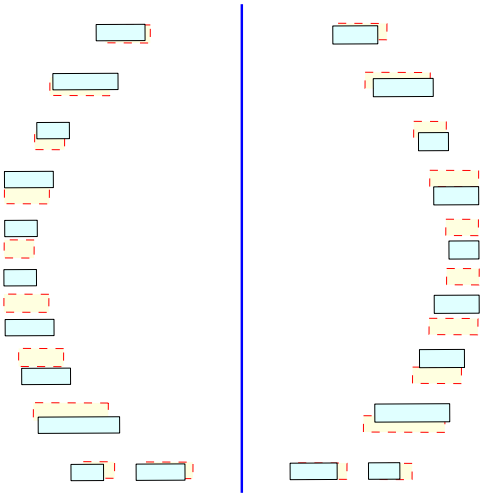


FIGURE 2.2: An almost symmetric pattern (blue) and its reflection (yellow). The patterns do not overlap much despite looking nearly symmetric.

Psychological and computer vision studies agree that the biological vision has evolved to be good at detecting edges due to edges being good at carrying information about the perceived scene (Morrone and Burr, 1988; Tolhurst and Dealy, 1975). Using this knowledge, one can measure the similarity of two objects as the distance between their surfaces in 3D or edges in 2D. This approach is not practical for raster images (because it requires reliable edges detection), but is used for 3D mesh geometry. Common ways to define similarity in this case are the Hausdorff distance or variations of integral point-to-point distances (Mitra et al., 2013b).

Another approach to measure symmetry appears in the feature-based detection algorithms: the value of a vote in the *symmetry voting space* can be used directly as the symmetry measure of a corresponding part of an image or an object. The advantage of this approach is its robustness. By design, the voting equations include penalty terms for imperfect matches. For example, the symmetry phase score used by Loy and Eklundh (2006) controls how well the orientations of two features correspond to a given transform.

The symmetry measures discussed above are induced by respective symmetry detection algorithms. Thus, they are not tailored to accurately reflect the perception of symmetry by the human vision. In psychology, a related concept called the *pattern goodness* was introduced and experimentally measured by Garner and Clement (1963). Howe (1980) tested the effects of partial symmetry on the pattern goodness. An early example of the research that aims to measure symmetry as it is perceived is the work by Yodogawa (1982). They introduced the *symmetropy* – the measure of symmetry based on the informational entropy. Yodogawa (1982) proposed to decompose an image into "primitive symmetries"

using Walsh basis patterns and to calculate the symmetropy as the entropy of the Walsh power spectra. They evaluated the predictive power of the symmetropy by correlating it to the pattern goodness.

The concepts of the pattern goodness and the symmetropy were developed for primitive patterns. Although being extendable, they have not been tested in psychological experiments in images or 3D geometry as far as I know. In contrast, Klapaukh (2014) conducted an experiment to measure the perception of symmetry in graphs as a part of his analysis of force-directed graph layouts. Klapaukh (2014) adapted the algorithm proposed by Loy and Eklundh (2006) to work on graphs by simply transforming edge positions to SIFT features. Then, he used the cumulative symmetry vote as the symmetry score of a graph. Welch and Kobourov (2017) in their experiment compared three mathematically-defined symmetry measures on graphs (including the measure proposed by Klapaukh (2014)) to the human perception of symmetry. Although they found that the metrics strongly disagreed on some test cases, they did not confirm the superiority of one metric over the others.

2.2 DETECTING SYMMETRY IN QUA-KIT DESIGNS

The context of the design exercises and the design data, as it is discussed in Chapter 1, puts the presented work in a special place among the existing research in symmetry detection. On the one hand, the design exercise lets students arrange building blocks on a two-dimensional plane. Hence, symmetry emerges in 2D, and there is no need to employ computationally expensive 3D algorithms, such as the one proposed by Mitra, Guibas, and Pauly, 2006. On the other hand, the polygonal data format potentially contains more information than raster images, since there is no need to detect objects. The graph problem addressed by Klapaukh (2014) is the closest one, but geometry of qua-kit designs is more complex than of the graph drawings of Klapaukh (2014).

To take the advantage of the geometric representation and the two-dimensional nature of the design exercise, I follow the approach of Klapaukh (2014). They adapted the algorithm of Loy and Eklundh, 2006 by constructing SIFT point features from graph edges. In the contrary, I propose to use edges of building blocks (i. e. building walls) as spatially extended features. As a result, a pair of edge features votes for a continuous region in the symmetry voting space rather than for a single point. In the following, I advocate this choice by discussing how the presented problem differs from the others.

SOURCES OF UNCERTAINTY In terms of robustness challenges, the presented work is closer to the 3D symmetry detection than to the image-based detection. Unlike the image-based problem, the design data does not have any distortions caused by a perspective projection mapping or by a structure of a photo lens. There is no partial occlusion, because there is no need for projection. However, there is a unique qua-kit-specific source of uncertainty: the imprecise positioning of building blocks. Keeping the editor interface as simple as possible, qua-kit allows students to move and rotate the blocks freely and does not aid this process via grid or edge snapping. Thus, a student may position a block a few degrees away from where they wanted, depending on how comfortable they are with the editor control interface.

FEATURE CORRECTNESS AND EXPRESSIVENESS As discussed in Section 2.1, expressive feature descriptors help to speed up a detection algorithm by reducing the number of potential symmetry matches. Mitra, Guibas, and Pauly (2006) successfully used surface normals and curvatures as feature descriptors of complex 3D objects. Unfortunately, the building blocks in the selected qua-kit exercise consist mostly of planes and right angles; this renders the curvature information useless and reduces the expressiveness of such features.

Loy and Eklundh (2006) used 128-dimensional SIFT descriptors to find matching feature point pairs. The SIFT had proven to be a very indicative and tolerant to noise and distortions in images. This choice made the symmetry voting space sparse. Loy and Eklundh (2006) applied the Gaussian blur filter on the voting grid and then searched for local maxima to find candidate symmetry transforms.

Klapaukh (2014) adapted this approach for graph drawings by encoding edge positions as SIFT descriptors. Graph edges could not provide as much information as patches of a raster image: instead of a rich histogram of color gradients, an edge had the length and the orientation only. Consequently, less symmetry matches were pruned at the voting stage. In my opinion, another drawback of this approach was that a pair of graph edges could vote for a single point in the voting space only. Figure 2.3 illustrates the problem: it presents two line segments and a region of possible reflection symmetry axes between them. For every point in this region, there exist one or more (imperfect) symmetry axes and a pair of points of the two line segments, such that these two points are symmetric with respect to the corresponding axes. The point features proposed by Klapaukh (2014) do not capture for this effect, whereas it can be especially important in the presence of very long edges.

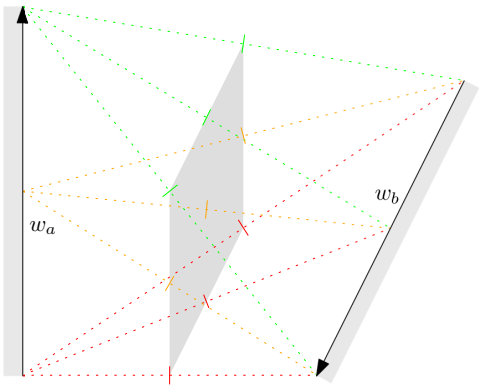


FIGURE 2.3: Possible axes of partial reflection symmetry for two oriented edges (walls). Every point in the gray region in the center lies in the middle of a line segment between some pair of points of the two edges.

Given the qua-kit data, the two possible alternatives to the wall (segment) features are the corner (point) features and the block (solid) features. In terms of psychology, the wall features are preferable, because a brain pays more attention to edges (Morrone and Burr, 1988; Tolhurst and Dealy, 1975). In terms of modeling, the solid structure features represented by polygons are too complex, and the corner features are not expressive enough.

SCALE PROBLEM If edges are used as spatially extended features, the edge length cannot be used as a feature descriptor. The only edge property left to be used for matching is the orientation. Alongside the homogeneity of the objects in the exercise (building blocks with mostly right angles), this contributes to the abundance of weak symmetry patterns. On the smallest scale, majority of building blocks are symmetric on their own: every rectangular block features rotation and reflection symmetry. Then, a pair of identical aligned buildings is symmetric too. On any larger scale there is a chance of symmetry occurrence; the greater the area of a region, the more objects it contains, the more potential symme-

try matches may occur. Obviously, I have to discard symmetries within objects, because design of an object is not a part of the student exercise. Beyond that, I need to make sure that small but prominent symmetry patterns do not get lost in weakly featured symmetry at the larger scale. Addressing this problem, Loy and Eklundh (2006) proposed to use the Gaussian distance weighting function to favor features that are closer to each other.

REPRESENTATION AMBIGUITY Two building blocks can be stacked together or even overlap. Then, they appear visually the same as a single building block (Figure 2.4). This causes a representation ambiguity: two groups of blocks may have different polygonal representation, but the same visual appearance and, hence, the same perceived symmetry patterns. The imprecise positioning issue adds to the difficulty: the aligned blocks may slightly vary in size or be positioned a small distance apart, while still looking as a whole in the eyes of a student.

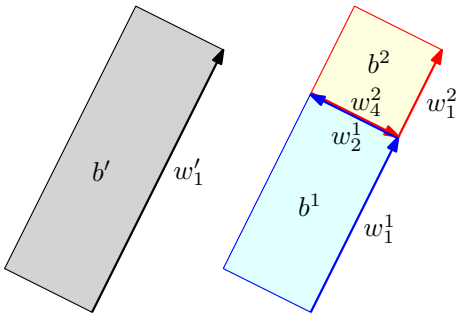


FIGURE 2.4: Two stacked building blocks (right) are not distinguishable from a single block of the same size (left).

The representation ambiguity challenge potentially could be solved by a polygon union algorithm. However, the imprecise positioning makes the polygon union unreliable. This is another reason for my choice of symmetry detection features: in terms of symmetry voting, the contribution of spatially extended wall features w_1^1 and w_1^2 together equals to the contribution of a single wall w_1' (Figure 2.4). The point features would fail in this case. Note, the adjacent walls w_2^1 and w_4^2 still contribute to the error, reflected as additional noise in the symmetry voting space.

GESTALT EFFECTS Perception of a design drawing is strongly affected by the effects described by the Gestalt laws of closure, proximity, similarity, continuation and good form (Soegaard, 2010). Figure 2.5 illustrates some of these effects on a sample design submission. The groups of building blocks form courtyards; perception fills gaps between buildings, and the formed shapes seem to exhibit some types of symmetry. However, a closer look at each courtyard reveals the absence of any symmetry in a strict mathematical sense. The arrangement of buildings and courtyards clearly demonstrates some *order* though. And, there are some blocks forming small symmetry patterns in every courtyard. In this study, I explore whether partial symmetry of design elements explains the perceived order of the whole design.

FIGURE 2.5: Gestalt effects strongly influences the design perception; so it is hard to draw a line between weak symmetry and other forms of order.

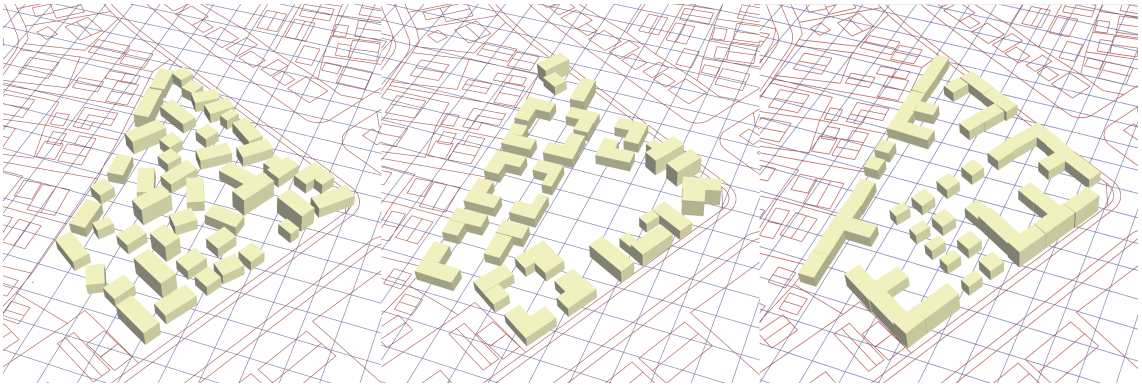


FIGURE 2.6: A qua-kit design exercise `EdX FC-03x-1`. Left: the design template. Center and right: student submissions.

2.3 METHODOLOGY

The task of a qua-kit design exercise is to redesign and submit a design in the 3D editor by moving and rotating objects (*building blocks*) on a plane. All students start with the same design template. A student can submit a design several times; only the last version of a design is shown in the design gallery. Figure 2.6 presents three example designs as they appear in qua-kit; the design on the left is the template, two others are student submissions. The technical details of the exercise and the user interface are presented in Chapter 1.

The size of a design submission in qua-kit is usually limited to $40 - 400$ editable building objects and a static environment consisting of polygons and lines. The presented exercise (`EdX FC-03x-1`) counts 47 polygonal building blocks. The blocks slightly vary in height, but I ignore this in the presented work. The environment consists of line markings on the ground and provides information about surrounding roads and buildings. The environment is not processed by the symmetry detection algorithm as well.

NOTATION Define a wall as a tuple $w = (c_w, d_w)$, where $c_w \in \mathbb{R}^2$ is the center of a wall, and $d_w \in \mathbb{R}^2$ is the orientation vector of a wall. A wall has no thickness; the width (size) of a wall is $2||d_w||$. Define a building block as a tuple $b = (c_b, W_b)$, where $c_b \in \mathbb{R}^2$ is the center of a building block $W_b = \{w_1^b, \ldots, w_{n_b}^b\}$, $n_b \geq 3$ is the list of building walls in the counter-clock-wise (CCW) order. Define $\mathcal{S} = \{b_1, \ldots, b_n\}$ be an urban design submission – a finite set of building blocks. Sometimes, I consider $W = \{w : \exists b_i \in \mathcal{S} : w \in W_{b_i}\}$ – a (finite) set of walls in a design submission.

Figure 2.7 presents a building block using the proposed notation. Note the CCW orientation of the walls: to the left of the vector d_w is always a building; there may be another building or the empty space to the right of the vector d_w. Building center is an average of all building corners: $c_b = \sum_{i=1}^{n_b} (c_{w_i} + d_{w_i})$. All walls must form a polygon without self-intersections but possibly with holes (courtyards). With these restrictions, a set of oriented walls fully describes an arbitrary 2D polygonal design drawing.

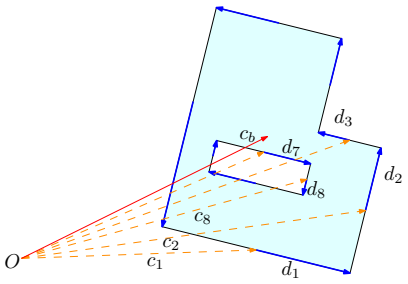


FIGURE 2.7: Representation of a building block. To simplify the notation I replace wall indices w_i^b by simple indices i.

The proposed algorithm matches all wall pairs to find symmetry pattern parameters. Although the algorithm needs to keep track of the correspondence between walls and blocks, the symmetry matching happens in pairs of walls rather that blocks.

To simplify the notation, I denote a pair of walls being matched as a and b meaning a pair of arbitrary walls w_i^a, w_j^b in two buildings a, b. The algorithm evaluates the *symmetry score* for all possible pairs of points on the walls a, b. I represent the points on the walls as follows:

$$\begin{aligned} p_a(s) &= c_a + sd_a, \\ p_b(t) &= c_b + td_b, \end{aligned} \tag{2.1}$$

where c_a, c_b are wall centers and d_a, d_b are CCW-oriented half-wall vectors; $s, t \in [-1, 1]$. I also use the normalized orientation vectors:

$$r_a = \frac{d_a}{||d_a||}, \qquad r_b = \frac{d_b}{||d_b||}. \tag{2.2}$$

Given an urban design submission $\mathcal{S}$, the (partial and approximate) symmetry pattern can be defined as a transform T and two sets of buildings $B_1, B_2 \subseteq \mathcal{S}$, such that $B_2 \approx T(B_1)$. Then, the problem of symmetry detection can be formulated as a search of T and maximal sets B_1 and B_2.

I consider three types of symmetries: reflection, rotation, and translation. In case of reflection and rotation, the maximal block sets are equal $B_1 = B_2$. The perfect translation is only possible for infinite frieze or wallpaper patterns. Thus, the proposed algorithm searches for the longest chain of translation transforms $B_n \approx T(B_{n-1}) \approx ... \approx T^{n-1}(B_2) \approx T^n(B_1)$. To unify the notation, I define $B = B_1 = B_2$ for the reflection and rotation and $B = \cup_{i=1}^{n} B_i$ for the translation.

The representation of the symmetry transform T depends on the type of symmetry. For the rotation symmetry, it is the number of folds k and the coordinates of the rotation center. The proposed algorithm is limited to a search of a fixed fold rotation (i. e. the algorithm must be executed n times for n different values k of the k-fold rotation). Hence, I can say that the k-fold symmetry transform is defined by two parameters – coordinates of the symmetry center. For the reflection symmetry, the transform is defined by the symmetry axis, which can be described by two parameters either. For the translation symmetry, the transform is defined by a 2D translation vector. Therefore, every symmetry transform considered in this work is represented by two parameters, denoted as τ_1, τ_2.

To estimate how well two walls a and b match under a symmetry transform T, I define the symmetry score $S_{ab}(\tau_1, \tau_2)$. To estimate the symmetry score of a pair of building blocks, one needs to sum up the scores of all possible pairs of walls. To estimate the symmetry score of a pattern, one needs to sum up the scores of all possible pairs of blocks:

$$S_B(\tau_1, \tau_2) = \sum_{b_i \in B} \sum_{b_j \in B \setminus \{b_i\}} \sum_{w_k^i \in W_{b_i}} \sum_{w_l^j \in W_{b_j}} S_{w_k^i w_l^j}(\tau_1, \tau_2). \tag{2.3}$$

There is no reason to assume that different symmetry types are perceived equally well by a person. For instance, experimental studies suggest that reflection symmetry is recognized best, especially if the symmetry axis is vertical (Welch and Kobourov, 2017). Therefore, I maintain the bilateral reflection, translation, and k-fold rotation symmetry scores separately, denoted as S_{b}, S_{t}, and $S_{\mathrm{r}k}$ respectively.

PROBLEM STATEMENT The symmetry score $S_B(\tau_1, \tau_2)$ is defined for any symmetry transform and any group of buildings. Hence, the goal of the algorithm is to find such a pair of parameters τ_1, τ_2 and a set of buildings B to maximize the score. Furthermore, a design submission $\mathcal{S}$ might have no symmetry patterns or might have several of them. Thus, I have to introduce the minimum threshold score S_0; if the score of a pattern is not smaller than S_0, the pattern is accepted. The value S_0 depends on the symmetry type, on the number of objects, and on the geometry itself; in this work, I find it statistically, by using the submissions and the perception data of the exercise.

Combining the definitions together, I formulate the symmetry detection problem as follows: given a design submission $\mathcal{S}$, a symmetry type $\mathrm{s} \in \{\mathrm{b}, \mathrm{t}, \mathrm{r}2, \mathrm{r}3, ..., \mathrm{r}k_0\}$, and the minimum symmetry score $S_{\mathrm{s},0}$, find all symmetry transforms $T_{\mathrm{s}} \equiv (\tau_1, \tau_2)$ and corresponding maximal sets of building blocks $B \subseteq \mathcal{S}$, such that $S_{\mathrm{s},B}(\tau_1, \tau_2) \geq S_{\mathrm{s},0}$.

2.3.1 METHOD OUTLINE

As it was discussed in Section 2.2, I propose to use spatially extended oriented line segments (walls) as local features for detecting symmetry. Unlike commonly used point features, walls consist of an infinite amount of points. Thus, a pair of walls may contain an infinite number of point pairs contributing to a symmetry pattern. While it sounds complicated, the pattern parameters can be represented and bounded analytically and evaluated on a grid in the *symmetry voting space*. As a result, a pair of walls always *votes* for a small range of parameter values in a symmetry voting space instead of a single point. This approach has an additional benefit that the voting score function is continuous in the voting space and does not need to be smoothed on a grid before further processing steps, such as the search of the score maxima.

Similar to the works of Loy and Eklundh (2006) and Mitra, Guibas, and Pauly (2006), the proposed algorithm consists of the following stages:

1. *Symmetry voting*: traverse through all block pairs that are not further apart than a pre-defined constant distance; traverse through all wall pairs in corresponding blocks and add up the scores $S_{\mathrm{s}}(\tau_1, \tau_2)$ on a grid in the voting space $\mathcal{T}_1 \times \mathcal{T}_2 \subseteq \mathbb{R}^2$.

2. *Parameter search*: on the voting grid, find at most a constant number of local maxima satisfying $S_{\mathrm{s}}(\tau_1, \tau_2) \geq S_{\mathrm{s},0}$; find corresponding symmetry transform parameter values (τ_1, τ_2).

3. *Object grouping*: for a list of most promising symmetry transform parameter values, traverse through all pairs of blocks again to find if a pair belongs to a symmetry pattern with given parameter values; for every pattern B and parameters τ_1, τ_2, calculate the pattern score $S_{\mathrm{s},B}(\tau_1, \tau_2)$.

4. *Pattern unification*: apply heuristics to compare the detected patterns; merge similar patterns, exclude or create new patterns according to the symmetry group laws.

The output of the algorithm is a set of symmetry patterns that agree and complement each other. A single output pattern is a parameter pair (e.g. reflection axis or rotation center) and a set of blocks constituting the pattern.

2.3.2 SYMMETRY VOTING

As the first stage of the symmetry detection algorithm, I construct a grid in the symmetry voting space; pairs of building walls vote for small patches of the voting grid. The better the walls are aligned w.r.t. the symmetry type, the higher symmetry score is written into corresponding points in the voting space.

As it was discussed in the beginning of this section, the symmetry voting space is two-dimensional: $\mathcal{T}_1 \times \mathcal{T}_2 \subseteq \mathbb{R}^2$. The three symmetry types considered in this work have similar structure of the score function:

$$S_{ab}(\tau_1, \tau_2) = \Phi^p_{ab}(\tau_1, \tau_2) L_{ab}(\tau_1, \tau_2) D_{ab}, \tag{2.4}$$

where parameters τ_1, τ_2 are the coordinates in the voting space (their meaning depends on the symmetry type); a and b are walls of two distinct building blocks. Following the approach of Loy and Eklundh (2006), I define the symmetry score in Equation 2.4 as the product of three components:

- $\Phi^p_{ab}(\tau_1, \tau_2)$ is the phase score. Φ is defined as a cosine similarity of the observed and the ideal angles between d_a and d_b. p is a parameter controlling how strict the algorithm should be in deciding whether to consider two walls aligned: the higher the value of p, the steeper the curve of the diminishing score. In practice, I choose values $p = 8$ or $p = 16$.

- $L_{ab}(\tau_1, \tau_2)$ is the length score. It measures the fraction of the walls being part of the symmetry pattern.

- D_{ab} is the distance (scale) score. It is used to discount the symmetry score of the blocks being far apart and does not depend on the symmetry type. I use a Gaussian distance weighting function $D_{ab} = \exp\left(-\frac{||c_b - c_a||^2}{2\sigma_D^2}\right)$ if I want to bound the maximum scale for the pattern detection, or a constant function $D_{ab} = 1$ otherwise. Here, σ_D is a predefined constant that controls the maximum detection scale.

Algorithm 1 Symmetry voting process

```
procedure SymmetryVoting(Submission 𝒮)
    G ← initialize the voting grid with zeros.
    for all blocks a ∈ 𝒮 do
        for all blocks b in a neighborhood of a do
            for all wall pairs w_a ∈ W_a and w_b ∈ W_b do
                τ1 min, τ1 max, τ2 min, τ2 max ← calculate the patch extents
                for all τ1 ∈ [τ1,min, τ1,max] and τ2 ∈ [τ2,min, τ2,max] do
                    G(τ1, τ2) ← G(τ1, τ2) + S_{w_a w_b}(τ1, τ2)
    return G
```

IMPLEMENTATION At first, I need to initialize the *symmetry voting grid* – a discrete representation of the score in the symmetry voting space. For all symmetry types, a two-dimensional array (i. e. a matrix) is sufficient to represent the voting grid. The submission spatial extent and the structure of the symmetry transforms define the quantization of the voting space. For example, rotation centers can be found within the bounding box of submission, the angle of the reflection axis is defined to be within $[0, \pi)$. The extent of possible parameter values is evenly divided into $m \times m$ cells. For the qua-kit exercises, I use either 128×128 or 256×256 grids. The grid is initialized with zeros, indicating no symmetry found yet.

Every pair of walls potentially may have a positive symmetry score at any cell in the voting grid (i. e. for any possible symmetry transform). Assuming the submission contains n walls, a naive implementation would traverse through all wall pairs and wall grid cells. On every iteration, it would add the wall-to-wall score $S_{ab}(\tau_i, \tau_j)$ (possibly the zero score) to the value of a current cell (i, j). Asymptotically, this process would take $O(n^2 m^2)$ steps.

I use two methods to improve the execution time over the naive algorithm. First, I have implemented a quadtree structure to store and query the blocks. A quadtree allows to search for an object in 2D space in $O(\log n)$ time. If one wants to detect symmetry pattern not larger than some predefined constant size, the algorithm does not need to traverse through all wall pairs. Define k as the maximum possible number of walls within the predefined distance; then the traversal of all wall pairs in neighborhoods of this size takes $O(n(\log n + k))$ steps. This is useful for large submissions, though it does not affect the performance as much on the given qua-kit exercise. Second, I use an observation that the symmetry matching for two walls can be positive for a small patch of the voting grid only. Assume l is the average length of a wall in the submission, measured in cell widths. Then, the proposed voting algorithm takes $O(l^2 n(\log n + k))$ steps, traversing through the small patches rather than the whole grid for every pair of walls.

The Algorithm 1 summarizes the implementation notes in pseudocode. The presented algorithm does not specify the way to calculate the size of a patch to be updated and the formula for calculating the score $S_{ab}(\tau_1, \tau_2)$. Given the score decomposition presented in Equation 2.4, three pieces of the implementation need to be specified for each symmetry type:

- equations for calculating extents $\tau_{1\,\min}, \tau_{1\,\max}, \tau_{2\,\min}, \tau_{2\,\max}$;

- phase score $\Phi_{ab}(\tau_1, \tau_2)$;
- length score $L_{ab}(\tau_1, \tau_2)$.

The definition of the voting space and these three formulae are derived in the Appendix A.

2.3.3 PARAMETER SEARCH

The second stage of the proposed symmetry detection algorithm is the search for the best symmetry transforms. This amounts to finding maximums on the voting grid. Since the submission may have zero or many symmetry patterns, the search involves some heuristics for deciding how many local maximums to count. I have implemented a simple iterative procedure: find the global maximum on the grid; if it passes the validity checks then save the point, subtract the score from the found and neighbor cells and repeat the procedure. The algorithm performs the following validity checks at each iteration:

- check if the score is greater than the global constant minimum S_0;
- check if the score is greater that the fraction of the best score for this submission (e. g. $1/10$ of the best score);
- stop if more than a predefined constant number of maximums have been already found.

The output of this stage is the list of symmetry transform parameters.

2.3.4 OBJECT GROUPING

After the candidate symmetry transforms are found, the algorithm searches for corresponding blocks to form symmetry patterns. Essentially, this means the search for wall pairs that have a positive score for the given symmetry parameters. However, to avoid incomplete patterns, I have had to implement a specific search algorithms for each symmetry type rather than using a mere traversal.

