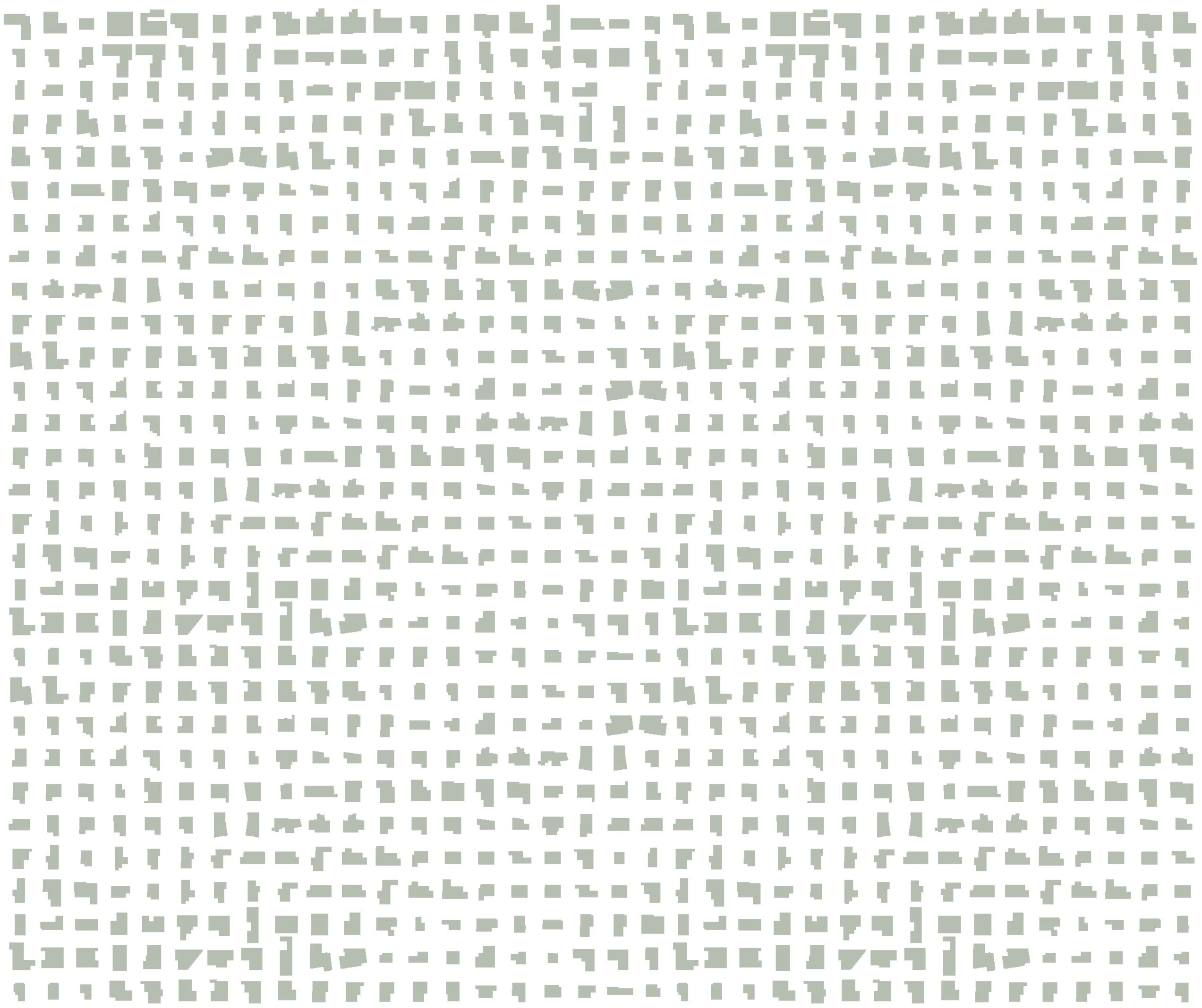
FLOOR PLAN PARAMETRICS

- Procedural Floor Plan Generation Within Residential Development



Sofia Malmsten | Architecture and Urban Design | Chalmers University of Technology

ABSTRACT

Floor plan design is regarded as on of the major tasks within architecture and housing development. It is a challenge of creating appropriate shapes and locations of rooms, and the process normally requires parallel design steps across different scales. The task can often be complex and results in a time consuming process.