REFLECTION SYMMETRY A reflection symmetry pattern is built from pairs of blocks on opposite sides of the symmetry axis. The issue arises when several pairs of blocks are too far apart from each other. The symmetry voting process estimates the position of a line (symmetry axes), but does not estimate a specific line segment. Thus, two independent symmetry patterns having the same symmetry axis would be recognized as one.

To solve the problem, I implement a special data structure: a tree-based map of disjoint line segments in one-dimensional space. When a line segment is added to this map, it checks if the segment overlaps with any other segment present in the map. If it does, the union of the overlapping segments is calculated, otherwise the new segment is added. I use this data structure to associate segments on the symmetry axis with lists of block pairs, such that the segments cover the projections of the blocks onto the symmetry axis. As a result, I get a few disjoint line segments and corresponding sets of blocks. If a segment

corresponds to a single block pair, or if its aggregate symmetry score is too low, the pattern is discarded.

ROTATION SYMMETRY A complete k-fold rotation symmetry pattern must consist of at least k blocks. To satisfy this requirement, I maintain chains of blocks rather than pairs. First, I detect all possible pairs and put them into a graph; the blocks are the graph nodes and the CCW rotation symmetries are the oriented weighted graph edges. Second, I search for all paths in the graph that satisfy a few requirements:

- none of the paths is fully contained in another path;
- no two paths start with the same node (block);
- paths maximize the weight – the cumulative symmetry score;
- the length of a path is at least k (or $k-1$ if almost-complete chains are allowed).

The union of all found graph paths (chains of blocks) forms the set B.

TRANSLATION SYMMETRY The translation symmetry case is almost the same as the rotation symmetry. The only difference is the minimum chain length is constant 2 instead of k.

2.3.5 PATTERN UNIFICATION

The last stage of the symmetry detection algorithm is to eliminate the duplicate patterns and refine the remaining patterns based on relations between them. At this point, the program has the complete information about the detected symmetry patterns. The work to be done at this stage is to simply try a few unification rules on pairs and groups of the detected patterns. I have implemented the following rules:

- *Joining similar patterns:* if two patterns have approximately equal parameter values, and none of the block sets is a strict subset of another, then their parameter values are averaged and the block sets are united.
- *Eliminating surpassed patterns:* if two patterns have approximately equal parameter values, and the block set of one is fully contained in the block set of another, then the former is removed from the result.

There is a potential for implementing many more rules stemming from the group theory. For example, presence of a 2-fold and 3-fold rotation patterns with the same sets of blocks implies a 6-fold rotation pattern. Unfortunately, I could not incorporate these rules due to the time constraints.

2.4 EVALUATION

I have tested the algorithm on synthetic data and on the qua-kit exercises. The synthetic data consists of a few blocks that form clearly pronounced symmetry patterns. It comes in two variants: a set of perfect procedurally generated shapes featuring dihedral group symmetries and a few hand-drawn shapes featuring imperfect but clearly distinguishable symmetry patterns. The qua-kit designs presented in this chapter are submitted for the MOOC exercise `EdX FC-03x-1`. The exercise details are described in Chapter 1.

2.4.1 SYMMETRY VOTING

The main contribution of this work to the symmetry detection research is the symmetry voting algorithm. Figures 2.8, 2.9, and 2.10 illustrate the results of this algorithm. As the first stage of the symmetry detection, it gets two-dimensional geometry and returns a grid of symmetry scores in the symmetry voting space. The figures show the input geometry in the first column of pictures and visualize the output grids in the rest of the columns. The grid pictures represent the symmetry score in normalized grayscale: white color corresponds to the zero score, black color corresponds to the maximum observed symmetry score. Thus, local score maximums corresponding to optimal symmetry transform parameters are seen as distinct black spots on the pictures.

Figure 2.8 presents the results of the voting process on perfectly symmetric shapes. One can easily find out symmetries of the shapes without looking at the first column by counting the black spots. For example, the top shape has one 2-fold rotation center and one vertical reflection axis (in-block horizontal reflection is not detected). The bottom shape has ten reflection axes, one 2-fold rotation, and one 5-fold rotation, which indicate D_{10} dihedral symmetry group.

Note the reflection voting pictures (the second column in Figure 2.8), it may seem not intuitive at first glance. The horizontal axis of the reflection symmetry grid is the angle of the symmetry axis. The angle value is cyclic: the middle of the grid corresponds to the horizontal orientation, the left and right sides correspond to the vertical orientation. Thus, the two spots on the reflection picture in the first row correspond to the same vertical reflection axis. The vertical axis of the grid is the distance from the center to the symmetry axis in the shape image coordinates. Most of the reflection axes in Figure 2.8 pass through the center in the first column of pictures, so the black spots lie on horizontal lines. The line of the black spots in the second row is tilted because the center of the shape is not aligned to the center of the picture.

Figure 2.9 presents the results of the voting process on hand-drawn shapes, which feature imperfect but clearly distinguishable symmetries. The black spots on the pictures look blurred and slightly fade in the noise. Among the rotation symmetries, 3-fold is detected best, because it is characterized by 60° angles: in urban design, 60° angles are not as common as 90° angles and pop-up more easily.

Figure 2.10 presents the results of the voting process on qua-kit submissions. The submitted designs exhibit diversity of unique ideas. Despite many of them *seem* to have partially

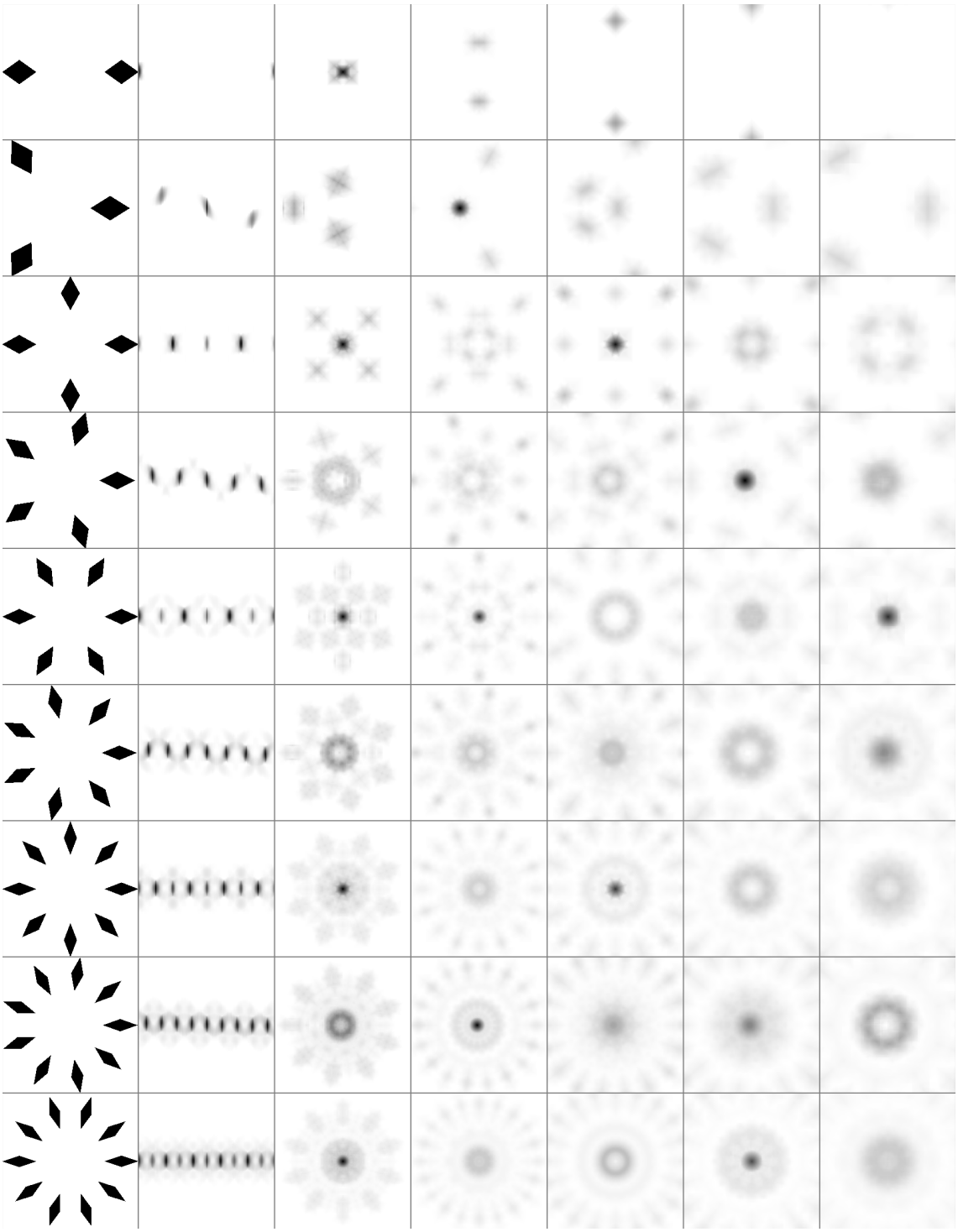


FIGURE 2.8: Symmetry voting results for a synthetic geometry. The black spots in the voting space pictures clearly point to the symmetry score maximums.

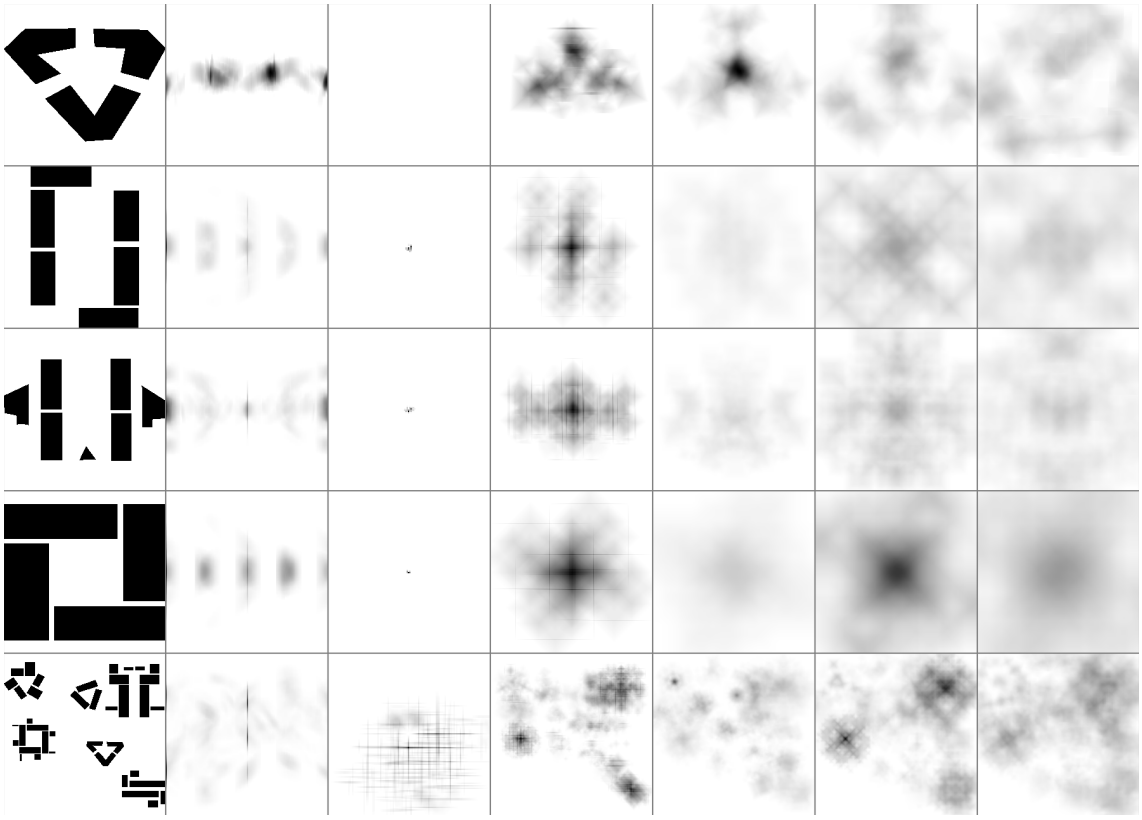


FIGURE 2.9: Symmetry voting results for a synthetic geometry. Imperfect symmetry patterns are distinguishable, though the presence of noise makes the automatic detection harder.

symmetric elements, only a few submissions appear to feature *proper* symmetric patterns. Usually, these are repeated or mirrored arrays of similar blocks (e. g. rows 2, 3, 8 in Figure 2.10). Reflection symmetry is loosely present in arbitrary shapes (e. g. rows 1, 5, 9). The submission in row 4 is a rare example of a complete 5-fold and 7-fold symmetry. The 5-fold symmetry maximum is clearly visible in the last column; 7-fold symmetry voting grid is not included in the figure.

Figure 2.10 shows that the computed score in the symmetry voting space is rather noisy in real design submissions, as the black spots of symmetry peaks fade in the predominantly grayish voting grids. The main reason for this is the regularity of design elements: abundance of right angles and similar block dimensions. The regularity of the elements encourages students to align the elements, thus introducing more false score maximums in the symmetry voting space. Nevertheless, proper complete symmetry patterns stand out very well when they are present in a design. If a submission features no proper symmetry patterns, the voting grid may appear confusing and cause false symmetry detections. However, the voting grid still can be useful in this case: it reveals the information about the structure of the design. It can be used to measure the *weak signs* of symmetry when an object does not exhibit any symmetry patterns in the strong sense.

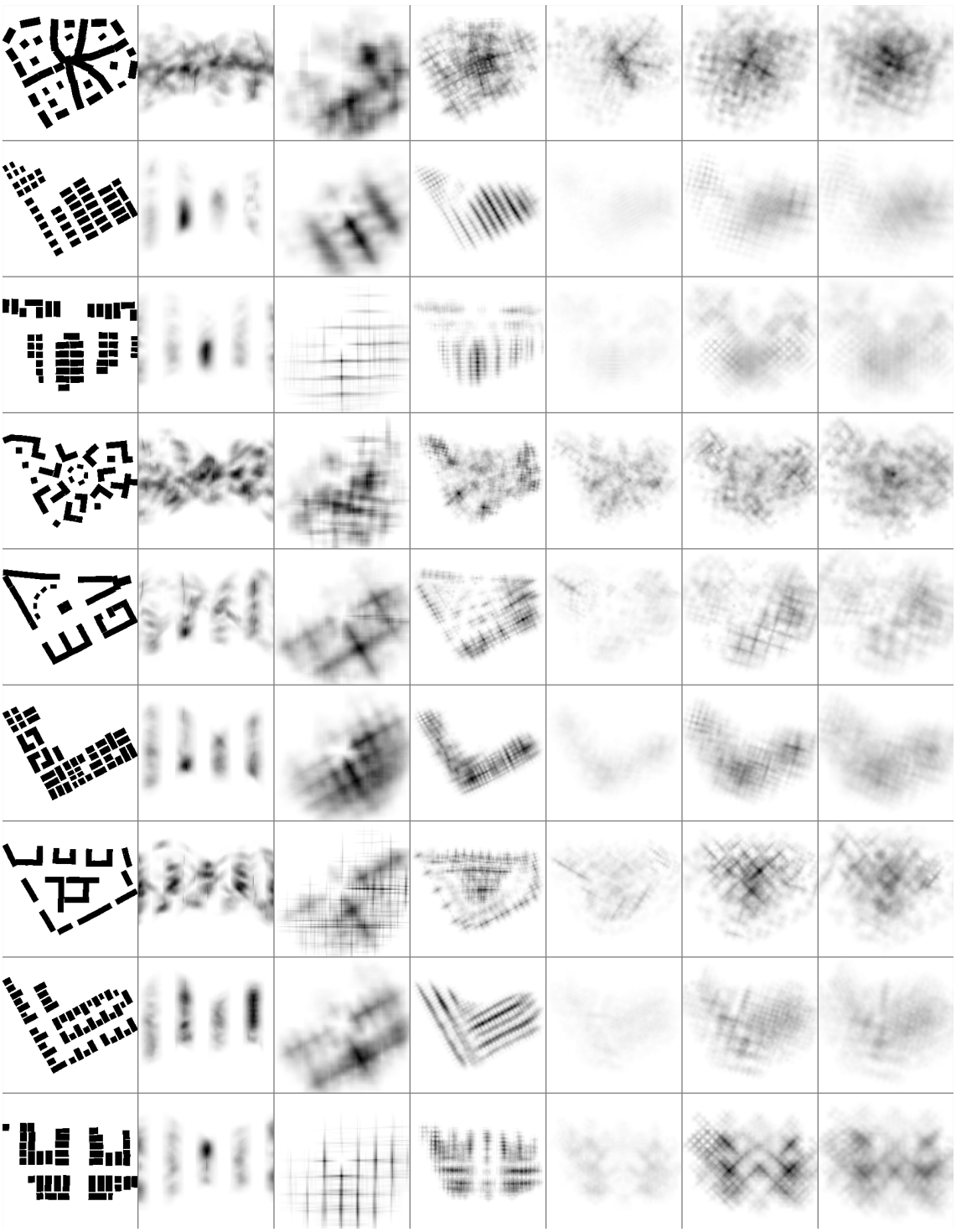
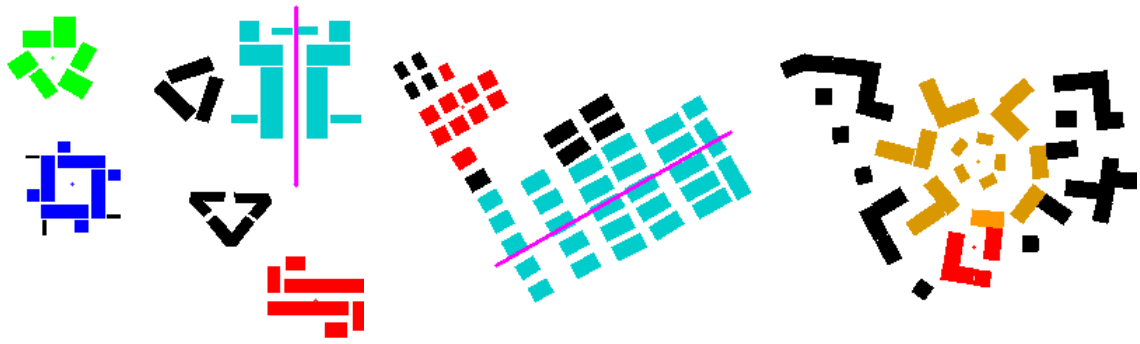


FIGURE 2.10: Symmetry voting results for the exercise `EdX FC-03x-1`. Complete symmetric patterns are rare in real submissions and are far from ideal. Nevertheless, "good" patterns stand out as clear black spots in the voting space pictures.



(a) Synthetic test (b) Grid-like submission (c) Circular-style submission

FIGURE 2.11: Results of symmetry detection – pattern locations and block groups. The blocks are colored according to the detected patterns: **cyan** – reflection symmetry; **red** – 2-fold rotation; **green** – 3-fold rotation; **blue** – 4-fold rotation; **orange** – 5-fold rotation. Only a few most prominent (almost) non-overlapping patterns are shown on these pictures.

2.4.2 PATTERN DETECTION

After the voting grid is computed, the next symmetry detection stages use it to find the best transform parameters and construct the patterns. This part of the algorithm is not fundamentally different from the other feature-based symmetry detection approaches. Though, some circumstantial adaptations have been necessary for the implementation due to differences in the data representation.

Figure 2.11 illustrates the results of the symmetry detection algorithm by visualizing some of the detected patterns in different colors. Without additional hand-tuned thresholding, the algorithm may return 10-20 candidate patterns, many of which overlap. In Figure 2.11, I have manually selected the patterns with the highest symmetry scores discarding significantly overlapping patterns.

Figure 2.11a represents the detected patterns in the hand-drawn shapes featuring imperfect but explicit symmetry. The four most significant patterns have been detected with hight symmetry scores. However, the figure does not show the weaker patterns that I have discarded due to overlaps with the presented patterns. For example, the algorithm has detected a 2-fold rotation in place of the displayed 4-fold rotation (blue). Interestingly, the 2-fold pattern includes the two thin rectangles on the lower-right and upper-left corners of the 4-fold pattern; the 4-fold pattern does not include them due to the incompleteness but still has a higher symmetry score. Note, two triangular blocks groups (black) have not been recognized as 3-fold symmetry; apparently, their angle values are too far from 60°.

Figures 2.11b, 2.11c represent the detected patterns in the exercise submissions. These examples show that the detection is tolerant to small deviations in geometry. Note the 5-fold pattern in Figure 2.11c: the outer circle of blocks is assigned to the 5-fold pattern, even though it should be recognized as 7-fold symmetry. This is a limitation of the algorithm: the higher fold numbers mean the smaller fold angles, which in turn mean the lower tolerance to imperfect block positioning.

The submissions shown in Figures 2.11b, 2.11c are ones of the best in terms of presence of symmetry. For the majority of submissions, the algorithm finds a large number of small rudimentary patterns. On the one hand, I believe this reflects the reality: the students have not been asked to design symmetric urban districts. On the other hand, the parameter search, the object grouping, and the pattern unification could be improved in some ways to achieve better results.

2.4.3 EXECUTION TIME

The execution time has been one of the priorities during the development of the symmetry voting algorithm. Currently, the reflection and translation symmetry voting take approximately 0.2 seconds and the rotation symmetry voting takes approximately 3.5 seconds for one submission. Symmetry voting including three variants of rotation voting and the rest of the detection steps takes approximately 16 seconds. The synthetic tests run for approximately 10 seconds per drawing.

2.5 DISCUSSION

The proposed symmetry voting algorithm takes a special place in the symmetry detection research: it is based on the 2D image symmetry detection works (e. g. Loy and Eklundh (2006)), but has to process polygonal geometry data similarly to the 3D symmetry detection works (e. g. Mitra, Guibas, and Pauly (2006)). In theory, the use of spatially extended edge features allows me extract more information about symmetry in objects. Effectively, I avoid a class of issues related to the quality of feature detection by using all available geometry for the symmetry voting.

My original goal of the symmetry detection in urban design has been to use it for the interactive optimization (Chirkin and König, 2016). Given the qua-kit design representation, a conventional optimization algorithm has to search over the space of all possible block positions and orientations. This is a very computationally expensive task, so various heuristics have been invented to simplify it. Instead of using optimization heuristics, I have suggested to reduce the complexity of the problem by introducing natural constraints. First, I have proposed an interactive process, such that the optimization algorithm can do small changes only and is invoked by a user interactively. Second, I want to use detected symmetry patterns as emerging optimization constraints: an optimization algorithm cannot move blocks in a way that breaks existing symmetries. This should drastically reduce the dimensionality of the optimization problem and speedup the algorithm. However, I have not yet reached the point when symmetry detection can be fast and reliable enough for this purpose. At the same time, the symmetry analysis has proven to be worth studying on its own.

The test results show that detecting symmetry in the exercise submissions is much harder than in the synthetic data. Analysis of the voting grids and the detection output suggests several reasons for this:

1. the choice of local features – edges – does not allow representing some of symmetry patterns perceived by the human vision;

2. the detection of score maxima and the block grouping are not robust enough;

3. the exercise submissions do not have as much symmetry as it has been anticipated.

The first potential reason is the way people perceive groups of building blocks. The size of the exercise scenario allows a submission to feature up to two levels of details: there are just enough building blocks to outline small clusters of buildings enclosed by roads, while still keeping thin pathways between individual blocks (see Figure 2.10). As explained by the Gestalt grouping laws, a cluster of blocks is perceived as a whole. As a result, a person searches for symmetry among the clusters rather than individual blocks. However, the algorithm takes into account all blocks separately and detects all inconsistencies in block sizes and positioning. Some of the block walls still count towards the symmetry patterns among the clusters. However, the length of walls on a cluster boundary grows slower than the total length of the walls inside a cluster. Thus, the voting score contributed by the boundary walls fades in the noise for a sufficiently big cluster. A solution to this problem is to run the symmetry detection algorithm either for a single group of buildings or for generalized geometry of the group itself rather than for individual buildings. The building generalization problem is successfully studied in geographic information system (GIS) research (Wang et al., 2015; Yan, Weibel, and Yang, 2008). A promising direction for future research would be to integrate the symmetry detection algorithm with a building generalization algorithm.

The second potential reason is the stages of the algorithm past the symmetry voting. I have implemented the most straightforward approach for the search of symmetry score maxima and for the block grouping. One may try a few other techniques to improve the performance of these stages. For example, I add two blocks to a symmetry pattern if at least one pair of their walls matches, whereas the blocks may better be discarded in some cases.

The last potential reason for the algorithm not meeting the expected performance in case of the design submissions is the way people might think about symmetry. Perhaps the symmetric form of building blocks and the Gestalt effects trick the human perception into overestimating the amount of symmetry in a design drawing. To explore this problem, I suggest that the biological vision recognizes some weak signs of symmetry, even if they do not add up to a distinctive pattern. Then, these signs are nothing but the sense of alignment, proportion, and *order*.

Acknowledging the concept of the relation between order and symmetry, I propose to extend the idea of the symmetry detection and the symmetry measure to the idea of the order detection and the order measure. In this context, symmetry becomes a feature of extreme order: the presence of distinctive symmetry patterns suggests a highly ordered design, but the fact of being ordered does not imply any particular symmetry pattern. This conjecture can be tested if there is a way to measure the perception of order. Chapter 3 describes an experimental study that aims at evaluating the perception of order in the exercise submissions. At the same time, I can use the symmetry voting grids to derive a

metric to reflect the measure of order. Combining these two gives a computational model of the perception of order. The goal of this thesis is to answer if this model has a predictive power and explore what it can tell about the perception of symmetry and order in urban design.

3

THE CROWDSOURCED ORDER MEASURE

In this chapter, I describe an experimental study that I have conducted using qua-kit to analyze the perception of order in two-dimensional design drawings. I have asked students of an online course to do a series of comparisons to select "a design that features more order" in pairs of randomly chosen qua-kit submissions. Using this comparison data, I construct a crowd-sourced measure of order.
The key component of the study is the Bradley-Terry paired comparisons model. I analyze the behavior of the model and its applicability to the problem and the available data. Then, I describe the experiment setup, the collected data, and the model output. In the end, I discuss the results in connection with the analysis of symmetry and order in qua-kit designs.

Chapter 2 proposes computational methods to evaluate symmetry and order. This chapter approaches the problem from another direction: I ask individuals to assess the orderliness of a design. Potentially, this approach can help to analyze how order is perceived by a person, without even understanding the corresponding properties of a design. However, its major challenge is that people do not perceive orderliness in absolute values. It is much easier to tell if one object is more ordered than another rather than grading a single object using a scale.

A similar idea is discussed in Chapter 1 in the context of the exercise grading system. Section 1.4.1 discusses the problem of ranking designs and voters based on paired comparison tests. Even though the topics have the same roots in the ranking works, such as the Bradley-Terry model (Bradley and Terry, 1952), they are significantly different. The exercise grading requires maintaining two ratings (voters and designs) and needs to be adjustable in the face of new votes online. The order measure discussed in this chapter needs to be constructed only once, given the full available evidence.

The experiment I describe in this chapter belongs to a well-studied class of paired comparison tests. In the following, I use one of the most developed models and modify it to get most information out of the available experimental data. The output of the model is the scores assigned to each design submission in the qua-kit exercise; I use these scores as the measure of perception of order in qua-kit designs.

3.1 STATE OF THE ART

A paired comparison test is a setting that can be abstracted as a series of experiments on a set of items; in every experiment, two items are compared in a pair. Given the set of

items and the series of experiments, one wants to estimate the *scores* of the items. This test can represent individuals or teams playing a competitive game (e. g. chess, soccer), where the score stands for the skill of a player. Alternatively, this can be a psychological test for measuring the perception of some stimuli, such as the preferences for different types of food or services. In the latter case, an item is usually referred as a psychological stimulus, and its score is referred as the scale of a stimulus. In this chapter, an item is an exercise submission, and the score is the perceived order measure of a submission. In my experiment, a person is asked to choose "a design that features more order" in a pair of randomly selected exercise submissions.

Denote $a_1, \dots a_n$ as the items – n objects of the experiment. Then, define $\pi = \pi_1, \dots \pi_n$ as the scores of the items. Given the series of m paired comparisons, the problem is to estimate the scores π. Normally, this is done statistically: scores $\pi_1, \dots \pi_n$ are modeled as random variables and assumed to belong to a single family of distributions. Once the scores are estimated, they can be used to predict the outcomes of new comparisons. There are two commonly used models of the paired comparison test.

Based on precedent psychological studies, Thurstone (1927) stated the law of comparative judgment. The law establishes a connection between results of comparing pairs of psychological stimuli and the psychological scale of the stimuli. The law is formulated as follows:

$$\pi_i - \pi_j = x_{ij}\sqrt{\sigma_i^2 + \sigma_j^2 - 2r_{ij}\sigma_i\sigma_j}, \tag{3.1}$$

where x_{ij} is the fraction of times item a_i has been preferred over item a_j, σ_i is the standard deviation of score π_i, and r_{ij} is the correlation between scores π_i and π_j. A notable use case V in the work of Thurstone (1927) assumes that the standard deviations of the scores are equal and the correlations are zero. The case V greatly simplifies the analysis of the law; it was studied statistically by Mosteller (1951) and eventually became known as the Thurstone-Mosteller model. This model assumes the normal distribution for scores and their differences $(\pi_i - \pi_j)$.

An alternative is the Bradley-Terry model, named after the research article by Bradley and Terry (1952). Although, to the best of my knowledge, the first work on the same model was published by Zermelo (1929). Both papers interpreted the relation between items and comparisons the same way. In this model, a constraint $\sum_{i=1}^{n} \pi_i = 1$ is usually added to bound the scores of items. The probability that item a_i is preferred to item a_j is defined as follows:

$$\Pr(\text{item } a_i \text{ beats item } a_j) = \frac{\pi_i}{\pi_i + \pi_j}. \tag{3.2}$$

If one substitutes $\psi_i = \log \pi_i$, this model implies the logistic distribution of the score difference $(\psi_i - \psi_j)$. Thus, the Thurstone-Mosteller model and the Bradley-Terry model are usually opposed as the models based on the normal distribution and the logistic distribution respectively.

In the presented formulation, both models are limited to a binary comparison and do not allow draws. However, there have been proposed a few generalizations to the models. For example, Bradley and Terry (1952) proposed a multiple-way comparison setting: a judge would arrange two or more items in the order of their preference. A few authors noted a straightforward extension to allow for ties in a binary comparison by simply counting

half-win for both alternatives (e. g. this approach was used by Kaye and Firth (2017) and Turner and Firth (2012)). However, Henery (1992) claimed that such an extension would introduce a bias to the estimated scores; thus, they proposed another, explicit way to extend the Thurstone-Mosteller model to allow for ties. They also noted that a similar extension would be possible for the Bradley-Terry model and would yield very similar results.

A number of studies compared the two models with mixed results; both of them performed similarly on most data sets, although psychological studies sometimes preferred the Thurstone-Mosteller model (Luce, 1977). Handley (2001) strongly advocated the use of the Bradley-Terry model over the Thurstone-Mosteller model due to the former being simple and well-studied. They argued that both models had performed similarly, but the Bradley-Terry model allowed for a more elaborate data analysis via developed statistical hypothesis tests. Handley (2001) also noted that the Bradley-Terry model could handle incomplete data better than the Thurstone-Mosteller model.

Stern (1990) proposed a paired comparison model representing performance of items in terms of the gamma distribution. They showed that their gamma comparison model generalized Bradley-Terry and Thurstone-Mosteller models by substituting the shape parameter α of the gamma distribution with fixed values $\alpha = 1$ and $\alpha \to \infty$ respectively. Furthermore, they tested the model on real and simulated data numerically and found that the difference between the estimated scores for various values of α was negligible unless the comparison sample was very large (i. e. thousands comparisons per item). Intuitively, this can be explained by the fact that the cumulative distribution functions of the normal and logistic distributions have extremely similar shapes under certain parameter values. That being said, I have decided to use the Bradley-Terry model for my experimental study.

ESTIMATING THE SCORES Define $w = \{w_{ij}\}_{i,j=1}^{n}$, w_{ij} is the number of times item a_i it preferred to a_j in a given data set; $w_{ii} = 0$, and $w_i = \sum_{j=1}^{n} w_{ij}$. Then, one can construct a log-likelihood function for the model presented in Equation 3.2 given the series of observed experiments:

$$\log \mathcal{L}(\pi|w) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \left(\log \pi_i - \log(\pi_i + \pi_j) \right). \tag{3.3}$$

Maximizing the log-likelihood function gives the maximum likelihood estimate (MLE) of its arguments π. Ford (1957) suggested a simple iterative algorithm to solve the problem:

$$\pi_i^{(k+1)} = \frac{w_i}{\sum_{j=1}^{n} \frac{w_{ij}+w_{ji}}{\pi_i^{(k)}+\pi_j^{(k)}}}, \tag{3.4}$$

where $\pi_i^{(k)}$ is an estimate of π_i at the step k. In addition, Ford (1957) presented a sufficient condition for this iterative process to converge independently of the starting value π_i^0.

To discuss the convergence properties of the estimation algorithm, I need to introduce a graph representation of the comparison data. Define the comparison graph containing n nodes. Each node in the graph represents an item; a directed weighted edge between nodes i and j represents comparisons of items a_i and a_j. The weight of the edge is w_{ij}, i. e. the number of times a_i beats a_j. The absence of an edge (i, j) means $w_{ij} = 0$. Obviously, the graph must be connected for the existence of a unique solution. For example,

if the comparison graph has two connected components (two subgraphs with no edges-comparisons between them), then there is no way to rank the two corresponding groups of items together.

Ford (1957) formulated the following sufficient condition for his iterative algorithm to converge to a unique finite solution:

> In every possible partition of the objects into two nonempty subsets, some object in the second set has been preferred at least once to some object in the first set.

In terms of graph theory this means that there must exist a directed path between any pair of nodes in the comparison graph. This is a very strong requirement: it means that every item must win and loose at least once.

Hunter (2004) showed that Equation 3.3 was a special case of the minorize-maximization (MM) family of algorithms. They presented a few generalizations of the Bradley-Terry model and presented a general scheme for deriving MM estimation algorithms. They proved that Equation 3.3 converged under a weaker assumption replacing the "preferred" condition with the "compared" condition. That is, according to Hunter (2004), the MM algorithm shown in Equation 3.3 converges to a unique finite solution if and only if the undirected version of the comparison graph is connected. Kaye and Firth (2017) noted that the results of Hunter (2004) had become the standard for fitting Bradley-Terry models.

The Bradley-Terry model in Equation 3.2 can be interpreted in another way. Imagine items a_i, a_j compete doing some task in time, and the winner is the one that finishes the task faster. Let the score characteristics of item a_i be its speed (or rate) π_i. Then, one can model the task completion time as a random variable $T_i \sim \text{Exp}(\pi_i)$, where $\text{Exp}(\pi_i)$ is the exponential distribution with the rate parameter π_i (as in the Gamma model of Stern (1990) with the shape parameter $\alpha = 1$). In this case, the probability of finishing first coincides with Equation 3.2:

$$\Pr(T_i < T_j) = \frac{\pi_i}{\pi_i + \pi_j}. \tag{3.5}$$

Here, T_i, T_j are the latent variables: they cannot be observed in the data. Some authors approached the modeling problem with this parameterization from the Bayesian statistics point of view (Adams, 2005; Guiver and Snelson, 2009). Caron and Doucet (2012) managed to derive an efficient expectation-maximization (EM) algorithm for estimating the model by using a different set of latent variables:

$$Z_{ij} = \sum_{k=1}^{w_{ij}+w_{ji}} \min\left(T_{ki}, T_{kj}\right), \tag{3.6}$$

where index k indicates an independent comparison test and $(w_{ij} + w_{ji})$ is the total number of tests between items a_i, a_j. From Equation 3.6 it follows that $Z_{ij} \sim \text{Gamma}(w_{ij} + w_{ji}, \pi_i + \pi_j)$, where $\text{Gamma}(\alpha, \beta)$ is the gamma distribution with scale parameter α and rate parameter β. Caron and Doucet (2012) used the following prior distribution function to estimate parameters π:

$$p(\pi) = \prod_{i=1}^{n} f_{\text{G}(\alpha,\beta)}(\pi_i) = \left(\frac{\beta^\alpha}{\Gamma(\alpha)}\right)^n \prod_{i=1}^{n} \pi_i^{\alpha-1} e^{-\beta\pi_i}, \tag{3.7}$$

where $f_{\mathrm{G}(\alpha,\beta)}(t)$ is the probability density function of the Gamma distribution. Caron and Doucet (2012) showed that the maximum a posteriori probability (MAP) estimate can be approximated by an iterative process as follows:

$$\pi_i^{(k+1)} = \frac{\alpha - 1 + w_i}{\beta + \sum_{j=1}^{n} \frac{w_{ij}+w_{ji}}{\pi_i^{(k)}+\pi_j^{(k)}}}. \tag{3.8}$$

Interestingly, substituting parameters $\alpha = 1$ and $\beta = 0$ in Equation 3.8 makes it identical to Equation 3.4. In practice, parameters $\mu > 1$ and $\beta > 0$ serve as dumping factors in the iterative process and increase robustness of the algorithm.

A few R[21] packages implement the Bradley-Terry model and its extensions. For example, the `BradleyTerry2` package provides a comprehensive set of functions for the model estimation (Turner and Firth, 2012); the `prefmod` package covers a wider variety of pairwise comparison models (Hatzinger and Dittrich, 2012). Kaye and Firth (2017) developed the `BradleyTerryScalable` package as an alternative to `BradleyTerry2` that could handle data in case of an incomplete pairwise comparison graph.

SAMPLING THE SCORE DISTRIBUTION To proceed with further analysis, I sample the scores as proposed by Caron and Doucet (2012). $\forall i, j : 1 \leq i < j \leq n$ such that $w_{ij} + w_{ji} > 0$:

$$Z_{ij}^{(k)} \mid w, \pi^{(k-1)} \sim \text{Gamma}\left(w_{ij} + w_{ji}, \pi_i^{(k-1)} + \pi_j^{(k-1)}\right). \tag{3.9}$$

To simplify further equations and algorithms, assume $Z_{ij}^{(k)} \equiv 0$ for all i, j such that $(i \geq j)$ or $(w_{ij} + w_{ji} = 0)$. Then, for $i : 1, \ldots n$:

$$\pi_i^{(k)} \mid w, Z^{(k)} \sim \text{Gamma}\left(\alpha + w_i, \beta + \sum_{j=1}^{n} Z_{ij}^{(k)} + \sum_{j=1}^{n} Z_{ji}^{(k)}\right). \tag{3.10}$$

3.2 MODELING WITH SIMULATED DATA

Acknowledging the sparsity of the comparison data in my experiment, I have decided to follow the Bayesian interpretation of the Bradley-Terry model as proposed by Caron and Doucet (2012). I use the `BradleyTerryScalable` package developed by Kaye and Firth (2017), because they have shown that their implementation have successfully estimated the scores for the simulated data set containing one thousand items and a relatively sparse comparison matrix.

The prior assumption of the model is that the item scores are independent identically distributed random variables drawn from the Gamma(α, β) distribution (Equation 3.7). Shape α and rate β parameters of the prior distribution are to be determined purely from the domain-specific knowledge and logical reasoning and must not depend on the observed data.

The scale of the distribution is not identifiable: the data contains comparison results only, hence the absolute scale of all scores can be set to any value. This is clearly seen in

21 `https://www.r-project.org/`, last accessed on 01.09.2018

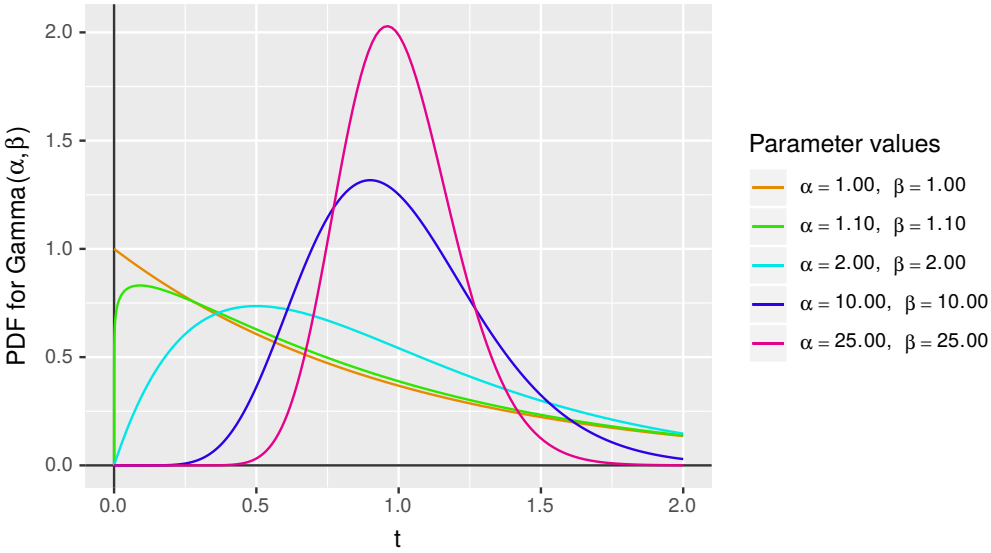


FIGURE 3.1: Probability density function of the Gamma distribution. Shape parameter $\alpha = 1$ is a special case – exponential distribution. Rate parameter β is not important, because the scale of the score is not identifiable.

Equation 3.2: multiplying all scores by the same constant does not change the probability of a comparison outcome. `BradleyTerryScalable` package normalizes the scores to have the unit expected value $\mathrm{E}\pi \equiv 1$. The expected value of the Gamma distribution can be found from its parameters: $\mathrm{E}\pi = \frac{\alpha}{\beta}$. Thus, I set $\beta = \alpha$, which leaves only one parameter to be defined prior to the experiment.

Figure 3.1 presents the probability density function (PDF) of the gamma distribution with different values of parameter α. The gamma distribution is always right-skewed (the mass is concentrated on the left, a long right tail is present). The peak (mode) of the distribution is between 0 and 1 because I set $\alpha = \beta$. In the extreme case, when $\alpha = 1$ the mode is at 0. Note, $\alpha < 1$ is not allowed, because the MAP process in Equation 3.8 would diverge in this case when $w_i = 0$. The choice of parameter α for the prior is determined by the belief about the shape of the score distribution. In the context of the qua-kit exercise, one can choose something in between the following assumptions.

- $\alpha \in [1..2]$: there are a lot of submissions featuring the low order score, but some of the submissions are extremely symmetric.
- $\alpha \in [2..5]$: there are a lot of average-scored submissions, some submissions are totally random (very low score), and a few submissions are extremely symmetric.
- $\alpha > 5$: most of the submissions scores are close to average, there is a small number of totally random and extremely symmetric submissions.

Ideally, one would choose the value for parameter α based solely on one of the assumptions above. However, the model has a caveat: α determines the limit for the performance of the model. To explain this issue, I need to introduce a new term. A natural way to evaluate the performance of the paired comparison model is to test if comparison decisions agree with

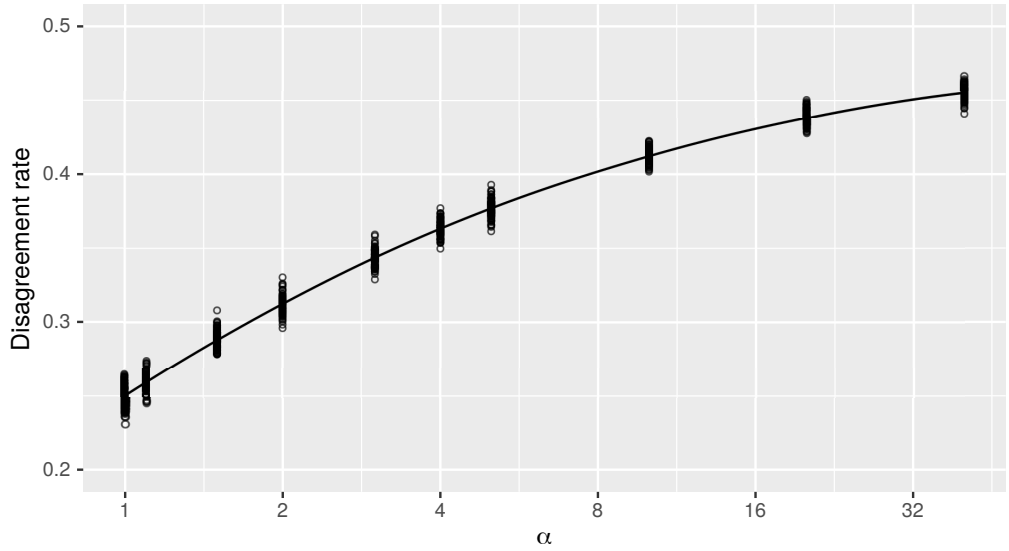


FIGURE 3.2: The base disagreement rate of comparison data. Assume the score distribution $\text{Gamma}(\alpha, \beta)$; then, the base disagreement rate $\text{Err}_0(\alpha)$ is the fraction of comparisons contradicting the items scores. The points represent the simulation results; the solid line is the theoretical expected value of the base disagreement rate.

the assigned scores: if item a_i beats item a_j and $\pi_i > \pi_j$ then the comparison agrees with the model. Define the *disagreement rate* $\text{Err}(\hat{\pi})$ as fraction of comparisons contradicting the scores:

$$\text{Err}(\hat{\pi}) = \frac{1}{m} \sum_{k=1}^{m} \mathbb{1}(T_{ki} < T_{kj}) \mathbb{1}(\hat{\pi}_i < \hat{\pi}_j), \tag{3.11}$$

where m is the total number of comparisons, $\mathbb{1}(A)$ is the indicator function, T_{ki} is the value of a latent variable as in Equation 3.5, and $\hat{\pi}$ is the scores estimate. If the scores are estimated poorly, the disagreement rate is high. However, due to the probabilistic nature of Equation 3.5, knowledge of true scores does not mean zero disagreement rate. This reflects the uncertainty in the decisions; in the context of the qua-kit exercise, individuals may disagree which design is more ordered. Therefore, I define the *base disagreement rate* as the disagreement rate of the true scores $\text{Err}_0 = \text{Err}(\pi)$.

The base disagreement rate depends on the form of the prior distribution. However, the only way I can change the prior distribution within the Bayesian version of the Bradley-Terry model is by changing the value of parameter α. Figure 3.2 presents the computed base disagreement rate using simulated data with varying parameter α. I have simulated the experiments for a few different values of α, 100 simulations per value. In theory, the maximum possible disagreement rate is equal to 0.5; this value would mean the judges do not distinguish the scores at all (the odds are always $1 : 1$). The base disagreement rate in Figure 3.2 is a function of α. It turns out that the smallest possible base disagreement rate is attained with $\alpha = 1$, equals 0.25. The exact form of the function is derived in Appendix B. This reveals a problem of the model: the prior cannot explain a comparison data set if its base disagreement rate is lower than 0.25. Note, the score estimate should

still converge to the true scores given enough comparison data, but the advantage of the Bayesian interpretation would be lost.

SIMULATION To simulate the experiment, I need to define four parameters: n, m, α, β. In most of the tests, I choose a low value of α to minimize the base disagreement rate: $\alpha = \beta = 1/n$. The choice of number of items n and comparisons m is made either to minimize the execution time ($n = 50$, $m \sim n^2$) or to resemble the real qua-kit data ($n \sim 500...1000$, $m \sim 3000...4000$). Furthermore, I create another sample of comparisons to test the performance of the model, $m_{\text{test}} = 500$.

The first step of the test is to simulate the true scores π and the comparisons. I do this by generating a sample of size n from Gamma(α, β) distribution, as per Equation 3.7. Then, I generate $(m + m_{\text{test}})$ random comparisons: $(i_k, j_k, T_{ki}, T_{kj})$. Here, $i_k \neq j_k \in [1, ...n]$ are the indices of items to be compared, T_{ki}, T_{kj} are the hidden random variables drawn from the exponential distributions Exp(π_{k_i}) and Exp(π_{k_j}) respectively. Then, I compute the outcome of each comparison by comparing T_{ki} and T_{kj}. This concludes the simulation part of the test.

To compute the MAP estimate the scores, I use the `BradleyTerryScalable` implementation of the EM process depicted in Equation 3.8. To get more information about the posterior distribution of the scores, I generate a scores sample using Equations 3.9, 3.10 (the sample size 100, the burn-in size 100, and the thinning size 5). Using the generated sample, I create an empirical cumulative distribution function (ECDF) for the posterior distribution of the scores and use it to estimate confidence intervals (CI) for the MAP estimates.

PERFORMANCE When the real scores are not known, the only way to test performance of the model is to to test its prediction power. To do that, I evaluate the disagreement rate. Figure 3.3 shows the results of simulations for different sizes m of the comparison set. In these simulations, I use values $n = 1000$, $\alpha = 1 + 1/n = 1.001$ and perform 100 tests for each value of m. The `BradleyTerryScalable` has failed to converge for small values $m < 2000$ due to the lack of comparison data. For a larger number of comparisons, the disagreement rate converges to the base rate with asymptotic speed approximately $O\left(m^{-0.866}\right)$. Figure 3.3 presents the rates computed in three ways: the base rate using the true scores (red), the rate using the estimated scores on the test data (blue), and the rate using the estimated scores on the training data (green). Clearly, one cannot use the training data to test the performance, because the model overfits the scores. However, if I don't know the true scores, I still can estimate the base disagreement rate by analyzing the difference between the estimated scores on the training and test comparison sets.

Figure 3.4 illustrates the performance of the model if the number of comparisons is large enough. In this example, I use a small number of items ($n = 50$) and a dense comparison graph ($m > n^2$). The black line in the figure is the diagonal $\pi_i = \hat{\pi}_i$; the closer points are to the diagonal, the better the estimate is to the true score. Besides the MAP estimate of the scores, I construct the 95% confidence intervals (CI) using the generated sample of the posterior distribution of the scores. The figure shows clearly that the model can estimate the true scores with a sufficient certainty if it possesses enough comparison data.

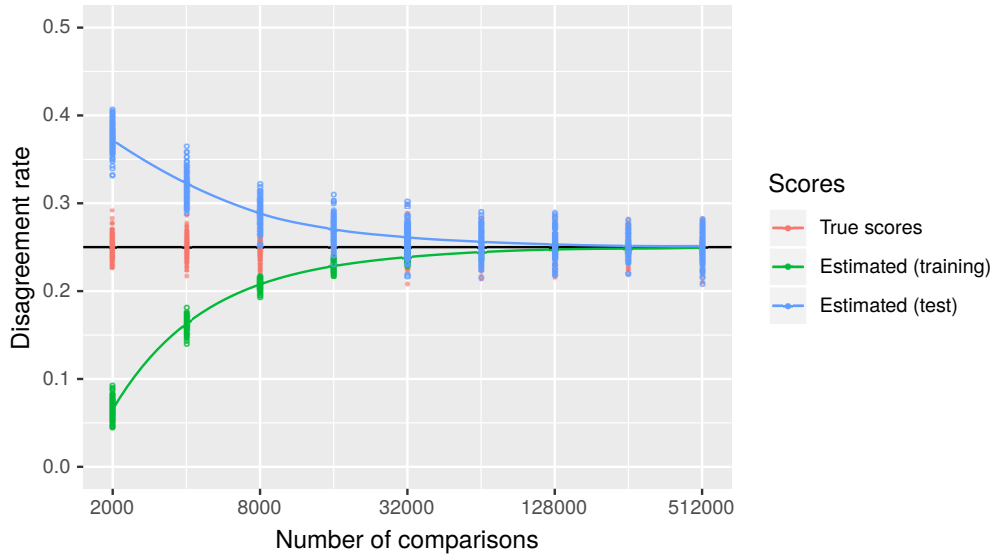


FIGURE 3.3: The disagreement rate of comparison data and the estimated model. $\alpha = 1.001$, $n = 1000$. The black line is the base disagreement rate $E_0 \approx 0.2501$. The disagreement rate of a model (blue line and points) can be approximated by function $\mathrm{Err}(m) \approx 0.2501 + 89.3m^{-0.866}$. The green line and points represent the disagreement rate of the model and its training sample; clearly, the model overfits the training data if the sample size m is small.

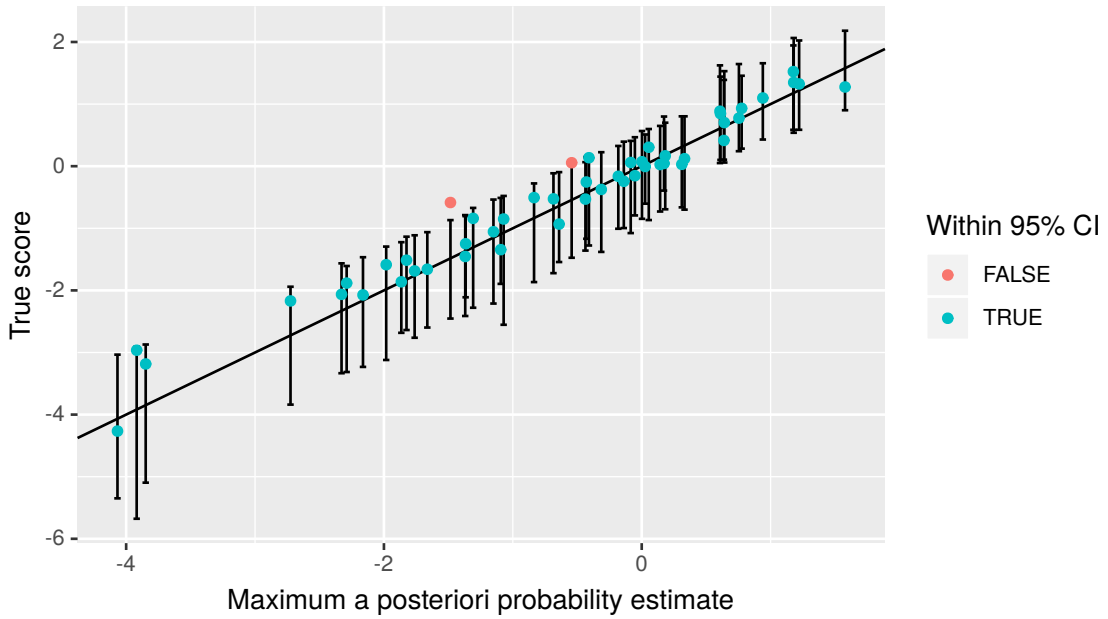


FIGURE 3.4: A scatter plot of true and estimated scores (logarithmic scale). $\alpha = 1.001, n = 50$, $m = 3000$. The black bars represent 95% confidence intervals around the estimated scores. As theory predicts, 48 out of 50 real scores fall into corresponding confidence intervals.