”Floor Plan Parametrics” is an investigation on how algorithms can be developed in order to support the floor plan creation. The focus is on generative design - a process where architects formulates rules and constrains, and a software generates possible solutions. By defining rules for floor plans the algorithm generates multiple possible room configurations inside a given apartment boundary.

The generative design approach applied on floor plans could assist the human brain, facilitate decision-making and streamline the planning. Risks and consequences can be foreseen at an early stage and architects can investigate a wider design scope in order to make informed decisions faster.

By investigating procedural algorithms from the gaming industry together with a set of architectural floor plans from already built projects a code prototype is developed. The implementation of the core mechanics of this algorithm is described together with required architectural aspects to take into consideration.

“Floor Plan Parametrics” will contribute to improved decision support and facilitate how floor plans can be created in a shorter period of time.

Sofia Malmsten

Bachelor degree - Architecture and Engineering - Chalmers University of Technology

Master program - MsC MPARC - Chalmers University of Technology

Chalmers University of Technology, Sweden 2020

Examinor: Ola Nylander
Tutor: Kaj Granath
Direction: Housing

Keywords: Floor plans, Generative Design, Apartments, Residential, Procedural Algorithms

CONTENT

01 INTRODUCTION

THE AUTHOR

EDUCATION

Mechanical Engineering, 1,5 years
Chalmers University of Technology

Architecture and Engineering, 3 years
Chalmers University of Technology

Architecture and Urban Design, MPARC, 2 years
Chalmers University of Technology

WORKING EXPERIENCE

NCC Housing, Gothenburg
Intern - Design Studio, 2014-2015
Brand positioning, Architectural trends within housing

Bonava Sverige AB, Stockholm
Intern - Design Studio, 2015-2018
Database development, Revit, Housing development, Simulations, Parametric design, Grasshopper, Social Sustainability

Buro Happold Engineering, Bath , UK
Industrial Placement - Sustainability and Physics, 2017-2018
Interoperability, Programming in C#, digital processes and BIM work flows, Revit, Rhino, Grasshopper, Parametric design

Architude, Gothenburg
Architect and Generative Design Consultant, 2019 -
Parametric and generative design, scripting in C#, solutions for BIM environments and work flows, web development

PROBLEM STATEMENT

Due to regulations and requirements housing developers and architects are facing a complex design process. Buildings need to have enough living space and every apartment need to get enough daylight. At the same time the buildings need to be energy efficient and provide the best terrace/balcony orientation as well as good views to the outdoors.

The list of aspects to consider goes on and results in a complex design process. In many other industries, digital tools and AI bring methods to tackle their complex and previously unresolved challenges, but within housing development it is still a relatively unexplored subject. BIM has been a popular topic for a while though and has meant that most of the housing planning today is done digitally, but the work flows and mindsets have still not been used to their full potential.

This thesis aims to investigate how to tackle complexity by using algorithms and digital tools. Focus is on the fundamental piece within residential design - the floor plan.

PURPOSE AND AIM

The hypothesis is that it is possible to digitize and decode the logic behind architecture in the design of residential floor plans in order to increase the efficiency within the housing development. The intention is to test that hypothesis and demonstrate the benefits of investigating a larger design scope in a short period of time through digital processes.

The aim is to highlight a generative design process where the architect formulates rules and constraints, and a software generates possible solutions as drafts for further development (figure 1.0). This enables architects to investigate a wider design scope.