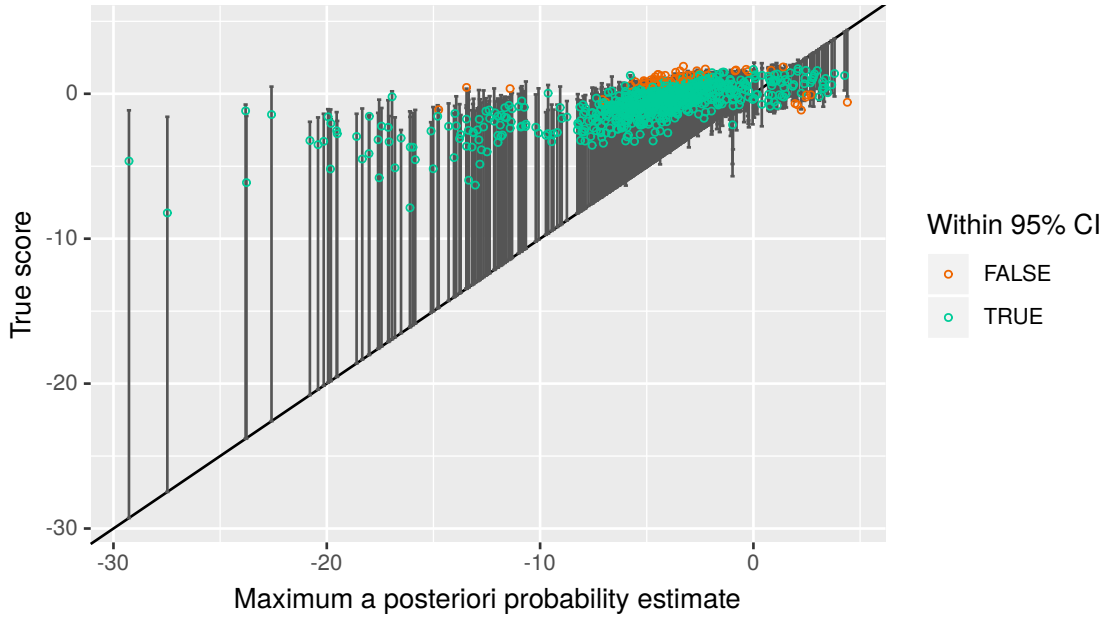


FIGURE 3.5: A scatter plot of true and estimated scores (logarithmic scale). $\alpha = 1.00125$, $n = 800$, $m = 3000$. Due to sparsity of the comparisons (m/n^2 is small), the estimates are biased (far from the diagonal), the confidence intervals are large. 747 out of 800 real scores fall into corresponding confidence intervals.

Figure 3.5 presents the results of the estimation given a data set comparable to the qua-kit exercise ($n = 800$, $m = 3000$). In this case, the comparison graph is sparse and precise estimation of the true scores is not possible. This uncertainty is reflected in the length of the confidence intervals: the items getting too few comparisons have the widest intervals. Note the good quality of the sampler: the CIs still cover approximately 95% of the corresponding scores. The disagreement rate in this example is approximately 0.292, but it varies from test to test in a range from 0.28 to 0.33.

Another interesting observation is that the MAP estimates are skewed when the number of comparisons is small. The algorithm tends to underestimate low scores and overestimate high scores. I suppose, this happens because extreme values of the scores require more comparisons to be estimated correctly. Imagine two items are compared two times. If their scores are almost equal there is a good chance each of them gets one vote. However, if their scores compare as $1 : 10$, one of the items most likely gets both votes. In the latter case, there is no way to distinguish if the odds were $1 : 10$, $1 : 100$, or $1 : 1000$.

The problem of the skewed score estimate can be partially addressed by relying more on the sampled posterior. The MAP estimate is the point estimate of the mode of a distribution. Since I have the posterior sampler, I can estimate the mode using the generated sample instead of the MAP algorithm. Figure 3.6 presents the mode estimate given the same data as in Figure 3.5. The result seems to be much better than the MAP estimate: the true scores are closer to the estimates. The mode estimate is more certain: its total relative CI length is less than that of MAP estimate by 8%. However, the disagreement rate is not significantly smaller (0.286 in this example, and $0.27 - 0.32$ on average). At the same time, it overfits less: in this test, the mode estimate has the disagreement rate 0.167 on the

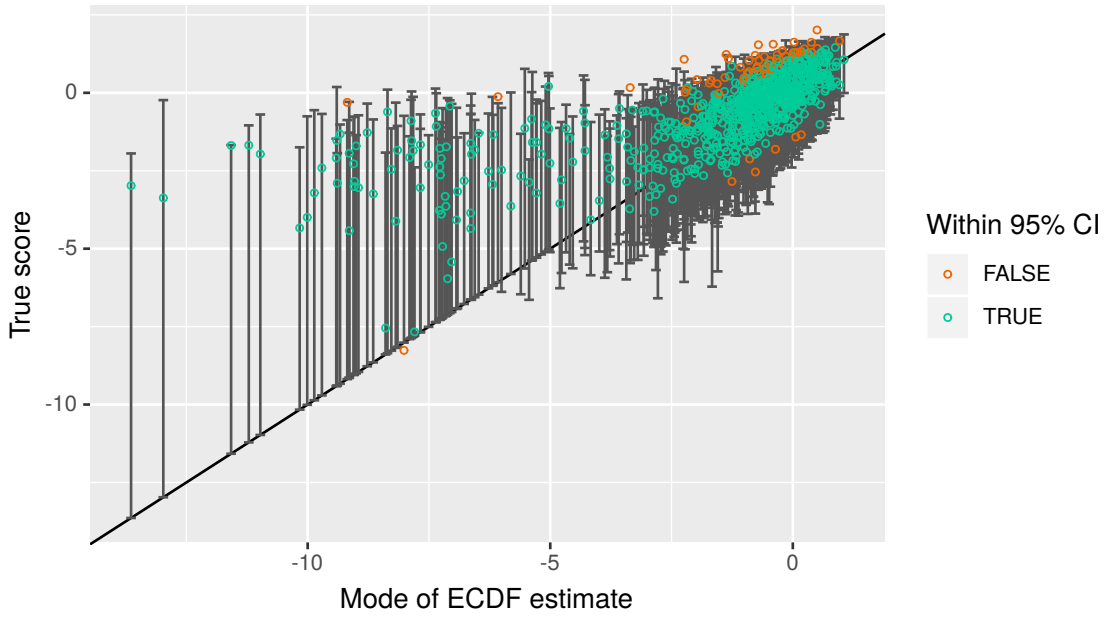


FIGURE 3.6: A scatter plot of true and estimated scores (logarithmic scale). $\alpha = 1.00125$, $n = 800$, $m = 3000$. In this graph, I estimate the modes of the posterior distributions using the generated sample. This process takes a longer time, but reduces the bias if the comparison data is sparse (m/n^2 is small). 750 out of 800 real scores fall into corresponding confidence intervals.

training set, whereas the MAP estimate has the rate 0.148. Asymptotically, the estimates coincide if m/n^2 is not small.

To conclude, the model is capable of estimating the scores and predicting the comparison outcomes if the number of comparisons in the training set is sufficient (i. e. if the comparison graph is connected). It is clear that the scale of the qua-kit experiment does not allow the model disagreement rate reach the base disagreement rate, yet it allows for much better performance than the random guess. I can use the sampling technique and the mode of distribution estimate to slightly improve the results. However, the main problem is the limitation of the model: it cannot explain the data if the base disagreement rate is smaller than 25%. The MAP estimate still *may* converge, but the theoretical results regarding the posterior distribution of the scores would not apply in this case.

IMPROVING THE BASE DISAGREEMENT RATE The limitation of the base disagreement rate being not smaller than 25% decreases the performance of the Bradley-Terry model if a given comparison data set does not have as much uncertainty. In the following, I propose a small modification to the model to workaround this limitation.

Recall the model interpretation in Equation 3.5. I suggest to modify it to have a steeper probability curve:

$$\Pr(T_i < T_j) = \frac{\pi_i^g}{\pi_i^g + \pi_j^g}. \tag{3.12}$$

Here, g is a constant factor; the greater its value is, the more sensitive the model becomes to the difference between π_i and π_j. If $g = 1$, the modified model is equal to the original

model. An equivalent modification is to keep the original model, but change the prior assumption:

$$p(\pi) = \prod_{i=1}^{n} f_{\mathrm{G}(\alpha,\beta)}(\sqrt[g]{\pi_i}) = \left(\frac{\beta^\alpha}{\Gamma(\alpha)}\right)^n \prod_{i=1}^{n} \pi_i^{\frac{\alpha-1}{g}} e^{-\beta \sqrt[g]{\pi_i}}. \tag{3.13}$$

In this case, the derivation presented by Caron and Doucet (2012) stays almost the same, except the maximization step of the EM algorithm ($\pi^{(k+1)} = \arg\max_\pi Q(\pi, \pi^{(k)})$). The modified function $Q(\pi, \pi^{(k)})$ to be maximized on each step is defined as follows:

$$Q(\pi, \pi^*) = \sum_{i=1}^{n} \left(\left(\frac{\alpha - 1}{g} + w_i \right) \log \pi_i - \beta \sqrt[g]{\pi_i} \right) - \sum_{1 \le i < j \le n} (\pi_i + \pi_j) \frac{w_{ij} + w_{ji}}{\pi_i^* + \pi_j^*}. \tag{3.14}$$

Unfortunately, it is not possible to find the maximum of $Q(\pi, \pi^*)$ in Equation 3.14 in the closed form due to term $\sqrt[g]{\pi_i}$. Thus, I approximate it by taking the Taylor expansion of $\sqrt[g]{\pi_i}$:

$$\sqrt[g]{\pi_i} = \left(1 - \frac{1}{g}\right) \pi_i^{*\frac{1}{g}} + \frac{\pi_i^{*\frac{1}{g}-1}}{g} \pi_i + O\left(\left(\frac{\pi_i - \pi_i^*}{\pi_i^*} \right)^2 \right). \tag{3.15}$$

Substituting 3.15 into 3.14 and solving of $\frac{\partial Q(\pi, \pi^*)}{\partial \pi} = 0$ results in the following approximated formula for the EM algorithm:

$$\pi_i^{(k+1)} = \frac{\alpha - 1 + g w_i}{\beta \pi_i^{(k)\frac{1}{g}-1} + g \sum_{j=1}^{n} \frac{w_{ij} + w_{ji}}{\pi_i^{(k)} + \pi_j^{(k)}}}. \tag{3.16}$$

As expected, substituting $g = 1$ into Equation 3.16 makes it identical to Equation 3.8. Parameter g represents the quality of the data: the larger the value of g is, the smaller the base disagreement rate is. Using a correct g ensures better quality of MAP estimates. However, it is difficult to guess correct g, because there is no way to know the base disagreement rate in advance.

Similar to the EM algorithm, I substitute the series expansion of $\sqrt[g]{\pi}$ to derive an approximated version of the modified sampler. Equation 3.9 does not change at all, and Equation 3.10 is rewritten as follows:

$$\pi_i^{(k)} \mid w, Z^{(k)} \sim \mathrm{Gamma}\left(\frac{\alpha - 1}{g} + w_i + 1, \frac{\beta}{g} \hat{\pi}_i^{\frac{1}{g}-1} + \sum_{j=1}^{n} Z_{ij}^{(k)} + \sum_{j=1}^{n} Z_{ji}^{(k)} \right), \tag{3.17}$$

where $\hat{\pi}$ is the MAP estimate of the score. Note, the approximation introduces a bias to the sampling process. To improve the quality of the result, I normalize the generated sample to have the unit mean.

To test the modified model, I have adapted the `BradleyTerryScalable` package; the source code is available online at `github.com`[22]. The simulation process is almost the same as I described for the original model, except that I compute $\pi = \pi_{\mathrm{orig}}^g$ after drawing π_{orig} from the gamma distribution. The rest of the test process stays the same.

Figure 3.7 presents the results on a low-uncertainty data set. The parameter values are $\alpha = 2$, $n = 50$, $m = 500$, $g = 20$. High value of g drops the disagreement rate from 0.317

22 `https://github.com/achirkin/BradleyTerryScalable`, last accessed on 01.09.2018

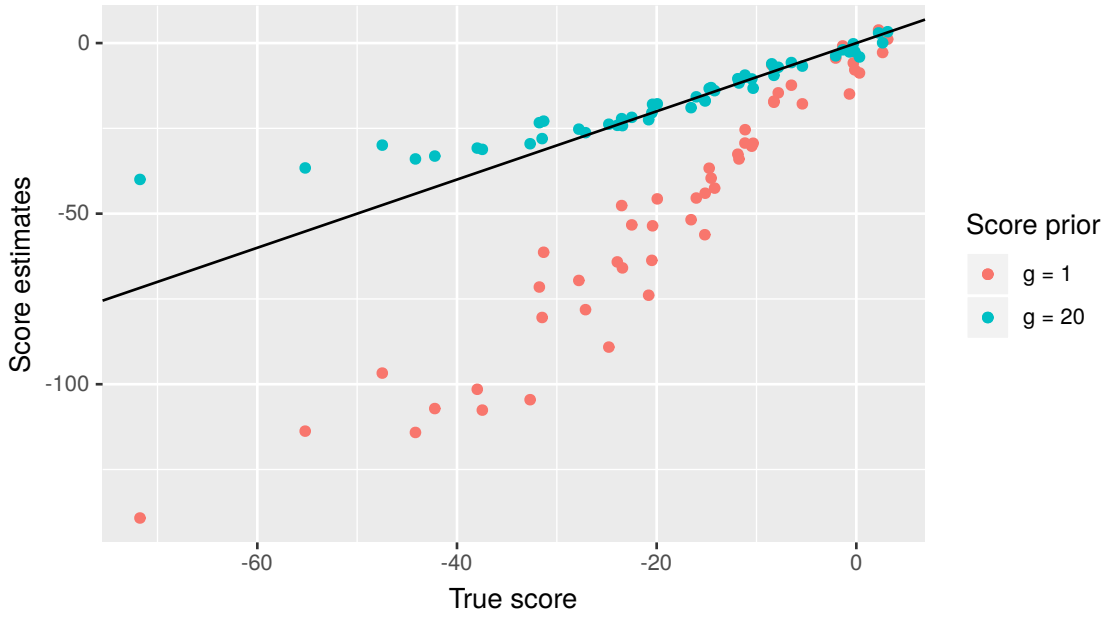


FIGURE 3.7: A scatter plot of true and estimated scores (logarithmic scale). $\alpha = 2$, $n = 50$, $m = 500$, $g = 20$. The modified model ($g = 20$) fits the true scores much better than the original model ($g = 1$), i. e. the former is closer to the diagonal (black) than the latter. The base disagreement rate is 0.026, the disagreement rates of the original ($g = 1$) and modified ($g = 20$) models are 0.060 and 0.034 respectively.

to 0.026 compared to the original model. Visually, it is clear that the estimates of the modified model are much closer to the true scores=1 (note, the estimates of the original model are re-normalized to match the scale of the true scores). The disagreement rate of the modified model is better than the rate of the original model either (0.034 vs 0.060). Interestingly though, the original model still converges to some values well, and both model converge to the same true values if the number of votes is high.

The proposed modification adds one more parameter to the paired comparison model. Higher values of $g > 1$ may compensate for higher values of α in terms of the base disagreement rate. Thus, one can choose α based solely on their belief about the prior distribution shape and select g according to the expected base disagreement rate. The dependency between the base disagreement rate and the model parameters is studied in Appendix B.

A drawback of the modified model is that the samplers proposed by Caron and Doucet (2012) can no longer be used to derive confidence intervals if $g > 1$. The proposed modified sampler (Equation 3.17) is an approximation and may give inaccurate results.

3.3 DATA COLLECTION

The experiment presented in this chapter was not a part of our online courses, although it was similar to the *compare* exercise described in Chapter 1. Participants of the experiment were familiar with qua-kit exercises. Akin to the compare exercise, a participant had to

choose one of two submissions in answer to the experiment question. Unlike the compare exercise, the experiment was not meant to grade the answers of a participant.

EXERCISE I chose the `FC-03x-1` exercise from the "Smart Cities" edX course. This exercise had received the most submissions and had finished by the time I conducted the experiment. The qua-kit database counts 569 unique submissions for this exercise (1296 update records).

PARTICIPANTS To make sure the decisions of participants were based on the same information, it was important that all participants understood the content of the presented drawings and knew the task context. Thus, I looked for participants familiar with qua-kit: my colleagues at the university, current and past students of our online courses who finished the qua-kit exercises. I sent the invitation for the experiment to my colleagues by email, embedded the link with the invitation in our currently conducted edX courses, and used the edX mailing system to reach the students of the finished course.

SETUP The invitation link redirected a participant to the qua-kit experiment page. If the participant did not have an account or was not logged in at the time, a temporary credentials were assigned to them. The experiment consisted in a series of comparisons; the credentials were used to identify a set of comparisons in a single experiment session.

First, the participant saw the briefing page that shortly explained the experiment setting and the goal. Then, the participant proceeded to the comparison page (presented in Figure 3.8). The interface of the page was minimalistic and was adapted to be viewed on mobile devices: it consisted of the two drawings of the designs and a short reminder "Please, click on a design that features more order".

The participant could continue selecting designs as long as they wanted. Every time, the site presented a pair of submissions at random. After twenty comparisons, the participant could see a small note below the drawings stating "Finish here if you are tired of clicking." If the participant proceeded by the link, they were asked to give some feedback by explaining how they had made their decisions. Thus, I encouraged but not required the participants to provide the feedback. At the same time, the participant could ignore the note below the drawings and continue the experiment past the first twenty comparisons.

The content of the experiment pages and the full source code is publicly available at the qua-kit project page[23].

CONTENT To make sure all submissions were in the same conditions, I generated the top-down drawings for them (see Figure 3.8). Thanks to the environment guidelines in the exercise scenario, most of the submissions had similar outlines and orientations of alignment. The qua-kit data base contained 1296 submission update records for the exercise (569 unique submissions). However, I had to manually remove some of them from the experiments due to broken geometry files, wild violation of the design area bounds, or a large number of intentionally overlapped buildings. As a result, 1232 out of 1296 submission drawings participated in the experiment.

23 `https://github.com/achirkin/qua-kit/blob/reflex/apps/hs/qua-server/src/Handler/Mooc/Analysis/VoteOrder.hs`, last accessed on 01.09.2018

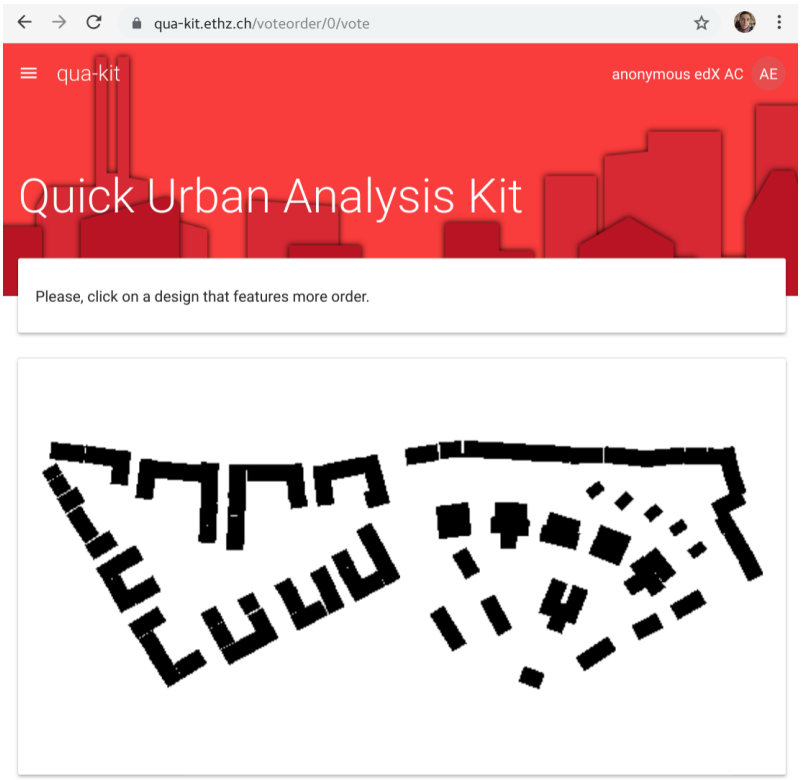


FIGURE 3.8: The web interface used in the experiment. A participant selects a drawing of a submission that, in their opinion, features more order.

RESULT The experiment started on 2018-06-25; at the time of the last dump on 2018-09-07, the database contained 3929 comparison records. Every comparison record contained the voting timestamp and identifiers of a participant (voter) and of two submission records (`better_id`, `worse_id`).

POST-PROCESSING The results of the comparison experiments were not suitable for the analysis as is. The reason is that some submission update records had exactly same drawings. There were two ways this could happen. First, some students submitted their design several times with no changes to geometry (e. g. updated the text description only). Second, some students could fail to do any modifications to their designs due to technical issues but managed to press the "submit" button and saved the initial random template.

To avoid duplicate submissions, I grouped submissions in the exercise based on their similarity. I used drawings (see Figure 3.8) to represent submissions: a drawing was a 256×256 binary-valued matrix (1 if there was a block and 0 if the space was empty). The similarity between two submissions was calculated as the Hamming distance: a number of different pixels at the same positions. If the distance was so small that two drawings were visually indistinguishable, the submissions were considered same. I used the threshold of 10 pixels.

After grouping the submissions, I had to re-numerate them to have a dense item-score representation. Then, I adapted the comparison records accordingly. As a result, some of the comparison records were pointing at the same items (submission groups), so I filtered out such records.

After the post-processing step, the experiment data set contained 943 items and 3919 comparisons. 34 items out of the 943 were very similar (the difference was less than 100 pixels compared to some other items). The comparisons came from 90 distinct voters. The highest number of votes given by a single person was 329, the median number of votes was 29. The median voter decision time was 2.6 seconds.

3.4 MODELING WITH COLLECTED DATA

GUESSING MODEL PARAMETERS As I show in Section 3.2, the ratio of the comparison sample size ($m = 3919$) and the design sample size ($n = 943$) is not favorable for the model ($943^2 \gg 3919$). This means the model disagreement rate ($\mathrm{Err}(\hat{\pi})$) would not approach the base disagreement rate of the data (Err_0), and the choice of the prior (parameters α, g) may affect the performance of the model. Furthermore, parameters α and g define the base disagreement rate (see Appendix B). Thus, a correct choice of the parameters is important for the understanding of the experiment limitations.

To select the appropriate prior, I have performed a series of tests using the obtained data and simulated samples of the same size. By comparing performance of the model trained on the real and simulated sets, I have chosen the best matching parameters. Figure 3.9 illustrates this procedure. The plots represent the disagreement rates for various values of α, with fixed parameters $g = 1$ and $g = 2$. The lines interpolate average disagreement rates across 100 tests for every value of α and g. In these plots, I look for such values of α where the simulation and the real data cross. There is no such point in the left plot; hence, I assume the model does not represent the data very well if $g = 1$. In the right plot, the lines cross at $\alpha \approx 2.6$ (mind the logarithmic scale). Such points exist for higher values of g too, but I try to stay close to the original model by using low value $g = 2$.

Besides the choice of α and g, Figure 3.9 and the same figures for higher values of g suggest two conclusions. First, the disagreement rate of the model trained on the real data barely depends on the prior assumption of α and β: unless α is close to 1, the test disagreement rate is approximately equal to 0.3. Second, the base disagreement rate of the simulated data at the "crossing points" is approximately equal to 0.23 for all tested values of g (2, 3, and 5).

MODELING RESULTS I run the model two times with the same parameter values, using the real and simulated data; the number of items $n = 943$; the number of comparisons $m = 3919$ is split into test and training samples $m_{\text{test}} = 783$, $m_{\text{training}} = 3136$; the modified model parameter $g = 2$; the shape parameter of the prior distribution $\alpha = 2.6$. The results of the tests are as follows:

data	$\mathrm{Err}_{\text{test}}(\hat{\pi})$	$\mathrm{Err}_{\text{training}}(\hat{\pi})$	Err_0
simulated	0.305	0.145	0.23
real	0.303	0.137	

The disagreement rates of the model using the simulated and real data are almost the same. Note, these are the results from a single test (rather than averaged results of multiple tests), and the rates may vary within approximately 10% of their values. The base disagreement

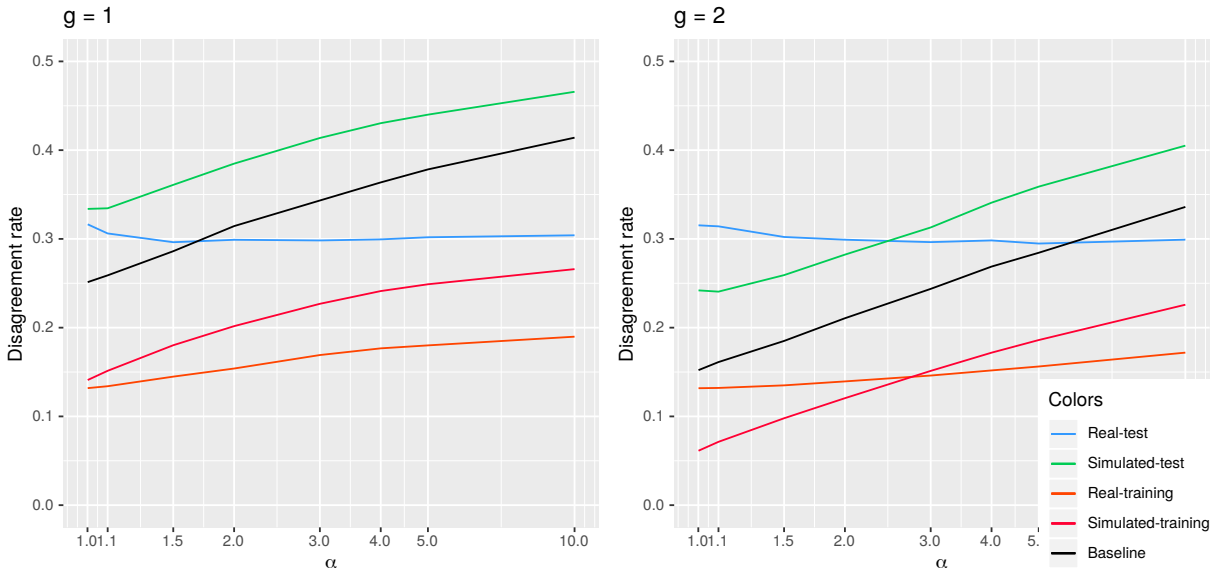


FIGURE 3.9: Guessing parameter values. The disagreement rates are calculated for various combinations of parameters α, g. For every parameter combination, I run 100 tests on a random simulated sample and on a random subsample of the real comparison data. I assume that the simulated sample represents the real data best if the corresponding disagreement rates match.

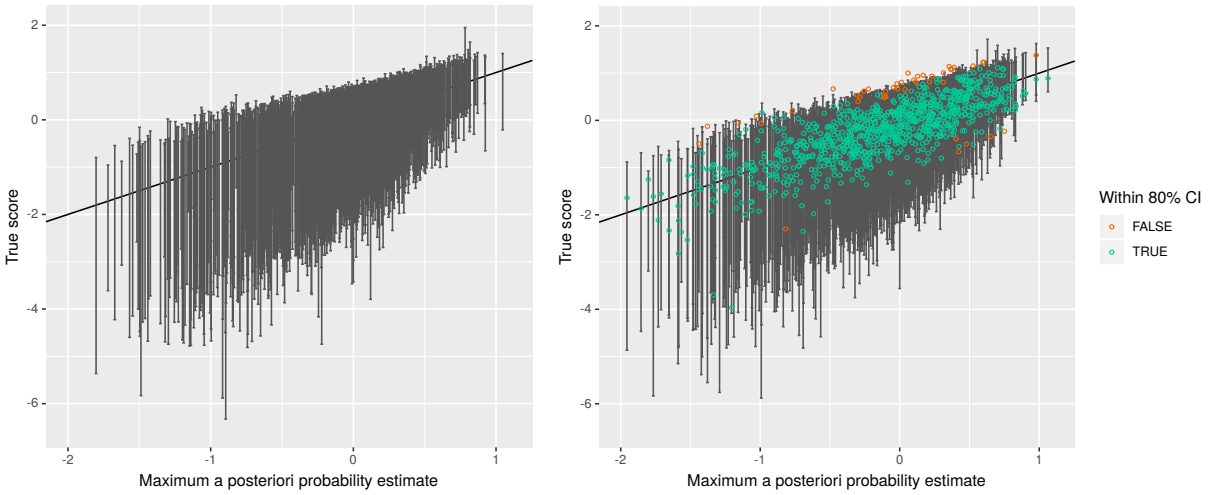
rate of the simulated data is approximately 0.23; the similarity of other metrics suggests that the base disagreement rate of the real data is the same.

Figure 3.10 presents the estimated scores and corresponding confidence intervals. To calculate the confidence intervals, I adjust the estimated scores to be similar to the scores that the original model could produce: I take the inverse transform $\hat{\pi}'_i = \hat{\pi}_i^{1/g}$ and re-normalize the result to have the unit mean. Then, I sample the scores using Equations 3.9, 3.10 as if the MAP estimate was produced by the original model. The resulting confidence intervals are not as accurate as in the original model: 80% intervals in Figure 3.10b cover 93% of the true scores. Nevertheless, this procedure still allows assessing the uncertainty of the estimated scores to a limited extent. In addition, the remarkable similarity between Figures 3.10a and 3.10b contributes to the confidence that the simulated data matches the real data.

Figure 3.11 presents five items that have the highest score and five items that have the lowest score. The difference between the two groups is obvious, though the differences within groups are debatable. A larger list of submissions ordered by their score is available in Appendix C.

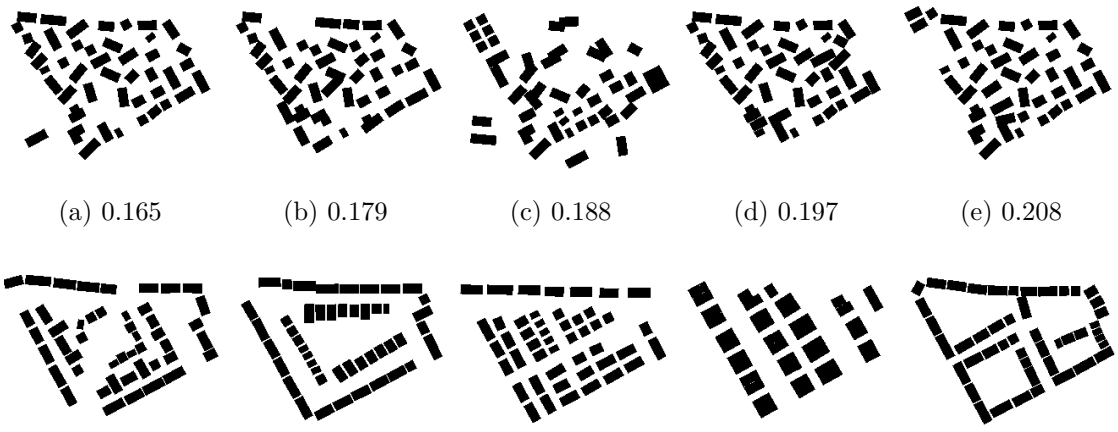
3.5 DISCUSSION

In the presented experiment, the designs are compared with respect to their orderliness. The output of the model is the relative score of the items; I call it the *perceived order score*. Indeed, this score is the processed response of individuals about their perception.



(a) Real data (b) Simulated data

FIGURE 3.10: MAP estimates of the item scores (logarithmic scale). $g = 2, \alpha = 0.26$ Although 80% confidence level is chosen for the simulated data, the confidence intervals cover approximately 93% of true scores, because the sampling process is not accurate for the modified model.



(a) 0.165 (b) 0.179 (c) 0.188 (d) 0.197 (e) 0.208

(f) 2.359 (g) 2.388 (h) 2.514 (i) 2.2521 (j) 2.848

FIGURE 3.11: Submissions with the lowest order score ((a)-(e)) and submissions with the highest order score ((f)-(j)), based on the MAP estimate.

Figure 3.11 clearly shows that the model performs well in this regard, even if the comparison data is not complete. I have asked a large group of participants to help me to develop the model via an online tool. Thus, the perceived order score is, in essence, the crowdsourced order measure. At this point, I have the order measure for the designs that have participated in the experiment. The next step is to combine this result and the symmetry voting algorithm from Chapter 2 to develop a new model that is capable of evaluating the order measure for designs without human feedback.

In this chapter, I analyzed the Bradley-Terry model in detail to make sure I get the best possible performance. It turned out that only the Bayesian interpretation of the model could work with incomplete data, but its prior assumptions did not represent data with low uncertainty well. Thus, I extended the model with an additional parameter to workaround the problem. This result can be used in other studies when it is possible that the uncertainty of data is lower than the original model allows.

By comparing the modified model with the real data and simulations, I estimated the optimal model parameters. This gave me an insight into the structure of the observed data. In particular, I concluded that the base disagreement rate of the observed comparisons was approximately 23%. This number is important for the analysis of the model performance: the prediction error cannot be less than 23% independently of the sample size. It also helps to assess the model performance given the available data by comparing the model disagreement rate 30% to the base rate 23% and the random guess rate 50%.

What does the estimated disagreement rate tell about the obtained data? I would say it reflects the uncertainty in the experiment. On the one hand, the uncertainty is inherent to the experiment because of the way a person subjectively perceives order in a design drawing. On the other hand, it can be controlled by the way the pairs of designs are chosen. The system selects the designs randomly, hence there is a chance to face two designs with almost identical scores. This increases the base disagreement rate, because there is no correct choice in the pair. By reducing the fraction of such pairs, the system can decrease the base disagreement rate of the experiment.

4

SYNTHESIS OF EVALUATION METHODS

This chapter brings together the models explored in the previous chapters. I use the symmetry voting algorithm to extract descriptors from designs and the crowdsourced order measure to represent the order perception. I combine these two in a regression model to create the computational order measure.
I design a convolutional neural network to explain the crowdsourced order score using the symmetry voting grids. Then, I test a set of regression models to combine the scores produced by the neural network for different types of symmetry. The remarkable result is that the final model reduces the disagreement rate of the output order measure compared to the scores estimated in the paired comparison test.

Chapter 2 proposes a computational method to evaluate symmetry. The proposed symmetry voting algorithm can process any input geometry, but it is hard to relate the output of the algorithm to the individual's perception of symmetry or order. Chapter 3 describes the Bradley-Terry model that allows evaluating the perceived order score using the paired comparison experiment. The paired comparison model evaluates the individual's perception, but it needs the human input for the evaluation.

This chapter fills the gap between the models discussed in the previous chapters and concludes the work on the computational order measure. The idea is simple: use the results of the symmetry voting algorithm as the input and the perceived order scores of the comparison model as the output; train a regression model on these input-output data and use it to predict the order scores of new designs.

4.1 MODEL

4.1.1 INPUT DATA PREPROCESSING

The input for the model is the information that is collected for the qua-kit exercise submissions. On the one hand, symmetry in a design is described by the symmetry voting grids for a predefined set of symmetry transform types, as described in Chapter 2. On the other hand, the voting experiment provides the crowdsourced order measure values for a number of submissions, as described in Chapter 3. These data must be prepared in a special way to ensure the best possible performance of the regression model.

PREPARING SYMMETRY VOTING DATA The model described in Chapter 2 provides two types of output: the symmetry voting grid and the list of symmetry pattern descriptors. I find the pattern descriptors being prone to returning false positives, time consuming for

the large-scale analysis, and not well structured to act as input of the regression model. On the contrary, the symmetry voting grid is as simple as a plain grayscale raster picture and takes less time to compute being an intermediate step in the symmetry detection algorithm. In addition, I may be able to extract more information about the orderliness of a design from the voting grid even in the absence of any pronounced symmetry patterns.

As a compromise between the quality of the data and the evaluation speed, I choose the symmetry voting grid size 32×32 cells and 8 pattern types: reflection, translation, and k-fold rotations for $k \in [2...7]$. I perform the symmetry analysis 10 times for each design submission in the qua-kit exercise, rotating the original geometry by $(2\pi j/10)$ radians for $j \in [0...9]$. In effect, the 943 unique design drawings become the 9430 data records for the regression model. This is a common technique in Machine Learning that is used to improve the performance of a model (e. g. neural network).

RE-EVALUATING THE ORDER SCORES Section 3.2 explores the performance limitations of the paired comparisons model with respect to the number of items (submissions) and comparisons. In particular, it shows that the number of user votes I obtained in the comparison experiment is not enough to make the model disagreement rate match the theoretical base disagreement rate of the experiment. The order scores coming from the experiment are used as the ground truth values for the regression, thus their quality is crucial for the regression model. To improve the score estimates, I modify the comparison model configuration.

First, I split the items (submissions) into two groups. I sort the items by the number of related comparison records; take 700 items that are voted most for the comparison experiment, and leave remaining 243 items untouched for the regression model. The outcome is two-fold: every submission participating in the comparison experiment has at least 5 votes now, but 243 submissions in the regression model do not have the order score and cannot be used for the model training. In addition, the split of items implies the split of votes. The number of comparisons used for the paired comparison model reduces from 3919 to 3097. There are 822 comparisons referring to the remaining 243 items, among which 44 comparisons referring to the remaining 243 items only.

Second, I use a cross-validation technique to estimate the model disagreement rate while not sacrificing more data for testing. I randomly split the comparison sample into 5 parts of approximately same size. Then, run the MAP score estimation process 5 times, by combining 4 parts of the sample for training and 1 part for calculating the disagreement rate. Finally, I take the median score values as the output and generate a score sample using the whole dataset. The mean cross-validation disagreement rate is approximately 0.294, and its standard deviation is 0.177. One can interpret this result as having the 95% confidence that the model disagreement rate is within $29.5 \pm 3.5\%$ bounds. This is not significantly better than the original 30.3% discussed in Chapter 3, but it leaves the remaining 243 items and 822 comparisons for the test of the regression model.

Last, I would like to acknowledge the fact that the design order scores are known with the varying confidence in the regression model. Therefore, I use the generated score sample to assign the order measure values to the design submissions. To minimize the impact of the score randomness, I select a few score vectors closest to the MAP estimate (using the

euclidean distance in log-transformed score space). In addition, I mix the selected scores ($\pi^{(i)}$) with the MAP estimate ($\bar{\pi}$) as $\pi^{(i)\prime} = 0.7\bar{\pi} + 0.3\pi^{(i)}$. As a result, the obtained scores are close to the MAP estimate, but their relative dispersions are preserved.

SUMMARY Combining the results of the two models, the input dataset for the regression consists of the following variables:

- voting grids – a multidimensional array containing $32 \times 32 \times 8 \times 9430$ elements – a square matrix for each type of symmetry transform and for each design submission (and its rotations);
- rotation angles – an array containing 9430 elements – rotations applied to each submission (I use it to test if the analysis is rotation-invariant);
- order scores – an array of size 9430 containing 7000 valid numeric scores and 2430 undefined values.

The purpose of the regression model is to construct a function of the voting grids that predicts the order scores while minimizing the disagreement rate. The 7000 data records with valid order score values represent the training set of the model; the remaining 2430 data records are used for tests.

I process the dataset further to simplify the task for a regression algorithm. I divide the voting grid values by their maximum values in the training set (for each symmetry transform type separately). Then, I transform the order score values π as follows:

$$y = \left(\frac{\log \pi - \min(\log \pi)}{\max(\log \pi) - \min(\log \pi)} \right)^2, \tag{4.1}$$

y is the output variable of the regression model; the procedure above ensures the distribution of y spans over interval $[0, 1]$ and has low skewness. One can easily restore the original score values by taking the inverse transform if needed.

4.1.2 ANALYSIS OF VOTING GRIDS

Figures 2.8, 2.9, 2.10 illustrate symmetry voting grids for synthetic and real geometries. In Chapter 2, I explain that peak values in a symmetry voting grid identify the presence of symmetry. Continuing this idea, I suggest that a design submission features more order if some of its voting grids have pronounced peaks; in contrast, a design submission features less order if its voting grids are noisy. Thus, I need an algorithm that could quantify the quality of peaks and the noise level in a voting grid to test this hypothesis.

I start with calculating simple statistics, by interpreting a voting grid as a one-dimensional sample of a random variable (grid value at every cell). This includes the mean, the standard deviation, and the percentiles: 50% (median), 75%, 90%, 100% (maximum). I do not compute smaller percentiles, such as 25% or lower, because they almost always equal to zero. In total, this gives 6 values for a symmetry transform type, or 48 values for a single design.

Intuitively, such combinations as the difference between the maximum and the median should be able to identify presence of a peak in a grid. However, they may fail if the grid contains a few peaks. Therefore, I have tried other ways to extract more information from the grids.

A straightforward approach is the principal component analysis (PCA) treating the 32×32 voting grids as vectors in a 1024-dimensional space. The PCA is a widely-known technique in statistics; in computer vision it is used for face recognition (Kirby and Sirovich, 1990) and other purposes. The PCA allows to decompose the voting grids into linearly independent components – principal vectors. The principal vectors are selected to maximize the variance of the training sample. For example, the first 1024-dimensional principal vector is oriented along direction of the maximum variance. Thus, using the first p principal vectors normally allows to approximate the data sample, reducing its size from $(32 \times 32 \times n)$ to $(p \times n)$ in my case. Unfortunately, I have found experimentally that adding the result of the PCA decomposition to the feature vector does not improve the quality of prediction for any meaningful value of $p < 500$.

Another approach to extracting information from the symmetry voting grids is computer vision feature detection methods (e. g. histogram of gradients (McConnell, 1986) or SIFT (Lowe, 1999)). However, I have found this process computationally expensive and not very effective due to the specifics of the grids being interpreted as images.

The last approach I have tried is using artificial neural networks (NN). Early computational models of NN inspired by the structure of a natural brain were developed in 1940s (McCulloch and Pitts, 1943). A conventional feed-forward artificial NN consists of parametric layers of "neurons" (represented by matrices or tensors) and non-parametric filters. An input signal propagates through the layers by means of matrix multiplication or applying filter operations. Elements of matrices are the *weights* controlling how the signal is transformed. The technique called backpropagation made the supervised learning of NNs possible and efficient (Werbos, 1975). The backpropagation is a fast way to find the weights minimizing the distance between the processed output of the input signal and the true value of the output signal. One type of NNs was found to be particularly efficient for processing images – the convolutional NN (Ciresan et al., 2011). In essence, a convolutional NN layer is a multidimensional discrete convolution kernel with coefficients to be found during the backpropagation. It allows the NN to recognize the continuity of local regions in an image while keeping the complexity and training times of the model at acceptable levels.

To create a NN, I use `mxnet` library (Chen et al., 2015), which employs cuDNN library (Chetlur et al., 2014) for fast NN training on GPU. I take into account two observations to design the structure of my NN. First, the voting grids do not feature complex patterns; the only feature the NN needs to find is the peaks of high symmetry. Second, the size of the sample even after the duplication is rather small for a NN (9430 items); thus, I need to keep the number of parameters low to keep the model from overfitting the training data.

After a series of tests, I have decided to set a single convolutional layer. Figure 4.1 illustrates the structure of the NN. First, a 32×32 symmetry voting grid is passed through eight convolutional filters (convolutional layer); hence, the filters can recognize up to eight

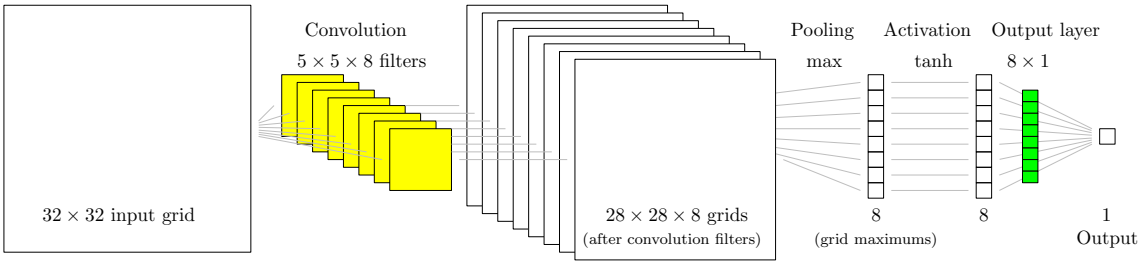


FIGURE 4.1: The proposed NN structure. The NN contains two parametric layers; in total, $5 \times 5 \times 8 + 8 = 208$ weights and 9 bias terms.

different patterns (up to 5×5 cells in size) explaining the output order score. The choice of the number and size of the filters is a trade-off: I need be able to find as many patterns as possible while keeping the number of parameters low. Every filter uses 5×5 weights and a bias term. Thus, the convolutional layer uses 208 parameters in total. The output of the convolutional layer is eight variants of the input grid processed by different filters. I use the (non-parametric) max-pooling layer of the maximum possible size to reduce every grid to a single value. After that, I use the (non-parametric) activation layer that re-normalizes the processed values to be in range $(-1, 1)$. Last, I apply a second parametric layer; it simply calculates a linear combination of the values to produce a single output, so it requires $8 + 1 = 9$ parameters for the weights and the bias. Therefore, the NN consists of 217 parameters to be found by the backpropagation.

Figure 4.2 illustrates the outputs of the convolutional layer. The first row of images in the figure presents the design drawing and the voting grids. The remaining rows represent the voting grids after being processed by the NN filters. The images encode the grid values from -1 to 1 in grayscale: -1 is white, 0 is gray, 1 is black. The majority of images are gray, because the corresponding filters do not match the input. The high-contrast images mean the filters match the input well. The darkest points on the images pass through the pooling layer. Note, the size of dark spots does not affect the result, since the pooling layer simply takes the maximum value of a grid.

The structure of the proposed NN is very simple compared to modern classification NNs. The two reasons for this are the relatively small sample size and the simplicity of discoverable patterns. In essence, the trained NN performs four steps to predict the output variable: apply a set of filters, take maximum values of each output, re-normalize, and compute a linear combination of the results. Compared to the simple statistic approach, this model can take into account the shape of a peak in a symmetry voting grid.

4.1.3 LEVEL 2 MODEL

A single NN takes a symmetry voting grid as an input and predicts the output variable y (as in Equation 4.1). It is possible to combine the eight voting grids into one 3D grid object and train a single NN. However, I have decided against it, because the grids represent different domains (e. g. angle-distance of the reflection symmetry and the xy coordinates of the rotation symmetry). A single NN applied over all eight grids together ends up overfitting the training data and having suboptimal performance on the test data.

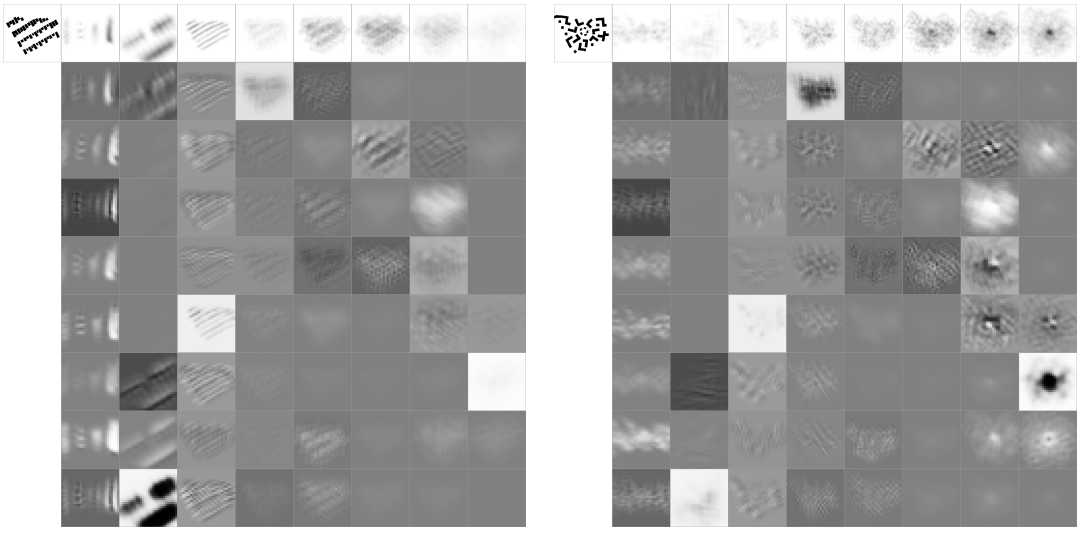


FIGURE 4.2: The outputs of the convolutional NN layer. The columns enumerate symmetry types. The first row is the original data, the remaining rows enumerate filters.

After training the NN models, I possess 8 new variables reflecting the symmetry levels in addition to 48 simple statistics and the artificial rotation-of-design variable. In total, this gives 57 input variables and one output variable for another regression model. I call this the *level 2 regression model* to distinguish it from the NN regression models. The level 2 model does not need to cope with image-like data (voting grids), thus I try various general-purpose regression models as candidates for it.

- Basic *linear regression* using standard R function `lm`. Probably, the simplest model representing the output variable as a linear combination of its inputs. If this model performs well, then the effects of different types of symmetry are additive in terms of their effect on perception of order.

- Linear regression with *lasso* (least absolute shrinkage and selection operator) regularization using the `glmnet` R package (Friedman, Hastie, and Tibshirani, 2010). Compared to the `lm`, this model makes it easier to investigate the importance of individual input variables for predicting the output.

- The *conditional inference regression tree* using the `partykit` R package (Hothorn and Zeileis, 2015). This model is based on recursively applying univariate splits: it selects the most important input variable based on a statistical significance test, then splits the training dataset into two subsets with smaller variance values of the output variable. At some point, the built tree structure partitions the output variable sample into small chunks, each predicting the output for a small range of input values. This model can detect non-linear dependencies between the input and output. In particular, it helps to find out if the rotation angle of a design affects the prediction.

- The *random forest* using the `randomForest` R package (Liaw and Wiener, 2002). The random forest algorithm was proposed by Breiman (2001). The idea is to use an ensemble of decision trees, each tree using a random subset of input variables.

I include this model into my tests, because it is considered to be a state-of-the-art efficient and easy-to-use approach to a wide range of regression and classification problems (Scornet, Biau, and Vert, 2015).

4.1.4 EVALUATION

To evaluate the performance of the level 2 model candidates and the significance of the input variables, I perform a 7-fold cross-validation (CV). I randomly split the 700 unique designs in the training set into 7 chunks of 100 designs. Then I train all regression models using 6 chunks of designs and test them on a remaining chunk. The 7000 training data records are split accordingly. Note, this procedure is similar to the CV setting in Section 4.1.1, except the latter splits the votes instead of designs.

To summarize, I split the input set of 9430 items into three groups: training (6000), CV (1000), and test (2430). The output variable y is known for the training and CV sets and is not known for the test set. The split of items implies the split of comparisons, because every comparison refers to a pair of items. I use the comparisons to calculate the disagreement rate of models and split them into three groups: training, CV+test mixed, and CV+test only. The comparisons in the second group refer to at least one item in the CV or test groups; the comparisons in the third group do not refer to the training group of items. The number of comparisons per group varies depending on the split of items. On average, the training group contains 2278 comparisons; the CV+test mixed group contains 1641 comparisons; the CV+test only group contains 221 comparisons. Note, the third group is a subset of the second group.

Figures 4.3, 4.4 represent the results of cross-validation. The labels on the x axis in both figures identify the level 2 models. The types of models are depicted as follows: `rf` – `randomForest`; `lm` – linear regression; `ctree` – conditional inference tree provided by `partykit`; `glm` – linear regression with lasso provided by `glmnet`. The models are trained with different subsets of input variables:

- `all` – all 57 input variables;
- `onn` – only NN outputs (8 variables);
- `nnr` – NN outputs and design rotation angle (9 variables);
- `oss` – only simple statistics (48 variables);
- 001, 01, 03 – regularization penalties for the `glmnet` model; 0.001, 0.01, and 0.03 respectively.

Figure 4.3 presents the correlations of the output: how well a regression model predicts the order score values estimated in the paired comparison experiment. In addition to the point estimates, I provide the 2σ confidence intervals provided by the CV. Note, Figure 4.3 tells how a regression model fits the results of the paired comparison model rather than the real comparison data. The figure suggests three conclusions. First, the average correlation for all models is between 0.6 and 0.75; these numbers determine the fraction of the score

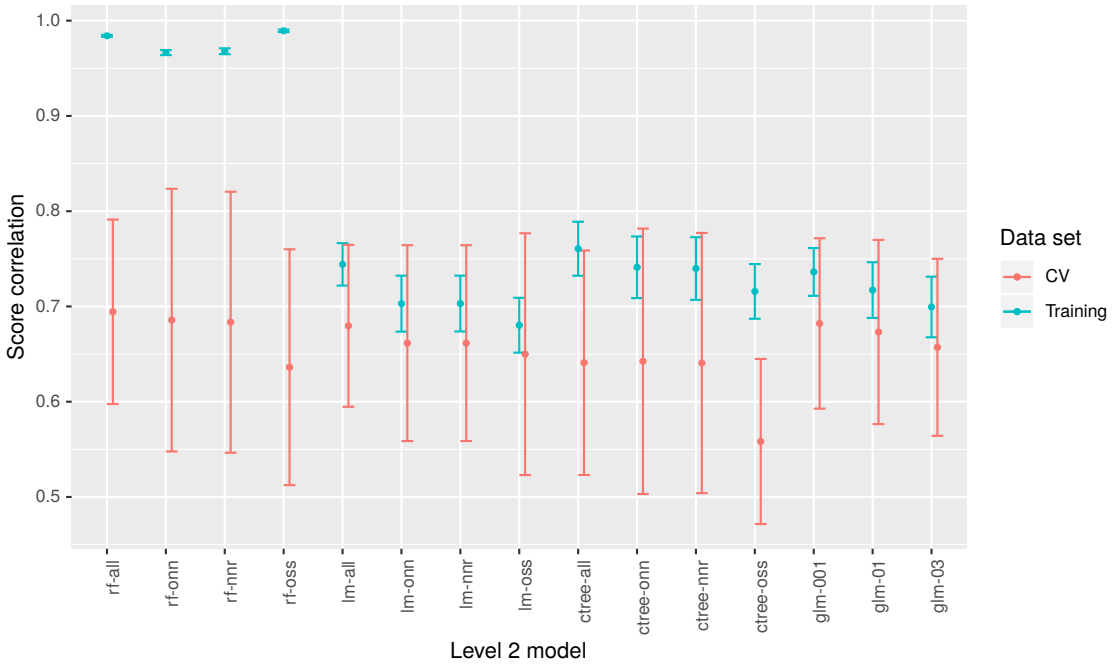


FIGURE 4.3: Correlation between the scores obtained in the paired comparison experiment, and the scores estimated by the regression model. The correlation mean values and 2σ intervals are computed using 7-fold cross-validation (CV) and a set of regression models.

variance that can be explained by the model inputs. Second, the correlation on the CV set almost does not change if the simple statistics and the design rotation are excluded from the input set, though the simple statistics alone are not far worse at explaining the output. This means that the output of NNs contains the same information as the simple statistics plus a bit more. Third, the random forest model severely overfits the training set and yet performs slightly better than others on the CV set.

Figure 4.4 presents the disagreement rates of the regression models. The solid black line at 0.2317 is the base disagreement rate as discussed in Chapter 3. The dashed black lines at 0.149 and 0.295 are the expected values of the training and CV disagreement rates of the paired comparison model. Note, the test only set of comparisons is the smallest; thus the corresponding disagreement rate estimates are not precise and the confidence intervals are wide (green bars). Figure 4.4 partially confirms the findings of Figure 4.3: the simple statistics variables carry slightly less information than the NN outputs, and the random forest model overfits the training set.