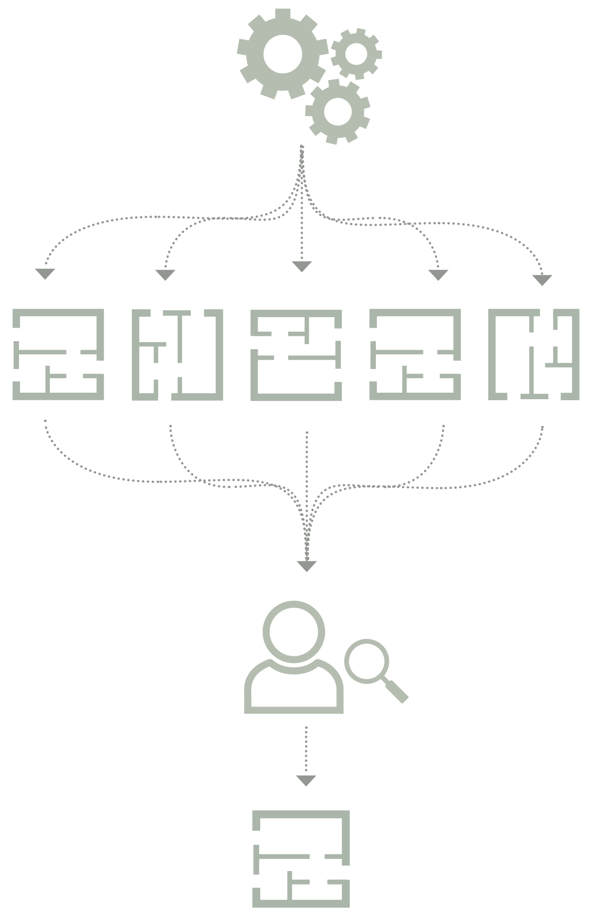


Figure 1.0

RESEARCH QUESTIONS

1

How can the logic behind residential floor plans be translated into quantifiable variables by using statistics of existing floor plans?

2

How well can an algorithm generate residential floor plans similar to those created by using traditional architectural methods?

METHOD AND PROCESS

The study is conducted in two parallel phases, where the first phase focuses on analysing existing floor plans. Aspects affecting the floor plan layout is investigated as well as how the complexity needs to be divided by dealing with problems in sub sequential steps. The focus is both ”research on design” and ”research for design”.

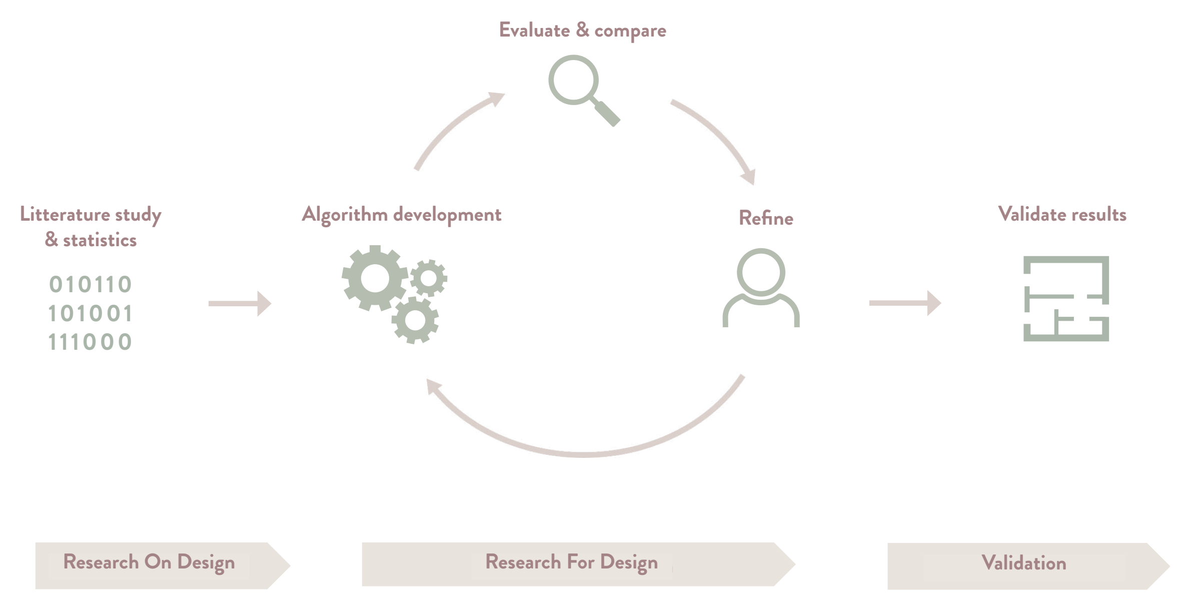
A literature study is conducted in order to examine what has already been done in terms of algorithms and space planning. Different approaches used within the field of architecture is investigated and evaluated. Approaches and algorithms from the gaming industry is also reviewed.

Research on design
In order to crate floor plans we need to understand why existing floor plans look like they do. Therefore the first step in the process is to analyse 525 floor plans and extract digital information from them. The chosen aspects to consider is based on recurrent appearance in the literature study.

Research for design
Research for design defines the second step in the process. An algorithm is developed in C# and the final product is a script executed in Grasshopper. By constantly evaluating the results the algorithm is modified in order to produce better results.

Multiple alternatives for a floor plan can be generated in milliseconds. Either a random selected layout can be used and further developed by an architect or all of the generated solutions can be evaluated and compared. Based on defined evaluation criteria the optimal solution for a given boundary can be selected and then further developed.

Validation
As a final step produced floor plans are visually validated and compared with real architectural floor plans.



DELIMITATIONS

”The creation of floor plans is a non-trivial technical challenge and its design is a sequential process that requires successive design steps across different scales - urban scale, building scale, unit scale”

- Stanislas Chaillou, Harvard Graduate School of Design | Feb. 24th, 2019

Scales
The creation of floor plans normally requires design steps across different scales (Chaillou S, 2019). In this project the limitation is set to unit scale. The algorithm will therefore not change the boundary of the apartment.

External aspects, such as adjacent apartments or site specific inputs, will not be taken into consideration in the project.

Geometrical inputs
The demarcation has been set to create floor plans within a defined boundary. External door placement (called access point here) is fixed, but by manually changing the position the floor plan will be changed as well. The facade, represented by a poly line, is a user input as well.

Levels
Application of auto generated floor plans will be done on a normal level in a multi-family house. Theoretically, the method can also be applied to a single family house in one level (Rodrigues A, 2013), but not on a house or apartment in two levels since vertical connections will not be treated.

National Regulations
In this project, Swedish regulations will be applied and considered as much as possible. However, rules and national constraints will largely be used as inputs to the algorithms in order to enabling future applications for projects with other national regulations.

Tolerances
Wall thickness and tolerances will not be treated as the focus in the project is conceptual space planning in early stages. Consideration will as much as possible be given to the functional dimensions and aspects according to SIS and BBR, but the result will not insure correct tolerances and dimensions.

Evaluation criteria
Measurable goals for evaluation of floor plans are not treated in this thesis. The focus will be on the generation part in a generative design work flow.

Floor plan references
A data set is built up as a reference bank for analysing and comparing the generated floor plans against real architectural plans. The references are built projects in Sweden between 2017 and 2020.

02 THEORY

TERMINOLOGY

Access Point
The word access point is used for describing the position of the centre of the entrance door by an x and y coordinate.

Adjacency
Adjacencies are described as a graph with nodes and edges. Each space in a floor plan represents a node, and each connection represents an edge. Adjacency does not take circulation into consideration.

AEC
Architecture, Engineering and Construction

BIM
BIM, Building information modelling, is a process supported by various tools and technologies involving the management and creation of digital representations of physical and functional characteristics of a facility. (National BIM standards, US)

Boundary
The word boundary is used for describing the geometrical outline of an apartment in the xy plane. It is represented as a closed 2D poly line describing the inner lines of the bounding walls for an apartment.