In contrast to Figure 4.3, Figure 4.4 illustrates the quality of the regression model with respect to the real comparison data rather than another model's estimates. The linear regression model and its lasso variant with a small regularization parameter seem to be the leaders, though the difference between the alternatives is always within the bounds of uncertainty.

The conditional inference tree model demonstrates slightly worse results than other models. At the same time, it suffers least from the overfitting: the corresponding disagreement rates

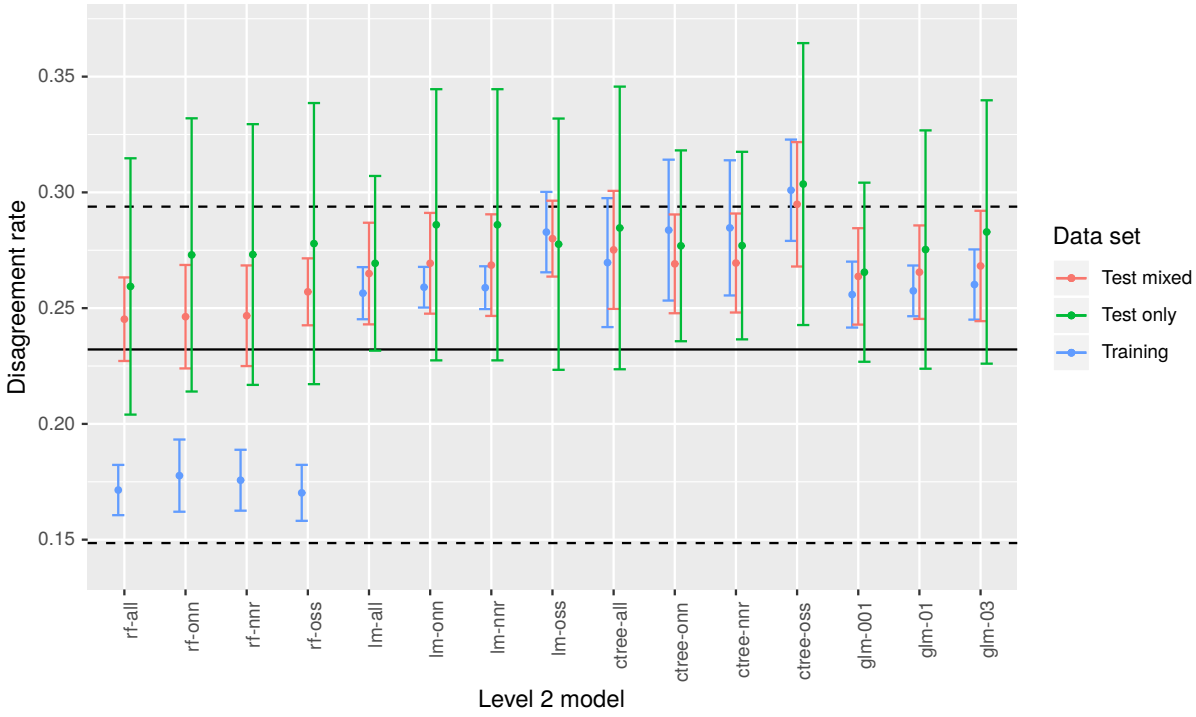


FIGURE 4.4: Disagreement rates of the order scores computed by various regression models and corresponding 2σ confidence intervals. All rates evaluated on the test data are above the base disagreement rate (≈ 0.2317), and most of them are below the disagreement rate of the paired comparison model ($\approx 0.295 \pm 0.035$).

are consistent across the training and test sets. Figure 4.5 illustrates the structure of the `ctree` model. It shows that the score built by the NN from the reflection symmetry voting grids is the most significant factor of the regression (root node `refl_nn`). The second important factor is translation symmetry (`trans_nn`); then goes either 2-fold or 3-fold rotation. Interestingly, only one simple statistics appears in the figure – 75%-percentile of the 4-fold rotation grids. The presented tree is limited to the depth 4; a larger tree contains more variables, including a few simple statistics.

The lasso regularization allows a linear regression model to drop input variables that do not contribute to the prediction significantly. The `glmnet` package uses a regularization penalty parameter λ to control the threshold for discarding the variables. Setting $\lambda = 0.03$ makes the model use six variables only: `refl_nn`, `trans_nn`, `rot2_nn`, `rot3_nn`, `rot4_nn`, `rot5_nn`. Reducing the penalty to $\lambda = 0.01$ adds four simple statistics to the aforementioned NN variables: `trans.0.5`, `rot2.1`, `rot3.0.5`, `rot5.1`, where 0.5 means median and 1 means maximum of the corresponding voting grid values. Surprisingly, these four statistics have a negative impact on the predicted score (negative coefficients in the regression model). With the smaller parameter value $\lambda = 0.001$ the model keeps 23 variables out of 57. This includes all the NN variables, a few percentiles and two standard deviations. Both standard deviations `trans.std` and `rot2.std` have negative coefficients, which confirms the suggestion that a noisy voting grid indicates the low order score.

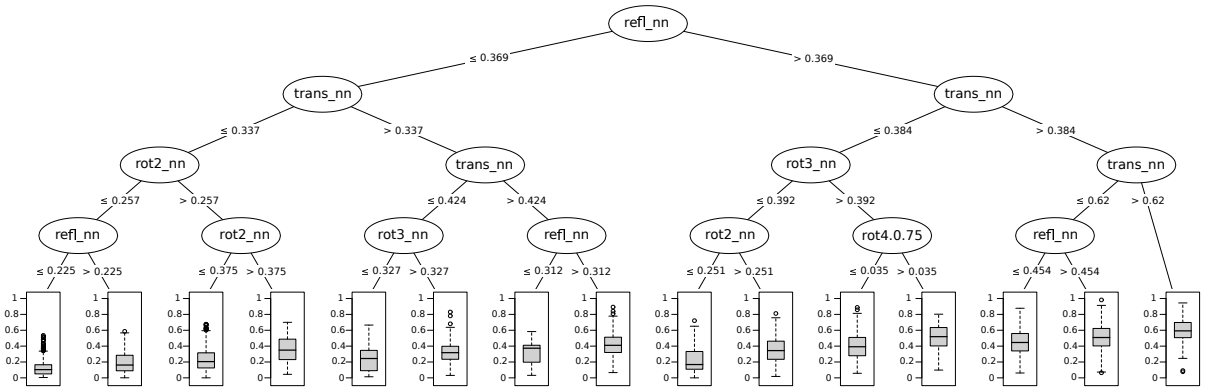


FIGURE 4.5: A conditional inference tree built by the `partykit` package. The nodes of the tree depict splits of the training sample, the leafs represent conditional distributions (subsamples). The tree is constrained by maximum depth 4 to simplify the visualization.

4.2 RESULTS

I choose the `glmnet` model of linear regression with lasso regularization parameter $\lambda = 0.001$. Its disagreement rate on the test+CV only comparison sample is approximately $26.6 \pm 3.9\%$. It is lower than the disagreement rate of the scores estimated in the paired comparison model ($29.5 \pm 3.5\%$), but higher than the hypothetical base disagreement rate of the comparison data (23.2%).

Analysis of various regression models leads to the following conclusions:

1. The symmetry voting grids contain enough information to construct the order measure with the disagreement rate of approximately 26.6%.

2. The NNs provide more information than the set of simple statistics considered in the model. Although performance of the simple statistics alone is comparable.

3. The contribution of different types of symmetry is likely to be additive, because linear models fit the NN data well.

4. Reflection symmetry plays the most important role in the perception of order, followed by translation, 2-fold, and 3-fold symmetry.

The presented regression model is the last piece in the computational model of the order perception. Combining the results of all the chapters together, it is possible to implement a qua-kit service to evaluate the order measure of a design online. Given the pre-trained model, the evaluation of a single submission takes less than a second.

4.3 DISCUSSION

Once a computational model is built and validated, it opens new opportunities for analysis of a target object or phenomenon via interpretation of the model properties. The three models discussed in this thesis provide plenty of such properties.

BASE DISAGREEMENT RATE The Bradley-Terry paired comparison model implies that the input data has an inherent property reflecting the uncertainty of participants' choices; I call it the base disagreement rate. It is the fraction of comparisons contradicting the true item scores as if they were known. By means of comparing the real data and simulations, I have found that the base disagreement rate in the conducted experiment must be close to 23.2%. The base disagreement rate indicates the fundamental limitation of the study: in theory, the participants of the experiment agree with each other in approximately 76.8% of random comparison trials.

COMPARISON MODEL DISAGREEMENT RATE By means of cross-validation, I have found that the disagreement rate of my paired comparison model is approximately $29.5 \pm 3.5\%$. In contrast to the base rate, this number represents the prediction quality of the model given the available training data. Analysis of the model behavior suggests that it could achieve disagreement rate 26.5% if the training sample contained 7000 records and disagreement rate 24% if the training sample contained 56000 records.

REGRESSION MODEL DISAGREEMENT RATE The experiments show that the disagreement rate of the presented regression model is $26.6 \pm 3.9\%$. This result may seem counter-intuitive: I build the regression model upon the scores estimated by the paired comparison model, but the disagreement rate of the former is lower than the disagreement rate of the latter! There are a few possible explanations of this effect:

- The regression disagreement rate has happened to be lower than the comparison disagreement rate by chance; both expected values are within each other's 2σ error span. Although this is possible, the chance must be extremely small; repeated randomized tests show the difference is consistent.

- The regression model is tailored to fit the scores that overfit the given comparison data; thus, one should compare the regression disagreement rate to the comparison disagreement rate on the training data. I rule out this option by means of additional tests; the tests are described in Appendix D.

- The regression model has a small number of parameters and fits the target order measure naturally. This explanation is supported by the fact that the selected regression model features almost identical disagreement rates on training and test comparisons samples (Figure 4.4) and similar correlations on training and CV item samples (Figure 4.3).

To sum up, only the last explanation is plausible and thus I put it forward as one of my conclusions. The question left unanswered is whether the regression model can reduce the disagreement rate further if the paired comparison model provides more accurate scores. The alternative is that 26% might be the limit of what the symmetry voting grids can

allow. Unfortunately, the only way to test it is to perform another comparison study with a much larger comparison sample, which is not possible within the scope of the presented study.

SYMMETRY AND THE ORDER MEASURE If the regression model improves the precision of the order score estimates, then the regression input variables must contain the information explaining the perception of order to a certain degree. Therefore, the symmetry voting grids introduced in Chapter 2 must agree with the perception of order. Moreover, I have observed that presence of clearly pronounced maximums in the symmetry voting grids is often the attribute of a submission with a high perceived order score. This supports the conjecture that the symmetry is a feature of extreme order.

SYMMETRY TYPES AND PERCEPTION OF ORDER The regression analysis confirms that computable symmetry indeed can explain the perception of order. At least, 73.4% of it in the presented experiment. Moreover, the effects of different types of symmetry feature some additivity: the linear models fit the estimated scores well. The significance tests of the `ctree` model suggest that the reflection symmetry has the most impact on the perception of order, which indirectly confirms results of the well-known psychological studies (Goldmeier, 1937). An interesting result is that the next-important variable is the translation symmetry score, followed by various rotation symmetries. However, these results are speculative, because the symmetry voting grids may be highly interdependent. For example, a high reflection symmetry score usually means a high translation symmetry score, due to the abundance of rectangular objects in the exercise scenario.

GENERALIZATION The presented computational order measure is trained on a single qua-kit exercise. Thus, I can guarantee that it agrees with perception of order in the submissions for this exercise only. Nevertheless, the algorithm can compute the measure for any input geometry. The question is whether the computed value reflects the individual's perception of order in that geometry.

The score values observed in the symmetry voting grids are proportional to the total length of edges (walls) involved in a symmetry pattern. Hence, the average magnitude of the symmetry voting grid values is proportional to the edge-length-to-area ratio. One can scale the grid values by this ratio to get more consistent results across different design scenarios. In practice, however, the distribution of geometry is very different in different scenarios (see Figure 4.6). This, in turn, makes the voting grids look very different and raises concerns whether the pre-trained NN can still identify the signs of symmetry.

Another topic worth investigation is how adding an object to a design affects its order measure. The presented qua-kit exercise (`EdX FC-03x-1`) does not allow users to add or delete objects. Thus, the comparison experiment cannot help to answer this question. Adding an object obviously changes the length-to-area ratio, and the voting algorithm shall be tweaked accordingly. The problem arises when the sizes of two compared submissions vary significantly: every object added to one submission or deleted from another one makes the two submissions less comparable.

The presented comparison experiment setting has an advantage that many of the visual variables stay fixed in the compared submissions. In general, qua-kit exercises can be very

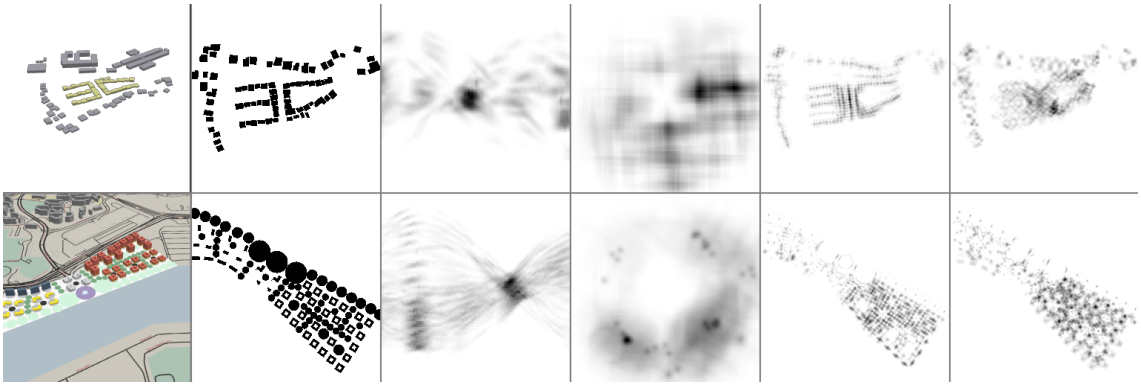


FIGURE 4.6: Examples of symmetry analysis for qua-kit exercises `EdX FC-03x-2` (top) and `EdX FC-04x-2` (bottom).

different. For example, the two scenarios in Figure 4.6 do not seem to be comparable at all. Even though one can compare the degree of order in their footprints ("blocks" column), their answer is not likely to be representative for the actual geometry ("preview" column).

The symmetry voting grids of both submissions in Figure 4.6 feature clearly pronounced peaks. Yet, the shapes of peaks are very different. There is a chance that the trained NN does not detect some of the peaks, because there are no similar peaks in the training sample. So far, the NN has been trained invariant to rotation of a design, but there have been no training samples to make it scale-invariant. A simple statistic, such as the maximum of the grid values, may prove helpful in this case though.

To summarize, there are many ways in which the proposed order measure shall be further investigated. The close-to-linear form of the measure suggests that generalization is possible. It requires two ingredients: more experiments involving different types of case studies and adapting the NNs for new shapes of the symmetry grids.

CONCLUSION

The central result of the presented work is the computational system that measures the degree of order in an urban design drawing. This measure is designed to reflect the perception of order, which is achieved via the supervised learning based on human feedback. On the one hand, I have developed the algorithm to measure the amount of symmetry in two-dimensional geometry. On the other hand, I have constructed the perceived order scores for a set of design submissions using the paired comparison experiment. Combining the two in the regression model have resulted in the development of the computational order measure.

This work was inspired by the idea of the *streetscore* article published by Naik et al. (2014). Authors of this article conducted an online paired comparison experiment to estimate the perceived safety of streetscapes using photos from the Google Streetview. Although the overall implementation scheme was the same, I could not use their methods for numerous reasons. The *streetscore* used a rich set of image feature detection algorithms to construct the descriptors for the regression, whereas I had to implement the symmetry detection algorithm for this purpose. They managed to obtain a large sample of comparisons (208738) and items (4109) and used the Microsoft TrueSkill algorithm to estimate the safety scores. I had a much smaller sample of comparisons (3919), and I wanted to get a better understanding of the participants' decisions. Thus, I used the Bradley-Terry model that allowed a more in-depth analysis of the model and the data.

I see the main purpose of the proposed computational order measure as a tool for analysis of relation between the computable symmetry metrics and the sense of order in urban design. The current study is limited to two-dimensional design drawings, but can potentially be extended to more complex geometry. I have outlined the methodology for the quantitative evaluation of the design feature that was assessed only qualitatively before. The order measure may be used as one of optimization metrics in generative design, thus being a small step towards AI-based design automation.

Analyzing the proposed order measure, I have come to a few conclusions. First, uncertainty is inherent to the paired comparison experiment; it can be controlled by modifying the way designs are selected for the comparison, but it cannot be eliminated due to uncertainty in human decisions. In the presented experiment this uncertainty is reflected as the hypothetical base disagreement rate (23.2%). Second, the perception of order can be explained by a linear combination of the computed symmetry scores to a large extent (disagreement rate $26.6 \pm 3.9\%$ in the presented experiment). Third, the intuitively obvious conjecture that symmetry is a feature of extreme order is confirmed by the observed data.

One of the stepping stones in my research was the development of the symmetry detection algorithm. I found that the known algorithms were not suitable for the two-dimensional geometry of the qua-kit exercises. Therefore, I adapted the symmetry voting stage of the

algorithm proposed by Loy and Eklundh (2006) to recognize symmetry in much greater detail using edges as spatially extended features.

Another research contribution of the presented work is the extension to the Bayesian interpretation of the Bradley-Terry paired comparison model (Caron and Doucet, 2012). I found that the original model could not represent the data with the low disagreement rate (less than 25%), due to its assumption of the prior distribution. I modified the model and derived the algorithm to estimate its (approximate) scores.

On the practical side, a major contribution of this work is the development of qua-kit. Qua-kit was developed in two years; the result is the online platform for urban design, analysis, and discussion. Qua-kit serves as an educational and data collection tool in the online courses at edX. Members of our team used it for public experiments, and other research groups show interest in using it in their research. The source code of the qua-kit project is published online under an open-source license.

An important product of the work related to qua-kit is the qua-kit database. The database contains all exercise submissions, corresponding geometry, descriptions, comments, and voting results. In addition, it contains submission update records – the information that allows recovering design process step-by-step, from the initial state to the submission. The anonymized database will be published online together with the thesis. I believe it can greatly help further research in generative design and reinforcement learning in design.

The presented work can be naturally continued in three research directions. First, the computational order measure shall be generalized to a broader range of design scenarios. I would perform a few more comparison experiments on different case studies and try to develop a revised order measure that would agree with the perception data in all studies. Then, I would explore how the measure interpolates between different studies. Second, one could investigate the decisions of judges in the comparison experiment: whether all participants prefer the same symmetry types, whether the participant's educational background affects the decisions, or whether there is a trainable skill responsible for the quality of decisions. The third research direction is applying the results of this work. A refined symmetry pattern detection algorithm can be used as an optimization constraint to drive the design optimization away from states of very low order. The order measure can be used as one of the optimization targets in generative design process or by a human designer.

A

DERIVING SYMMETRY VOTING FORMULAE

A.1 BILATERAL REFLECTION SYMMETRY

Bilateral reflection symmetry is described by a reflection axis. Therefore, the voting space in this case is a two-dimensional parametric space of lines. I define it using the Hesse normal form as a distance from the origin to a line and an angle to its normal. The angle is restricted to a range $[0, \pi)$, and the distance is restricted by the geometry size.

Assuming a potential axis vector r, the phase score of the reflection symmetry can be defined as follows:

$$\Phi_{\mathtt{b},ab}(r) = \max\left(0, r_a \cdot r_b - 2(r_a \cdot r)(r_b \cdot r)\right) \tag{A.1}$$

By putting a threshold on the phase score $\Phi_{\mathtt{b},a,b}(r) > 0.95$, I narrow down the range of θ to be voted for the wall pair. Thus, I run the voting procedure for several values of orientation vector r corresponding to possible values of θ.

Next, define l as a distance vector from the origin to the symmetry axis ($l \perp r$, see Figure A.1). Then, $(\theta, ||l||)$ are the two parameters of the voting space. Reflection with respect to the defined symmetry axis can be described by a transformation matrix:

$$B = \begin{pmatrix} -\cos 2\theta & -\sin 2\theta \\ -\sin 2\theta & \cos 2\theta \end{pmatrix}. \tag{A.2}$$

To proceed further, I use the following well-known properties of reflection matrices:

$$B = B^{-1}, \qquad |B| = -1. \tag{A.3}$$

The set of parameters values that define symmetric points on a pair of walls is described by the following equation:

$$c_b + td_b - l = B(c_a - sd_a - l) \tag{A.4}$$

Note the minus sign in front of sd_a: I reverse the vector to account for the strict counter-clockwise orientation of the walls. Matrix B depends only on angle θ; an unknown length of vector l depends on parameters s, t for a given value of θ. Therefore, by substituting values -1 and 1, I find the range of possible values $||l||$:

$$\begin{aligned} (1 - B)l &= c_b + td_b - B(c_a - sd_a) \\ (r^\perp - r^\perp B)l &= r^\perp(c_b + td_b) - r^\perp B(c_a - sd_a) \\ (r^\perp + r^\perp)l &= r^\perp(c_b + td_b) + r^\perp(c_a - sd_a) \\ r^\perp \cdot l &= 0.5r^\perp \cdot (c_b + td_b + c_a - sd_a) \end{aligned}$$

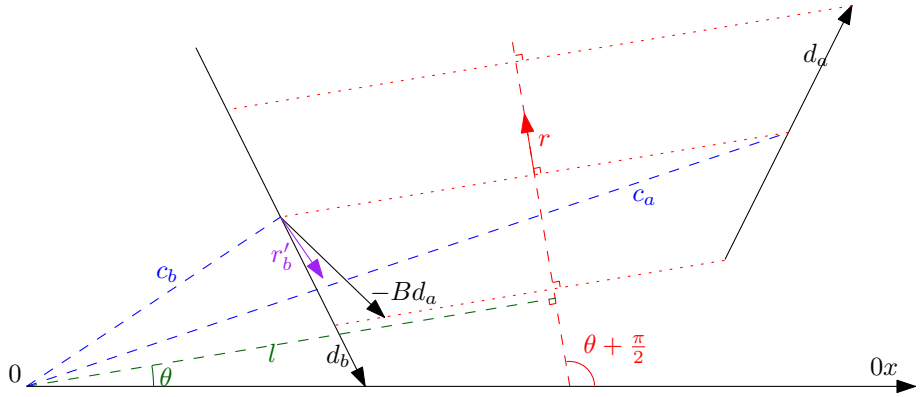


FIGURE A.1: Bilateral symmetry.

$$0.5(r^{\perp} \cdot (c_a + c_b)) - \Delta \leq ||l|| \leq 0.5(r^{\perp} \cdot (c_a + c_b)) + \Delta, \\ \text{where } \Delta = 0.5 \max(|r^{\perp} \cdot d_a|, |r^{\perp} \cdot d_b|). \tag{A.5}$$

In Equation A.5, $r^{\perp}$ is a unit vector orthogonal to r: $r^{\perp} = \frac{l}{||l||}$. Equation A.5 helps to reduce the algorithm execution time by narrowing down range of the voting grid traversal.

Define a corrected orientation vector for one of the walls:

$$r'_b = \frac{r_b - Br_a}{||r_b - Br_a||}. \tag{A.6}$$

Note, $r'_b \cdot r_b = -r'_b \cdot Br_a$ by construction due to Equation A.3. To find the length score of the wall pair, I find the possible range of solutions for Equation A.4. Figure A.1 illustrates that the range of solutions is the measure of the symmetry pattern – the total length of the walls being parts of the pattern. To compute it, I solve the system of equations A.4 under assumption that the walls are (almost) parallel:

$$t(r'_b \cdot d_b) = -s(r'_b \cdot Bd_a) + r'_b \cdot (Bc_a - c_b + (1 - B)l), \\ t(r'_b \times d_b) = -s(r'_b \times Bd_a) + r'_b \times (Bc_a - c_b + (1 - B)l). \tag{A.7}$$

Parameters t and s must stay within $[-1, 1]$. Then, the dot product equality can be turned into the system of inequalities for allowed parameter values:

$$\begin{cases} -r'_b \cdot d_b \leq t(r'_b \cdot d_b) \leq r'_b \cdot d_b, \\ r'_b \cdot (Bc_a - c_b + (1 - B)l) + r'_b \cdot Bd_a \leq t(r'_b \cdot d_b) \\ \qquad\qquad \leq r'_b \cdot (Bc_a - c_b + (1 - B)l) - r'_b \cdot Bd_a. \end{cases} \tag{A.8}$$

The length of the allowed interval derived from this system can be used as a symmetry score component. The cross product equality implies that $r'_b \times (Bc_a - c_b + (1 - B)l) \approx 0$. Thus, $|r'_b \times (Bc_a - c_b + (1 - B)l)|$ can be used as an additional damping factor to control the distance between the walls. Combining these together, the length score can be defined as follows:

$$L_{\mathfrak{b},ab}(\theta, ||l||) = \min(r'_b \cdot d_b, -r'_b \cdot Bd_a + r'_b \cdot (Bc_a - c_b + (1 - B)l)) \\ + \min(r'_b \cdot d_b, -r'_b \cdot Bd_a - r'_b \cdot (Bc_a - c_b + (1 - B)l)) \\ - |r'_b \times (Bc_a - c_b + (1 - B)l)|,$$

which simplifies to

$$\begin{aligned} L_{\mathtt{b},ab}(\theta, ||l||) = \min\,[&2(r'_b \cdot d_b) \\ &, -2(r'_b \cdot Bd_a) \\ &, r'_b \cdot (d_b - Bd_a) - |r'_b \cdot (Bc_a - c_b + (1-B)l)| \\ &] - |r'_b \times (Bc_a - c_b + (1-B)l)|. \end{aligned} \tag{A.9}$$

There are a few notes to one has to keep in mind implementing the algorithm:

- I use a threshold on the phase score ($\Phi_{\mathtt{b},a,b}(r) > 0.95$), hence $r'_b \cdot d_b$ and $-r'_b \cdot Bd_a$ are always positive and Equation A.9 makes sense;
- $L_{\mathtt{b},ab}(\theta, ||l||)$ in Equation A.9 can be negative; I use a threshold $L_{\mathtt{b},ab}(\theta, ||l||) > 0$ to avoid negative scores;
- $L_{\mathtt{b},ab}(\theta, ||l||)$ is symmetric w.r.t. to changing the roles of arguments a and b; this arises from the fact that $r'_a = -Br'_b$ via multiplying both sides of scalar and vector products by $-B$ in Equation A.9.

A.2 ROTATION SYMMETRY

Rotation symmetry is described by a rotation center and a fold degree. The presented algorithm performs voting for rotation centers on a grid for a specified fold degree $k \in [2..)$. Therefore, the voting space in this case is the same two-dimensional plane as the original geometry space.

For a fixed fold degree k, define the rotation angle as $\phi = \frac{2\pi}{k}$. Then, the rotation matrix that transforms a point of one wall onto a point of another wall is defined as follows:

$$R = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}. \tag{A.10}$$

If some points on the walls are part of a symmetry pattern, then they must satisfy the following equality:

$$c_b + td_b - c_{ab}(s,t) = R(c_a + sd_a - c_{ab}(s,t)), \tag{A.11}$$

where $c_{ab}(s,t)$ is a rotation center corresponding to the pair of points on the walls a and b. The rotation center can be found for any two walls and values of parameters s and t:

$$c_{ab}(s,t) = (I-R)^{-1}(c_b + td_b - c_a - sd_a). \tag{A.12}$$

Note, matrix $(I-R)$ is invertible for any $k > 1$. Thus, for any two walls there exist a quadrilateral (possibly degenerate) of possible rotation centres corresponding to $s, t \in [-1, 1]$. By substituting values -1 and 1 into Equation A.12, I find the patch of the voting grid to be updated. Though, to avoid numerical problems and make the voting values on the grid smooth, I extend this patch by $\max(||d_a||, ||d_b||)$ in each direction.