C#
Programming language in the .NET framework that will be used for writing the logics of the floor plan algorithms

Circulation
Circulation is a subset from the adjacency graph. The circulation graph represents possible walking paths through doors and openings in an apartment.

GAN
A generative adversarial network (GAN) is a machine learning system. Given a training set, this technique learns to generate new data with the same statistics as the training set.

Generative Design
Generative design is as well as parametric design a design process based on algorithmic thinking. It is an iterative process that involves a program for generating designs and a designer for defining rules and constraints as well as fine-tuning and evaluating the outcomes.
For each iteration in the iterative process the designer learns to refine the algorithms as the design goals become better defined over time (*Meintjes K, 2018)*.

The program does not need to be executed on a machine, it can also be done by a human with pen and paper. The designer does neither need to be a human, it can be an evaluation program in a testing environment or an artificial intelligence, for example a generative adversarial network - GAN.

Grasshopper
The Graphical User interface for Rhino. The algorithms for generating floor plans will be executed in this environment.

ICT
Information and communications technology

Interoperability
Interoperability describes the capability of different programs to exchange data via a common set of exchange formats.

Neural Networks
Neural networks are a set of algorithms, modelled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labelling or clustering raw input. A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes.

Parametric Design
Parametric design is a design process based on algorithmic thinking (Jabi W, 2013). The process focuses on parameters and rules that, together, define the relationship between design inputs and design outputs.

PCG
PCG (Procedural content generation) is a method for generating environments automatically using algorithms. PCG is a common approach in the gaming industry.

Parametricism
An architectural style based on computer technology and algorithms

Program
List of spaces with their absolute areas in m2

Revit
A BIM software for architectural design, MEP, structural design, detailing, engineering, and construction.

Space
A space, or zone, is in this project defined by a closed poly line in 2D. The poly line can be seen as the centre line for walls or an approximated zone division in an open floor plan, e.g.. between a kitchen and a living-room.

Space Syntax
Space syntax encompasses a set of theories and techniques for the analysis of spatial configurations.

Synthesis
The combination of components or elements to form a connected whole. In this project the synthesis represent the combination of components relevant for generated floor plans from an algorithmic perspective.

INVENTIONS & INNOVATIONS

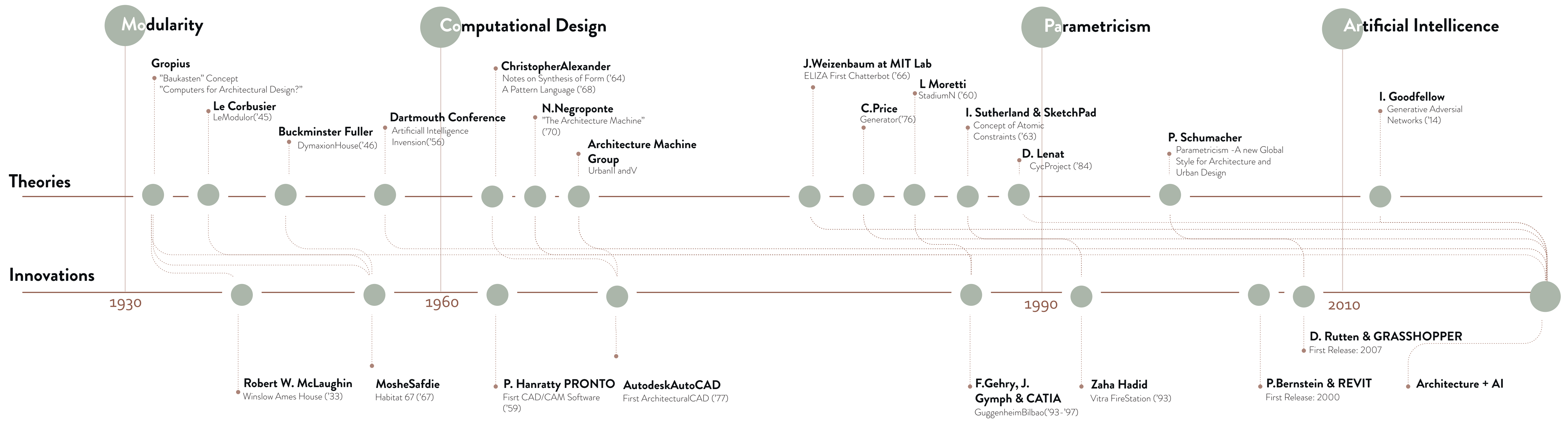


Figure 1.3 - Source: Stanisalas Chailou, 2019
Note that modifications have been made to the diagram

RELATED WORK

Since the 1960's various methods have been developed for computer-based solutions of layout problems (Whitehead & Eldars, 1964; Frew 1980). From the perspective of complexity theory, layout problems fall into the category of so-called NP-complete (Non-deterministic polynomial time) problems.

A literature study has been conducted in order to map previously done research and attempts to create floor plan solutions with algorithms. The literature study covers both related work from interdisciplinary research areas, such as the gaming industry, as well as research areas within the field of architecture and urban planning. The previously used methods and approaches is presented in this chapter.

Kalay (2004) described thee primary method categories that are relevant for layout problems: procedural, heuristic and evolutionary methods. In this thesis procedural algorithms are the main focus. The algorithms come originally from the gaming industry and there are a various amount of approaches and techniques for procedurally generating content for virtual environments.

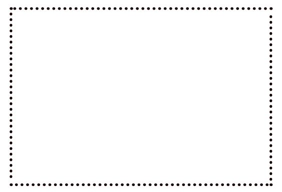
Techniques for content generation creates environments according to parameters of an algorithm rather than through manual creation by hand. These types of techniques have been proposed for a lot of aspects of virtual worlds, ranging from buildings to landscapes(Bower M, 2017).

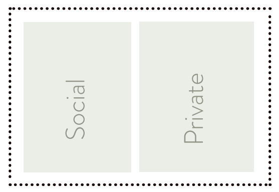
Five types of techniques for generating floor plans procedurally have been identified throughout the literature study and more elaborate descriptions can be found in the following sections.

As an addition to the three main categories presented by Kalay (2004) machine learning approaches have been investigated as well. Stanislas Chaillou (2019) shown how Generative Adversarial Networks can be applied on floor plans by letting training data in form of floor plans images train a network to create new images of floor plans.

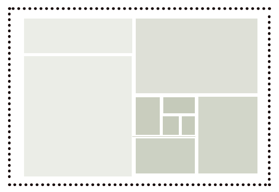
PROCEDURAL ALGORITHMS

01. SUBDIVISION


(a) Boundary as input



(b) Zone subdivision


(c) Zones to room subdivisions

Subdivision is a method where the boundary is subdivided recursively into smaller areas. There are however a multitude of variations on how to perform this subdivision and three recurrent methods are presented below.

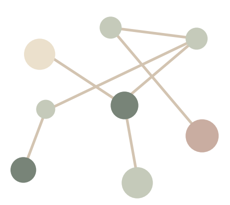
Marson and Musse (2010) introduce a room subdivision method based on squarified treemaps, a method for displaying hierarchical data using nested rectangles. Their methods input is a boundary represented by a poly-line in 2D and a list of rooms, with preferred dimensions and their social aspects, e.g. social area and private area. The algorithm recursively subdivide an area into smaller areas, and the algorithm tries to maintain an aspect ratio of 1 for all of the areas.

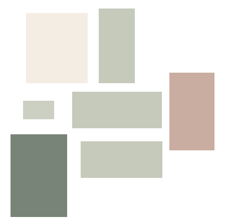
First their subdivision algorithm divide the boundary into zones. These zones are again subdivided to create into rooms. Possible connections between room types are pre-defined and doors are placed between the rooms in the generated floor plan. The shortest path is determined, connecting all rooms that need a connection with the passage. This path is transformed into a passage, and all rooms are adjusted to make room for it.