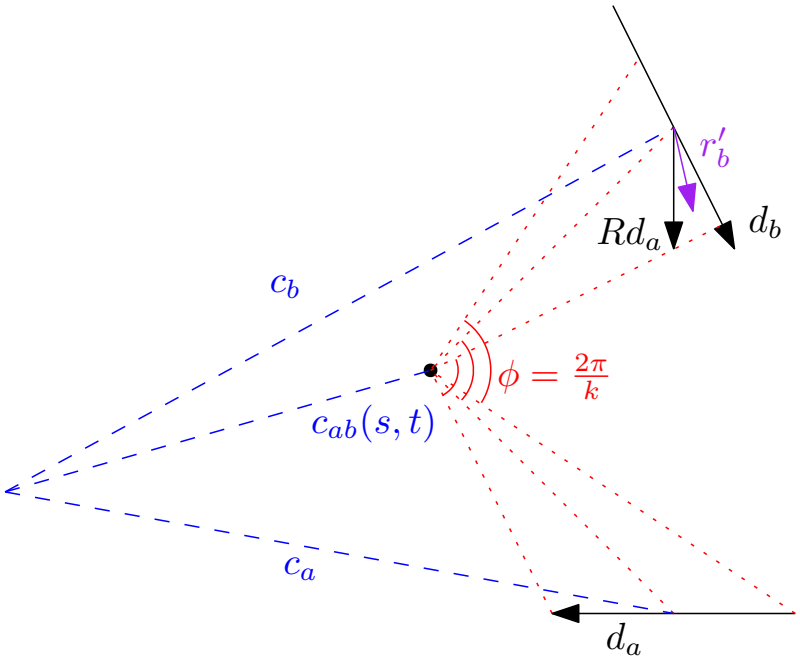


FIGURE A.2: Rotation symmetry.

Assume α is an angle between vectors r_a and r_b. To define the phase score, I use the cosine similarity to favor pairs of walls such that α is close to ϕ:

$$\begin{aligned} \cos\alpha &= (r_a \cdot r_b), \\ \cos\left(\alpha - \frac{2\pi}{k}\right) &= (r_a \cdot r_b)\cos\frac{2\pi}{k} + (r_a \times r_b)\sin\frac{2\pi}{k}. \end{aligned}$$

The latter expression can be expressed in terms of rotation matrix R:

$$\Phi_{\mathtt{r}k,ab} = (r_a \cdot r_b)R_{1,1} + (r_a \times r_b)R_{2,1}. \tag{A.13}$$

$\Phi_{\mathtt{r}k,ab}$ is the phase score of the k-fold symmetry for walls a, b. Note, that $\Phi_{\mathtt{r}k,ab}$ does not depend on the location of the rotation center.

Define a corrected orientation vector for one of the walls:

$$r'_b = \frac{r_b + Rr_a}{||r_b + Rr_a||}. \tag{A.14}$$

To find the length score of the wall pair, I find the possible range of solutions for Equation A.11. Figure A.2 illustrates that the range of solutions is the measure of the symmetry pattern – the total length of the walls being parts of the pattern. To compute it, I solve the system of equations A.11 under assumption $r_b \approx Rr_a$:

$$\begin{aligned} t(r'_b \cdot d_b) &= s(r'_b \cdot Rd_a) + r'_b \cdot (Rc_a - c_b + (1-R)c_{ab}), \\ t(r'_b \times d_b) &= s(r'_b \times Rd_a) + r'_b \times (Rc_a - c_b + (1-R)c_{ab}). \end{aligned} \tag{A.15}$$

Parameters t and s must stay within $[-1, 1]$. Then, the dot product equality can be turned into the system of inequalities for allowed parameter values:

$$\begin{cases} -r'_b \cdot d_b & \leq t(r'_b \cdot d_b) \leq r'_b \cdot d_b, \\ r'_b \cdot (Rc_a - c_b + (1-R)c_{ab}) - r'_b \cdot Rd_a & \leq t(r'_b \cdot d_b) \\ & \leq r'_b \cdot (Rc_a - c_b + (1-R)c_{ab}) + r'_b \cdot Rd_a. \end{cases} \tag{A.16}$$

The length of the allowed interval derived from this system can be used as a component of the symmetry score. The cross product equality implies that $r'_b \times (Rc_a - c_b + (1-R)c_{ab}) \approx 0$. Thus, $|r'_b \times (Rc_a - c_b + (1-R)c_{ab})|$ can be used as an additional damping factor to control the distance between the walls. Combining these together, the length score can be defined as follows:

$$\begin{aligned} L_{\mathtt{r}k,ab}(c) = &\min(r'_b \cdot d_b, r'_b \cdot Rd_a + r'_b \cdot (Rc_a - c_b + (1-R)c)) \\ &+ \min(r'_b \cdot d_b, r'_b \cdot Rd_a - r'_b \cdot (Rc_a - c_b + (1-R)c)) \\ &- |r'_b \times (Rc_a - c_b + (1-R)c)|, \end{aligned}$$

which simplifies to

$$\begin{aligned} L_{\mathtt{r}k,ab}(c) = \min [&2(r'_b \cdot d_b) \\ &, 2(r'_b \cdot Rd_a) \\ &, r'_b \cdot (d_b + Rd_a) - |r'_b \cdot (Rc_a - c_b + (1-R)c)| \\ &] - |r'_b \times (Rc_a - c_b + (1-R)c)|. \end{aligned} \tag{A.17}$$

There are a few notes to one has to keep in mind implementing the algorithm:

- I use a threshold on the phase score ($\Phi_{\mathtt{r}k,ab} > 0.95$), hence $r'_b \cdot d_b$ and $r'_b \cdot Rd_a$ are always positive and Equation A.17 makes sense;
- $L_{\mathtt{r}k,ab}(c)$ in Equation A.17 can be negative; I use a threshold $L_{\mathtt{r}k,ab}(c) > 0$ to avoid negative scores;
- $L_{\mathtt{r}k,ab}(c)$ is symmetric w.r.t. to changing the roles of arguments a and b if $k = 2$ only. Otherwise, it is greater than zero only if the wall b is counter-clockwise rotation of the wall a.
- For each pair of walls the algorithm updates a small patch on a raster grid, adding a small symmetry vote to corresponding center points. The patch size is determined by getting minimum and maximum of four corner points by substituting -1 and 1 into Equation A.12, and dilating it by $\pm(||d_a|| + ||d_b||)r'^{\perp}_b$.

A.3 TRANSLATION SYMMETRY

Translation symmetry is described by its translation vector. Therefore, the voting space in this case is the two-dimensional plane of possible translation vectors. The plain is further restricted by the maximum allowed size of a shift vector that is proportional to the scale of a symmetry pattern to be discovered.

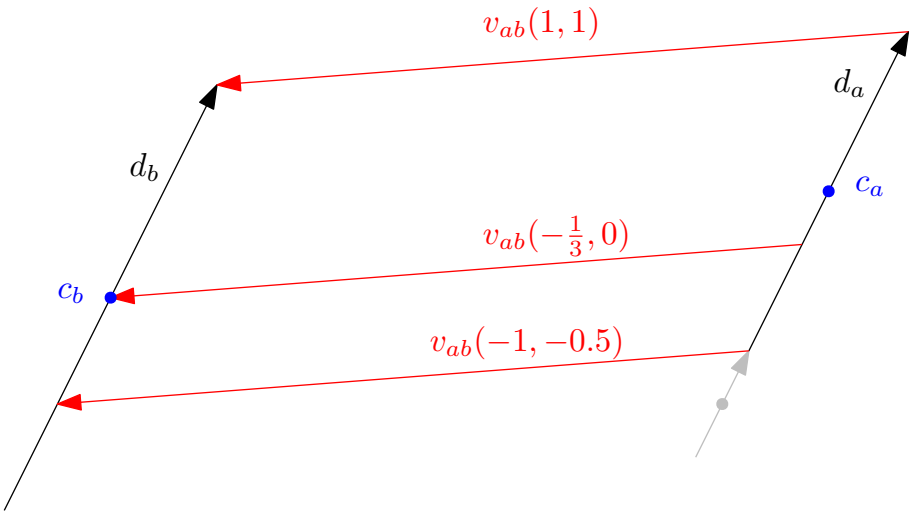


FIGURE A.3: translation symmetry; the wall b partially is the translation of the wall a; the size of an interval of possible origin (or destination) points for vector v serves as the length measure of the symmetry L_{ab} (in this case $L_{ab} = 2d_a = 1.5d_b$).

The translation vector for walls a, b is defined as follows:

$$v_{ab}(s,t) = c_b + td_b - c_a - sd_a. \tag{A.18}$$

This set has two equations and two unknowns. However, translation symmetry allows existence of parallel walls only; hence, I assume that Equation A.18 may have an infinite number of solutions defined by the linear dependence of parameters s and t (see Figure A.3). By substituting values -1 and 1 into the parameters, I find the minimum and maximum possible values of $v_{ab}(s,t)$.

The phase score of the pair is easy to compute as the cosine of the angle between the walls:

$$\Phi_{\mathtt{t},ab} = (r_a \cdot r_b), \tag{A.19}$$

I cut-off clearly non-parallel walls using the threshold $\Phi_{\mathtt{t},ab} > 0.95$ to reduce the runtime.

To define the length score of the wall pair, I find the possible range of solutions for Equation A.18. Figure A.3 illustrates that the range of solutions is the measure of the symmetry pattern – the total length of the walls being parts of the pattern. To compute it, I define the normalized average orientation vector $r = \frac{r_a + r_b}{||r_a + r_b||}$ to solve the system of equations A.18 under assumption that the walls are (almost) parallel:

$$\begin{aligned} t(r \cdot d_b) &= s(r \cdot d_a) + r \cdot (c_a - c_b + v), \\ t(r \times d_b) &= s(r \times d_a) + r \times (c_a - c_b + v). \end{aligned}$$

Parameters t and s must stay within $[-1, 1]$. Then, the dot product equality can be turned into the system of inequalities for allowed parameter values:

$$\begin{cases} -r \cdot d_b \leq t(r \cdot d_b) \leq r \cdot d_b, \\ r \cdot (c_a - c_b + v) - r \cdot d_a \leq t(r \cdot d_b) \leq r \cdot (c_a - c_b + v) + r \cdot d_a. \end{cases}$$

The length of the allowed interval derived from this system can be used as a component of the symmetry score. The cross product equality implies that $r \times (c_a - c_b + v) \approx 0$;

thus $r \times (c_a - c_b + v)$ can be used as an additional damping factor to control the distance between the walls. Combining these together, the length score can be defined as follows:

$$\begin{aligned} L_{\mathtt{t},ab}(v) = &\min(r \cdot d_b, r \cdot d_a + r \cdot (c_a - c_b + v)) \\ &+ \min(r \cdot d_b, r \cdot d_a - r \cdot (c_a - c_b + v)) \\ &- |r \times (c_a - c_b + v)|, \end{aligned}$$

which simplifies to

$$L_{\mathtt{t},ab}(v) = \min\left[2(r \cdot d_b), 2(r \cdot d_a), r \cdot (d_a + d_b) - |r \cdot (c_a - c_b + v)|\right] - |r \times (c_a - c_b + v)|. \tag{A.20}$$

Note, $L_{\mathtt{t},ab}(v)$ can be negative, thus a threshold $L_{\mathtt{t},ab}(v) > 0$ is used.

B

DERIVING BASE DISAGREEMENT RATE FORMULAE

The base disagreement rate does not depend on the number of conducted comparison trials. In other words, it represents the uncertainty of the score itself. The base disagreement rate depends on the parameters of the model only; thus, it can be chosen independently of the data as a part of the prior assumptions about the experiment.

Following the model proposed by Caron and Doucet (2012), the prior distribution of the scores is Gamma(α, β). A single comparison test is a Bernulli trial with probability of the outcome depending on two independent scores (Equation 3.12). Therefore, the base disagreement rate is the expected value of a binary random variable representing the disagreement between the scores and the outcomes:

$$\mathrm{Err}_0 = 2\int_0^{+\infty}\int_0^{+\infty} \Pr(T_1 < T_2 | \pi_1 = x_1, \pi_2 = x_2)\mathbb{1}(x_1 < x_2)\,\mathrm{d}F(x_1)\,\mathrm{d}F(x_2). \tag{B.1}$$

Here, $F(x)$ is the cumulative distribution function of the Gamma(α, β) distribution; T_i is the "task completion time" of an item with score π_i as in Equations 3.5 and 3.12 ($T_i \sim \mathrm{Exp}(\pi_i)$). Substitute the gamma distribution function and Equation 3.5:

$$\mathrm{Err}_0(\alpha, \beta, g) = 2\frac{\beta^{2\alpha}}{\Gamma^2(\alpha)}\int_0^{+\infty}\int_0^{+\infty} \frac{x_1^{g+\alpha-1}x_2^{\alpha-1}\mathrm{e}^{-\beta(x_1+x_2)}}{x_1^g + x_2^g}\mathbb{1}(x_1 < x_2)\,\mathrm{d}x_1\,\mathrm{d}x_2. \tag{B.2}$$

The latter does not depend on scale parameter β:

$$\mathrm{Err}_0(\alpha, g) = \frac{2}{\Gamma^2(\alpha)}\int_0^{+\infty}\int_0^{+\infty} \frac{x_1^{g+\alpha-1}x_2^{\alpha-1}\mathrm{e}^{-x_1-x_2}}{x_1^g + x_2^g}\mathbb{1}(x_1 < x_2)\,\mathrm{d}x_1\,\mathrm{d}x_2. \tag{B.3}$$

Simplify out the indicator function by changing the integration bounds:

$$\mathrm{Err}_0(\alpha, g) = \frac{2}{\Gamma^2(\alpha)}\int_0^{+\infty}\int_0^{x_2} \frac{x_1^{g+\alpha-1}x_2^{\alpha-1}}{x_1^g + x_2^g}\mathrm{e}^{-x_1-x_2}\,\mathrm{d}x_1\,\mathrm{d}x_2. \tag{B.4}$$

Then, transform the integrand into polar coordinates using the following substitution: $x_1 = \rho\sin^2\phi$, $x_2 = \rho\cos^2\phi$. As a result, the integration bounds are independent and the integrand variables are separable:

$$\mathrm{Err}_0(\alpha, g) = \frac{4}{\Gamma^2(\alpha)}\int_0^{+\infty}\int_0^{\frac{\pi}{4}} \frac{\sin^{2g+2\alpha-1}\phi\cos^{2\alpha-1}\phi}{\sin^{2g}\phi + \cos^{2g}\phi}\rho^{2\alpha-1}\mathrm{e}^{-\rho}\,\mathrm{d}\phi\mathrm{d}\rho. \tag{B.5}$$

Note the squared trigonometric substitution: it makes the exponent in the integrand independent of ϕ. Now, it is possible to integrate by ρ:

$$\begin{aligned}\mathrm{Err}_0(\alpha, g) =& 4\frac{\Gamma(2\alpha)}{\Gamma^2(\alpha)}\int_0^{\frac{\pi}{4}} \frac{\sin^{2g+2\alpha-1}\phi\cos^{2\alpha-1}\phi}{\sin^{2g}\phi + \cos^{2g}\phi}\,\mathrm{d}\phi \\ =& 4^{1-\alpha}\frac{\Gamma(2\alpha)}{\Gamma^2(\alpha)}\int_0^{\frac{\pi}{2}} \frac{\sin^{2\alpha-1}(1-\cos\phi)^{2g}\phi}{\sin^{2g}\phi + (1-\cos\phi)^{2g}}\,\mathrm{d}\phi.\end{aligned} \tag{B.6}$$

Use the tangent half-angle substitution to simplify out the trigonometric functions:

$$\mathrm{Err}_0(\alpha, g) = 4\frac{\Gamma(2\alpha)}{\Gamma^2(\alpha)} \int_0^1 \frac{x^{2g+2\alpha-1}}{(1+x^{2g})(1+x^2)^{2\alpha}}\,\mathrm{d}x. \tag{B.7}$$

Simplify the last equation further by substituting $x = \mathrm{e}^{-t/2}$:

$$\mathrm{Err}_0(\alpha, g) = 2\frac{\Gamma(2\alpha)}{\Gamma^2(\alpha)} \int_0^\infty \frac{\mathrm{e}^{-(g+\alpha)t}}{(1+\mathrm{e}^{-gt})(1+\mathrm{e}^{-t})^{2\alpha}}\,\mathrm{d}t. \tag{B.8}$$

Optionally, the numerator and the denominator of the integrand can be multiplied by $\mathrm{e}^{(g+2\alpha)t}$:

$$\mathrm{Err}_0(\alpha, g) = 2\frac{\Gamma(2\alpha)}{\Gamma^2(\alpha)} \int_0^\infty \frac{\mathrm{e}^{\alpha t}}{(1+\mathrm{e}^{gt})(1+\mathrm{e}^{t})^{2\alpha}}\,\mathrm{d}t. \tag{B.9}$$

In the case of the original model, the base disagreement rate can be expressed via the gamma function:

$$\mathrm{Err}_0(\alpha, 1) = \frac{1}{2} - \frac{\Gamma\left(a+\frac{1}{2}\right)}{2\sqrt{\pi}\Gamma(a+1)}. \tag{B.10}$$

It clear that $\mathrm{Err}_0(\alpha, 1))$ in Equation B.10 is monotonously increasing w.r.t. parameter α. Since the model is defined for $\alpha \geq 1$, the minimum possible base disagreement rate is attained at $\alpha = 1$:

$$\min_{\alpha \geq 1} \left(\mathrm{Err}_0(\alpha, 1)\right) = \mathrm{Err}_0(1, 1) = \frac{1}{4}. \tag{B.11}$$

The integral in Equation B.9 does not seem to be representable in analytic form, but it can easily be computed numerically. Parameter g is present only in the denominator of the integrand; thus, the base disagreement rate monotonously decreases w.r.t. g.

C

ORDER-RANKED SUBMISSIONS

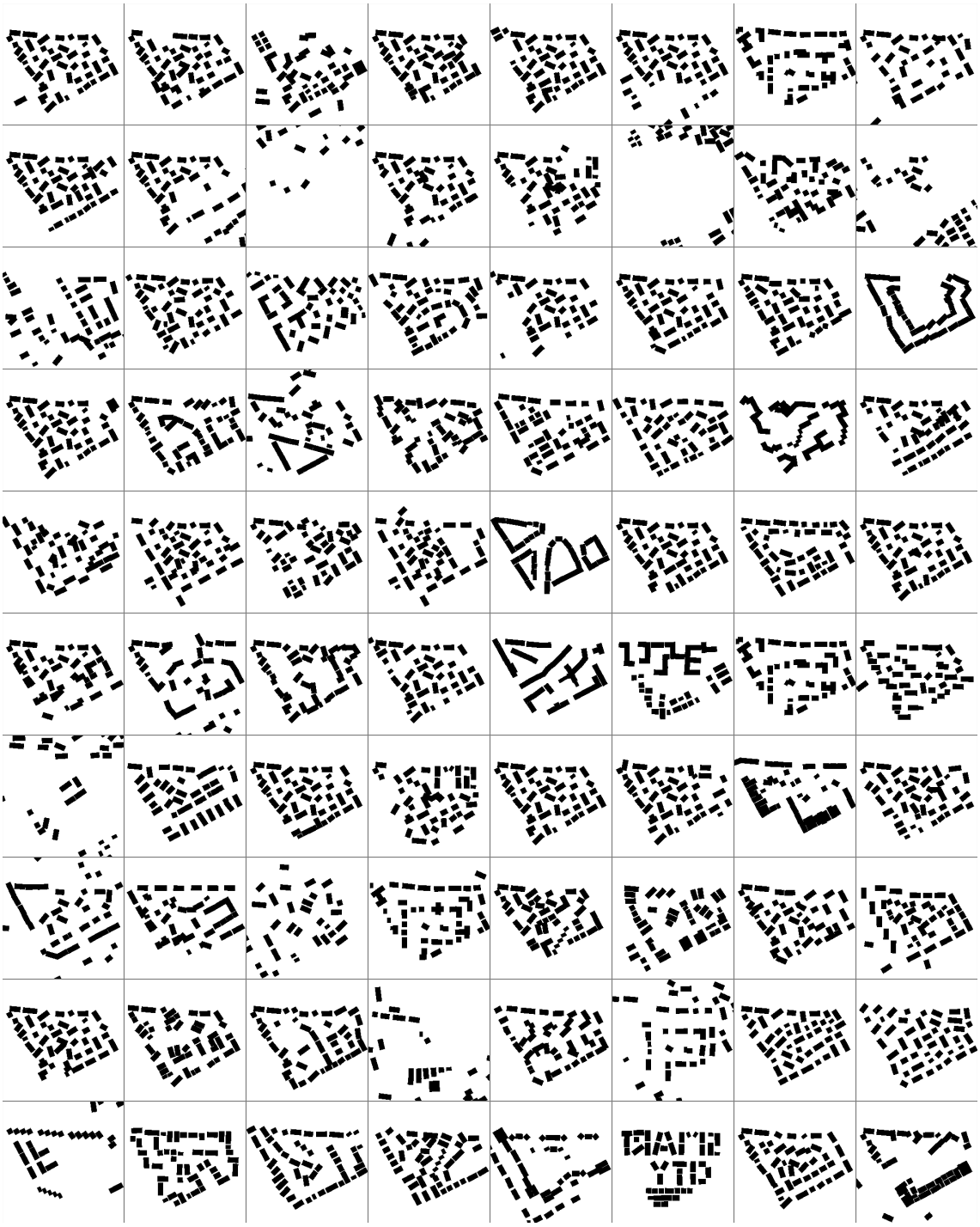


FIGURE C.1: Submissions with the lowest order scores according to the MAP estimate.

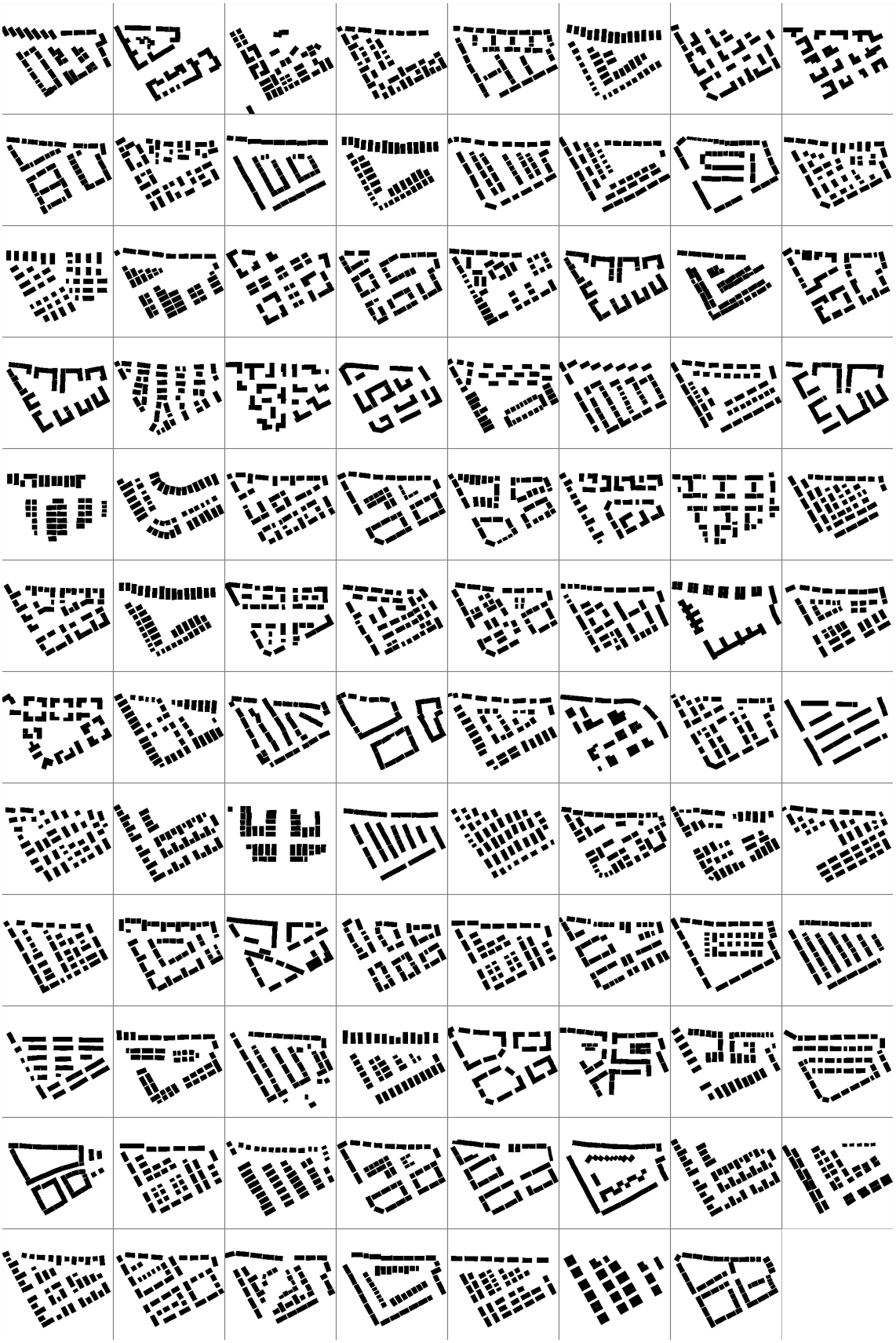


FIGURE C.2: Submissions with the highest order scores according to the MAP estimate.

D

MORE TESTS FOR THE REGRESSION MODEL

To prove that the regression disagreement rate is not affected by the comparison model overfitting the data, I perform additional tests in the form of the k-fold cross-validation analysis. First, I randomly reshuffle the sample of 943 unique submissions into k parts of similar size. Then, I run the model using $(k-1)$ parts of the sample for training and 1 part for the test.

In contrast to the tests presented in Section 4.1, I do not split the votes into training and CV samples for the comparison model and use all available votes for training. At the same time, nor the items of the training set neither the corresponding votes are used in the comparison model at all. Every time the test runs, the comparison model estimates a new set of scores for a required training set of submissions.

Figures D.1, D.2, D.3 illustrate the test results for 9-, 7-, and 5-fold cross-validation respectively. The results are consistent with Figure 4.4 and confirm that the average disagreement rate of the presented regression models is lower than 29.5%. Note, the length of the confidence intervals on the test-only comparison set is proportional to the number of items in the submission test set.

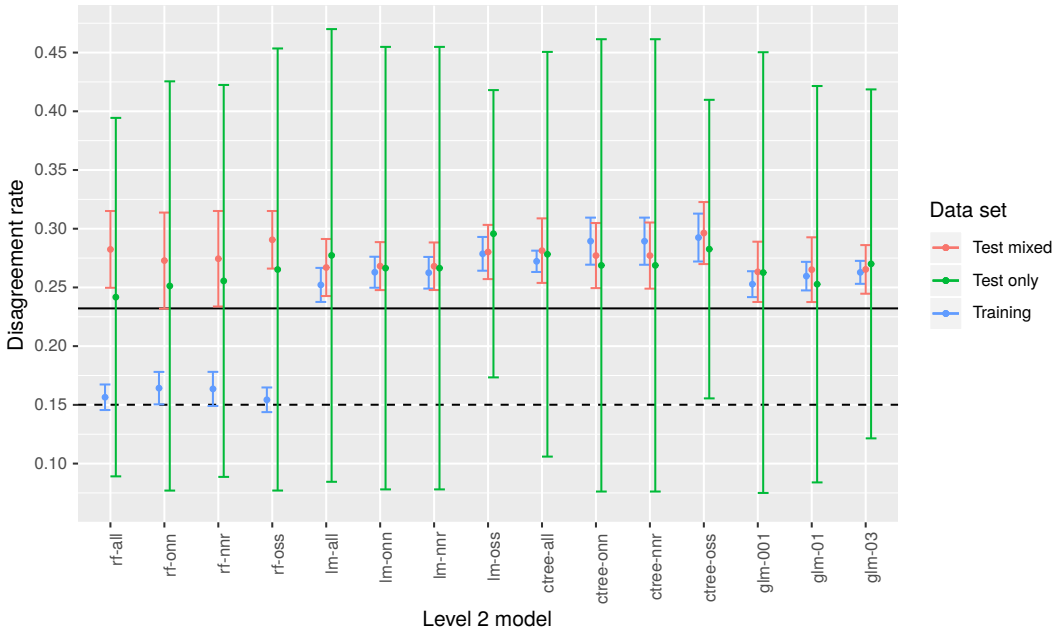


FIGURE D.1: Disagreement rates in 9-fold CV.

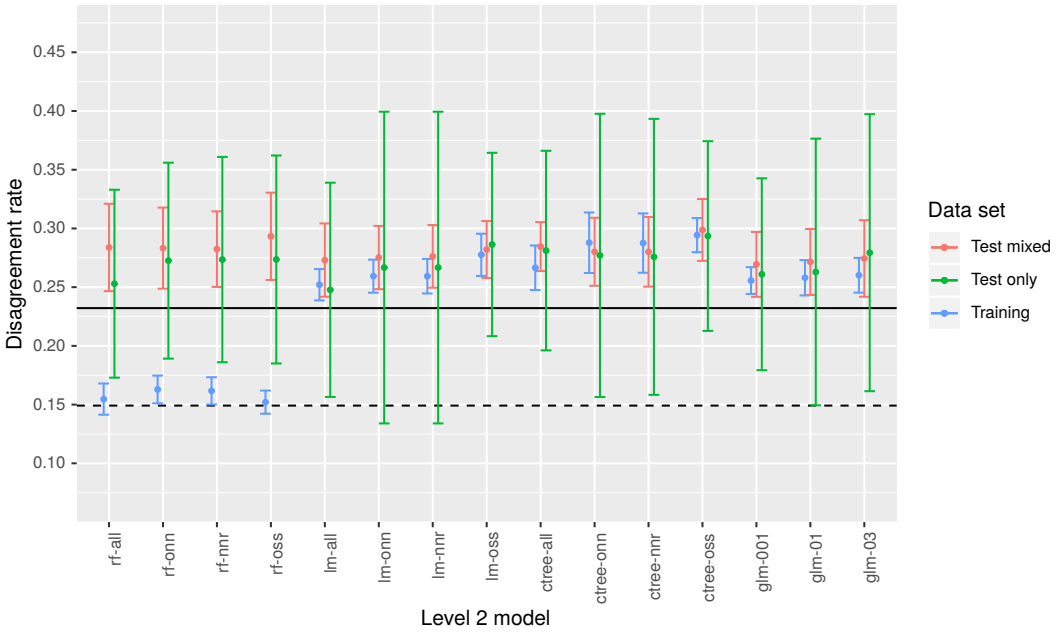


FIGURE D.2: Disagreement rates in 7-fold CV.

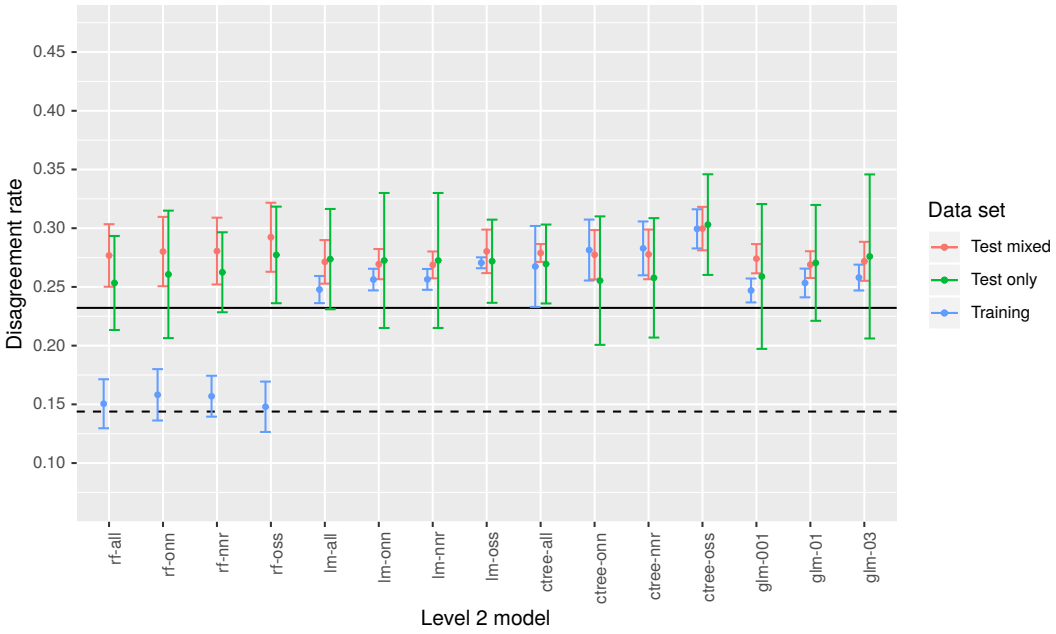


FIGURE D.3: Disagreement rates in 5-fold CV.

ONLINE SOURCES

[1] http://streetscore.media.mit.edu/, last accessed on 25.09.2018

[2] https://qua-kit.ethz.ch, last accessed on 30.09.2018

[3] https://www.yesodweb.com/, last accessed on 15.08.2018

[4] https://www.postgresql.org/, last accessed on 15.08.2018

[5] https://daemonite.github.io/material/, last accessed on 15.08.2018

[6] https://reflex-frp.org/, last accessed on 15.08.2018

[7] https://tools.ietf.org/html/rfc7946, last accessed on 22.10.2018

[8] https://github.com/achirkin/qua-view, last accessed on 22.10.2018

[9] https://www.imsglobal.org/activity/learning-tools-interoperability, last accessed on 22.10.2018

[10] http://edx.readthedocs.io/projects/edx-partner-course-staff/en/latest/exercises_tools/lti_component.html, last accessed on 15.08.2018

[11] https://github.com/achirkin/qua-kit/raw/reflex/doc/bin/specification.pdf, last accessed on 16.08.2018

[12] https://github.com/achirkin/qua-kit/tree/reflex/services/siren, last accessed on 16.08.2018

[13] https://www.edx.org/xseries/future-cities-0, last accessed on 20.08.2018

[14] http://u-tt.com/project/empower-shack/, last accessed on 20.08.2018

[15] https://www.edx.org/course/smart-cities-ethx-ethx-fc-03x-1, last accessed on 20.08.2018

[16] https://www.openstreetmap.org, last accessed on 20.08.2018

[17] http://www.fcl.ethz.ch/, last accessed on 23.10.2018

[18] https://www.edx.org/course/responsive-cities-1, last accessed on 23.10.2018

[19] http://ideasfortanjongpagar.com, last accessed on 20.08.2018

[20] https://github.com/achirkin/qua-kit, last accessed on 18.08.2018

[21] https://www.r-project.org/, last accessed on 01.09.2018

[22] https://github.com/achirkin/BradleyTerryScalable, last accessed on 01.09.2018

[23] https://github.com/achirkin/qua-kit/blob/reflex/apps/hs/qua-server/src/Handler/Mooc/Analysis/VoteOrder.hs, last accessed on 01.09.2018

BIBLIOGRAPHY

Adams, E. S. (2005). "Bayesian analysis of linear dominance hierarchies". In: *Animal Behaviour* 69.5, 1191.

Attneave, F. (1954). "Some informational aspects of visual perception". In: *Psychological review* 61.3, 183.

Bokeloh, M.; Berner, A.; Wand, M.; Seidel, H.-P.; Schilling, A. (2009). "Symmetry detection using feature lines". In: *Computer Graphics Forum*. Vol. 28. 2. Wiley Online Library, 697.

Bradley, R. A.; Terry, M. E. (1952). "Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons". In: *Biometrika* 39.3/4, 324.

Breiman, L. (2001). "Random forests". In: *Machine learning* 45.1, 5.

Caron, F.; Doucet, A. (2012). "Efficient Bayesian inference for generalized Bradley-Terry models". In: *Journal of Computational and Graphical Statistics* 21.1, 174.

Chen, T.; Li, M.; Li, Y.; Lin, M.; Wang, N.; Wang, M.; Xiao, T.; Xu, B.; Zhang, C.; Zhang, Z. (2015). "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems". In: *arXiv preprint arXiv:1512.01274*.

Chen, Y.; Sareh, P.; Feng, J.; Sun, Q. (2017). "A computational method for automated detection of engineering structures with cyclic symmetries". In: *Computers & Structures* 191.Supplement C, 153.

Chetlur, S.; Woolley, C.; Vandermersch, P.; Cohen, J.; Tran, J.; Catanzaro, B.; Shelhamer, E. (2014). "cuDNN: Efficient Primitives for Deep Learning". In: *CoRR* abs/1410.0759.

Chirkin, A.; Pishniy, M.; Sender, A. (2018). "Generilized Visibility-Based Design Evaluation Using GPU". In: *Proceedings of the 23nd CAADRIA Conference*. Vol. 2. Tsinghua University, Beijing, China, 483.

Chirkin, A. M.; König, R. (2016). "Concept of Interactive Machine Learning in Urban Design Problems". In: *Proceedings of the SEACHI 2016 on Smart Cities for Better Living with HCI and UX*. SEACHI 2016. San Jose, CA, USA: ACM, 10.

Ciresan, D. C.; Meier, U.; Masci, J.; Maria Gambardella, L.; Schmidhuber, J. (2011). "Flexible, high performance convolutional neural networks for image classification". In: *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*. Vol. 22. 1. Barcelona, Spain, 1237.

Cohen, A.; Zach, C.; Sinha, S. N.; Pollefeys, M. (2012). "Discovering and exploiting 3D symmetries in structure from motion". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 1514.

Corballis, M. C.; Roldan, C. E. (1975). "Detection of symmetry as a function of angular orientation." In: *Journal of Experimental Psychology: Human Perception and Performance* 1.3, 221.

Davis, L. S. (1977). "Understanding Shape. 2. Symmetry". In: *IEEE Transaction on Systems Man and Cybernetics* 7.3, 204.

Dresp-Langley, B. (2016). "Affine Geometry, Visual Sensation, and Preference for Symmetry of Things in a Thing". In: *Symmetry* 8.11.

Elo, A. E. (1961). "New USCF rating system". In: *Chess Life* 16, 160.

Embretson, S. E.; Reise, S. P. (2013). *Item response theory*. Psychology Press.

Ford, L. R. (1957). "Solution of a ranking problem from binary comparisons". In: *The American Mathematical Monthly* 64.8P2, 28.

Friedman, J.; Hastie, T.; Tibshirani, R. (2010). "Regularization paths for generalized linear models via coordinate descent". In: *Journal of statistical software* 33.1, 1.

Garner, W.; Clement, D. E. (1963). "Goodness of pattern and pattern uncertainty". In: *Journal of Verbal Learning and Verbal Behavior* 2.5, 446.

Giannouli, V. (2013). "Visual symmetry perception". In: *Encephalos* 50, 31.

Glickman, M. E. (1998). "The glicko system". In: *Boston University*.

Goldmeier, E. (1937). "Über Ähnlichkeit bei gesehenen Figuren". In: *Psychologische Forschung* 21.1. cited By 34, 146.

Gong, Q.; Li, J.; Liu, T.; Wang, N. (2018). "Generating urban fabric in the orthogonal or non-orthogonal urban landscape". In: *Environment and Planning B: Urban Analytics and City Science* 0.0, 2399808318761667.

Guiver, J.; Snelson, E. (2009). "Bayesian Inference for Plackett-Luce Ranking Models". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. Montreal, Quebec, Canada: ACM, 377.

Handley, J. C. (2001). "Comparative analysis of Bradley-Terry and Thurstone-Mosteller paired comparison models for image quality assessment". In: *PICS*. Vol. 1, 108.

Hatzinger, R.; Dittrich, R. (2012). "prefmod: An R Package for Modeling Preferences Based on Paired Comparisons, Rankings, or Ratings". In: *Journal of Statistical Software, Articles* 48.10, 1.

Henery, R. J. (1992). "An Extension to the Thurstone-Mosteller Model for Chess". In: *Journal of the Royal Statistical Society. Series D (The Statistician)* 41.5, 559.

Herbrich, R.; Minka, T.; Graepel, T. (2006). "TrueSkill^{TM}: a Bayesian skill rating system". In: *Advances in neural information processing systems*. Vol. 19. Cambridge, MA: MIT Press, 569.

Hothorn, T.; Zeileis, A. (2015). "partykit: A modular toolkit for recursive partytioning in R". In: *The Journal of Machine Learning Research* 16.1, 3905.

Howe, E. S. (1980). "Effects of partial symmetry, exposure time, and backward masking on judged goodness and reproduction of visual patterns". In: *Quarterly Journal of Experimental Psychology* 32.1, 27.

Hunter, D. R. et al. (2004). "MM algorithms for generalized Bradley-Terry models". In: *The annals of statistics* 32.1, 384.

Kaye, E.; Firth, D. (2017). *Fitting the Bradley-Terry model to large and potentially sparse datasets*. URL: `https://cran.r-project.org/web/packages/BradleyTerryScalable/vignettes/BradleyTerryScalable.html` (visited on 09/28/2018).

Kirby, M.; Sirovich, L. (1990). "Application of the Karhunen-Loeve procedure for the characterization of human faces". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.1, 103.

Klapaukh, R. (2014). "An Empirical Evaluation of Force-Directed Graph Layout". PhD thesis. Victoria University of Wellington.

Klumpner, H.; Brillembourg, A. (2014). "Made in Africa". In: *SLUM Lab* 9.

Krahe, J. L. (1986). "Detection of symmetric and radial structures in images". In: *International Conference on Pattern Recognition*, 947.

Kutateladze, S. S. (1976). "Symmetry measures". In: *Mathematical Notes* 19.4, 372.

Lee, S.; Liu, Y. (2010). "Skewed rotation symmetry group detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9, 1659.

Liaw, A.; Wiener, M., et al. (2002). "Classification and regression by randomForest". In: *R news* 2.3, 18.

Liu, Y.; Hel-Or, H.; Kaplan, C. S.; Gool, L. V. (2010). "Computational Symmetry in Computer Vision and Computer Graphics". In: *Foundations and Trends in Computer Graphics and Vision* 5.1–2, 1.

Lowe, D. G. (1999). "Object recognition from local scale-invariant features". In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Vol. 2. Ieee, 1150.

Loy, G.; Eklundh, J.-O. (2006). "Detecting Symmetry and Symmetric Constellations of Features". In: *Proceedings of the 9th European Conference on Computer Vision - Volume Part II*. ECCV'06. Graz, Austria: Springer-Verlag, 508.

Lu, H.; Gu, J.; Li, J.; Lu, Y.; Müller, J.; Wei, W.; Schmitt, G. (2018). "Evaluating Urban Design Ideas from Citizens from Crowdsourcing and Participatory Design". In: *Proceedings of the 23nd CAADRIA Conference*. Vol. 2. Tsinghua University, Beijing, China, 297.

Luce, R. D. (1977). "The choice axiom after twenty years". In: *Journal of Mathematical Psychology* 15.3, 215.

Marola, G. (1989). "On the detection of the axes of symmetry of symmetric and almost symmetric planar images". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11.1, 104.

McConnell, R. K. (1986). *Method of and apparatus for pattern recognition*. US Patent 4,567,610.

McCulloch, W. S.; Pitts, W. (1943). "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4, 115.

Miao, Y.; König, R.; Buš, P.; Chang, M.-C.; Chirkin, A.; Treyer, L. (2017). "Empowering Urban Design Prototyping". In: *Protocols, Flows and Glitches-Proceedings of the 22nd CAADRIA Conference, The Association for Computer-Aided Architectural Design Research in Asia (CAADRIA), China*, 407.

Miao, Y.; Koenig, R.; Knecht, K.; Konieva, K.; Buš, P.; Chang, M.-C. (2018). "Computational urban design prototyping: Interactive planning synthesis methods—a case study in Cape Town". In: *International Journal of Architectural Computing* 16.3, 212.

Minka, T.; Cleven, R.; Zaykov, Y. (2018). *TrueSkill 2: An improved Bayesian skill rating system*. Tech. rep.

Mitra, N.; Wand, M.; Zhang, H. R.; Cohen-Or, D.; Kim, V.; Huang, Q.-X. (2013a). "Structure-aware Shape Processing". In: *SIGGRAPH Asia 2013 Courses*. SA '13. Hong Kong, Hong Kong: ACM, 1:1.

Mitra, N. J.; Guibas, L. J.; Pauly, M. (2006). "Partial and Approximate Symmetry Detection for 3D Geometry". In: *ACM Transactions on Graphics* 25.3, 560.

Mitra, N. J.; Pauly, M.; Wand, M.; Ceylan, D. (2013b). "Symmetry in 3D Geometry: Extraction and Applications". In: *Computer Graphics Forum* 32.6, 1.

Morgan, Morris Hicky, trans. (1914). *The Ten Books on Architecture (De Architectura)*. By Marcus Vitruvius Pollio. Harvard University Press.

Morrone, M. C.; Burr, D. C. (1988). "Feature detection in human vision: a phase-dependent energy model". In: *Proceedings of the Royal Society of London B: Biological Sciences* 235.1280, 221.

Mosteller, F. (1951). "Remarks on the method of paired comparisons: I. The least squares solution assuming equal standard deviations and equal correlations". In: *Psychometrika* 16.1, 3.

Mouromtsev, D. I.; Sender, A. V.; Chirkin, A. M.; Lisitsa, N. I. (2017). "Automatic Analysis of Local Routes and Adjacent House Territory for Urban Planning Support". Russian. In: *Scientific and Technical Journal of Information Technologies, Mechanics and Optics* 17.1, 75.

Mueller, J.; Lu, H.; Chirkin, A.; Klein, B.; Schmitt, G. (2018). "Citizen Design Science: A strategy for crowd-creative urban design". In: *Cities* 72, 181.

Munowitz, M. (2006). "Symmetry Perforce". In: *Knowing: The Nature of Physical Law*. New York, NY, USA: Oxford University Press. Chap. 9.

Musialski, P.; Wonka, P.; Aliaga, D. G.; Wimmer, M.; Gool, L.; Purgathofer, W. (2013). "A Survey of Urban Reconstruction". In: *Comput. Graph. Forum* 32.6, 146.

Naik, N.; Philipoom, J.; Raskar, R.; Hidalgo, C. (2014). "Streetscore – Predicting the Perceived Safety of One Million Streetscapes". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Nikolenko, S. I.; Sirotkin, A. V. (2010). "Extensions of the TrueSkill^{TM} rating system". In: *Proceedings of the 9th International Conference on Applications of Fuzzy Systems and Soft Computing*. Citeseer, 151.

Ogawa, H. (1991). "Symmetry analysis of line drawings using the Hough transform". In: *Pattern Recognition Letters* 12.1, 9.

Park, M.; Lee, S.; Chen, P.-C.; Kashyap, S.; Butt, A. A.; Liu, Y. (2008). "Performance evaluation of state-of-the-art discrete symmetry detection algorithms". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 1.

Rasch, G. (1960). "Probabilistic models for some intelligence and achievement tests". In: *Copenhagen: Danish Institute for Educational Research* 56.

Reisfeld, D.; Wolfson, H.; Yeshurun, Y. (1995). "Context-free attentional operators: The generalized symmetry transform". In: *International Journal of Computer Vision* 14.2, 119.

Rock, I.; Leaman, R. (1963). "An experimental analysis of visual symmetry". In: *Acta Psychologica* 21, 171.

Ross, William David, trans. (1924). *Metaphysics*. By Aristotle. Oxford: Clarendon Press.

Scornet, E.; Biau, G.; Vert, J.-P., et al. (2015). "Consistency of random forests". In: *The Annals of Statistics* 43.4, 1716.

Sender, A. V.; Shiyan, A. V.; Chirkina, A. V.; Chirkin, A. M.; Mouromtsev, D. I. (2018). "An algorithm for search automation of lighting sources optimal arrangement in urban environment". Russian. In: *Scientific and Technical Journal of Information Technologies, Mechanics and Optics* 18.1, 122.

Shi, Z.; Alliez, P.; Desbrun, M.; Bao, H.; Huang, J. (2016). "Symmetry and Orbit Detection via Lie-Algebra Voting". In: *Computer Graphics Forum*. Vol. 35. 5. Wiley Online Library, 217.

Soegaard, M. (2010). "Gestalt principles of form perception". In: *Interaction-Design. org*, 8.

Stern, H. (1990). "A continuum of paired comparisons models". In: *Biometrika* 77.2, 265.

Thurstone, L. L. (1927). "A law of comparative judgment". In: *Psychological review* 34.4, 273.

Tolhurst, D. J.; Dealy, R. S. (1975). “The detection and identification of lines and edges”. In: *Vision Research* 15.12, 1367.

Treyer, L.; Klein, B.; König, R.; Meixner, C. (2016). “Lightweight urban computation interchange (LUCI): a system to couple heterogeneous simulations and views”. In: *Spatial Information Research* 24.3, 291.

Turner, H.; Firth, D. (2012). “Bradley-Terry Models in R: The BradleyTerry2 Package”. In: *Journal of Statistical Software, Articles* 48.9, 1.

Vasiliev, A. (1984). “Recognition of Symmetrical Patterns in Images”. In: *Proceedings - International Conference on Pattern Recognition*. Vol. 2. IEEE, 1027.

Wang, Y.; Zhang, L.; Mathiopoulos, P. T.; Deng, H. (2015). “A Gestalt rules and graph-cut-based simplification framework for urban building models”. In: *International Journal of Applied Earth Observation and Geoinformation* 35, 247.

Welch, E.; Kobourov, S. (2017). “Measuring Symmetry in Drawings of Graphs”. In: *Comput. Graph. Forum* 36.3, 341.

Werbos, P. (1975). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University.

Weyl, H. (1952). *Symmetry*. Princeton, New Jersey. Princeton University Press, p. 5.

Wu, C.; Frahm, J.-M.; Pollefeys, M. (2010). “Detecting Large Repetitive Structures with Salient Boundaries”. In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 142.

Yan, H.; Weibel, R.; Yang, B. (2008). “A multi-parameter approach to automated building grouping and generalization”. In: *Geoinformatica* 12.1, 73.

Yip, R. K. (2000). “A Hough transform technique for the detection of reflectional symmetry and skew-symmetry”. In: *Pattern Recognition Letters* 21.2, 117.

Yodogawa, E. (1982). “Symmetropy, an entropy-like measure of visual symmetry”. In: *Perception & Psychophysics* 32.3, 230.

Zabrodsky, H.; Peleg, S.; Avnir, D. (1992). “A measure of symmetry based on shape similarity”. In: *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 703.

Zermelo, E. (1929). “Die Berechnung der Turnier-Ergebnisse als ein Maximumproblem der Wahrscheinlichkeitsrechnung”. In: *Mathematische Zeitschrift* 29.1, 436.

Zhang, L.; Lin, W. (2013). “Computational Models for Top-down Visual Attention”. In: *Selective Visual Attention*. Wiley-Blackwell. Chap. 5, 167.

Artem Chirkin

MSc Computational Science

Wolfgang-Pauli-Strasse 27
HIT H 31.6
CH-8093 Zurich, Switzerland
☎ *+41 44 633 79 62*
+41 78 811 56 46
✉ *chirkin@arch.ethz.ch*

Personal data

Date of Birth January 16, 1990

Nationality Russian Federation

Education

2014–now **PhD candidate**, *Zürich, Switzerland*, *ETH Zürich*.
Swiss Federal Institute of Technology in Zürich
Chair of Information Architecture
Dr. sc. ETH Zürich

2013–2014 **MSc**, *Amsterdam, Netherlands*, *UvA*.
University of Amsterdam
MSc Computational Science

2012–2014 **MSc**, *St. Petersburg, Russian Federation*, *ITMO University*.
National Research University of Information Technologies, Mechanics and Optics
Master of Applied Mathematics and Computer Science
010400.68 "Computational Science in Multidisciplinary Research"

2008–2012 **BSc**, *St. Petersburg, Russian Federation*, *ITMO University*.
National Research University of Information Technologies, Mechanics and Optics
Bachelor of Applied Mathematics and Computer Science
010400 "Applied Mathematics and Computer Science"

Job experience

2014-now ETH Zürich, Chair of Information Architecture. Research assistant

2012-2013 "ACLab of NRU ITMO". Engineer (1 year)

2011-2012 "BTC-solutions". Java Developer (10 months)

2010-2011 "OpenWay". Intern of Customer Support Department (5 months)

2010 "Summer School OpenWay". Six-week internship in The OpenWay Company. Participation in a team project as a project manager and developer.

Publications (first author)

1. Chirkin, A., Pishniy, M. & Sender, A. *Generilized Visibility-Based Design Evaluation Using GPU* in *Proceedings of the 23nd CAADRIA Conference* **2** (Tsinghua University, Beijing, China, 2018), 483–492.
2. Chirkin, A. & König, R. *Concept of Interactive Machine Learning in Urban Design Problems* in *Proceedings of the SEACHI 2016 on Smart Cities for Better Living with HCI and UX* (ACM, San Jose, CA, USA, 2016), 10–13. ISBN: 978-1-4503-4194-3. doi:10.1145/2898365.2899795. http://doi.acm.org/10.1145/2898365.2899795.
3. Chirkin, A., Belloum, A. S. Z., Kovalchuk, S. & Makkes, M. X. *Execution time estimation for workflow scheduling* in *Proceedings of the 9th Workshop on Workflows in Support of Large-Scale Science* (2014), 1–10.
4. Chirkin, A. *Execution Time Estimation in the Workflow Scheduling Problem* MA thesis (University of Amsterdam, the Netherlands, 2014). http://dare.uva.nl/en/scriptie/496303.
5. Chirkin, A. & Kovalchuk, S. Towards Better Workflow Execution Time Estimation. *IERI Procedia* **10,** 216–223 (2014).

Publications (contributor)

6. Mueller, J., Lu, H., Chirkin, A., Klein, B. & Schmitt, G. Citizen Design Science: A strategy for crowd-creative urban design. *Cities* **72,** 181 –188. ISSN: 0264-2751 (2018).
7. Chang, M.-C., Buš, P., Tartar, A., Chirkin, A. & Schmitt, G. Big-Data Informed Citizen Participatory Urban Identity Design (2018).
8. Sender, A., Shiyan, A., Chirkina, A., Chirkin, A. & Mouromtsev, D. An algorithm for search automation of lighting sources optimal arrangement in urban environment. Russian. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics* **18,** 122–132 (2018).
9. Mouromtsev, D., Sender, A., Chirkin, A. & Lisitsa, N. Automatic Analysis of Local Routes and Adjacent House Territory for Urban Planning Support. Russian. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics* **17,** 75–80 (2017).
10. Miao, Y. *et al.* Empowering urban design prototyping: a case study in Cape Town with interactive computational synthesis methods (2017).
11. Klein, B., Chirkin, A., Standfest, M., Schmitt, G. & König, R. in *The Virtual and the Real in Planning and Urban Design* 93–111 (Routledge, 2017).