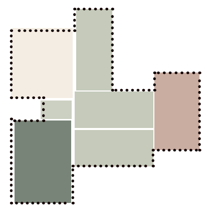
Hahn et al (2006) also focuses on rectangular floor plans but within office buildings. Starting from the building structure a hallway subdivision is applied. Next, the remaining regions are subdivided into spaces, for which a geometry is created.

Rhinde (2008) presented an subdivision implementation for no rectangular shapes. The inputs are outer walls, windows and doors and the division is based on the boundary's discontinuity points as well as the points between windows.

02. INSIDE-OUT


(a) Graph with rooms as nodes and adjacencies as edges


(b) Graph transformed to a 2D spatial layout


(c) Generated boundary encapsulating the rooms

An inside-out algorithm approach is conducted in two steps. In the first step the rooms are generated and in the second step the boundary is created from the outermost walls of the generated room layout.

Martin (2006) proposes an inside-out approach based on graph theories where he generates the room layout and then builds the exterior from that.

His algorithm starts by generating a graph with nodes representing the rooms and edges representing the connections between them. The entrance is seen as the root node and is the starting point for the graph creation.

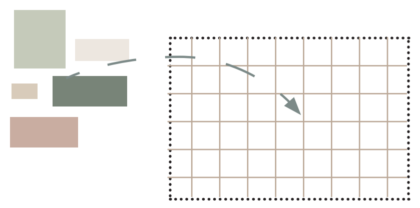
As the second step public rooms are added (an assumption is made that these type of rooms often are close to the entrance). Subsequently, private rooms are attached to the public rooms and as the last step stick-on rooms like closets are introduced.

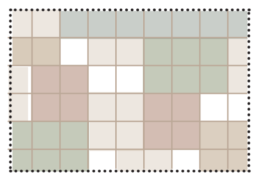
Once the graph is generated it is transformed to a 2D spatial layout. For each node, depending on the desired size of the room, a specific amount of "pressure" is applied to make them expand towards their desired size or a pressure equilibrium.

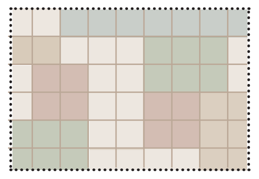
As a final step the boundary can be defined as common encapsulating perimeter of the placed rooms.

Merell (2010) present an inside-out method based on Machine Learning for automated generation of building layouts. The architectural program is created by using a Bayesian network trained on real-world data. The architectural program (the graph) is realized in a set of floor plans, obtained through stochastic optimization.

03. TILE PLACEMENT


(a) Predefined room pieces and quadrilateral mesh


(b) Placed pieces and remaining gaps


(c) Water tight layout

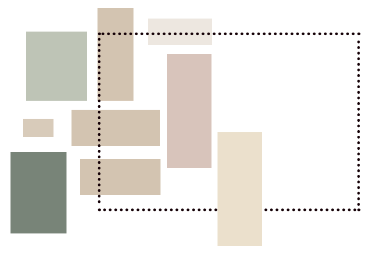
Tile placement algorithms tessellates a 2D domain into a quadrilateral mesh. Just like Tetris the algorithm create puzzle pieces which can be fit together to form a floor plan.

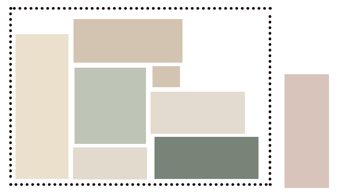
Peng et al. (2014) presented a method for covering an arbitrary shaped (e.g., non-axis aligned) 2D domain in a gap-less manner with flexible, non-overlapping, puzzle-like pieces. The pieces, which can be seen as templates for rooms, are predefined and placed recursively within the boundary until no more pieces can be placed.

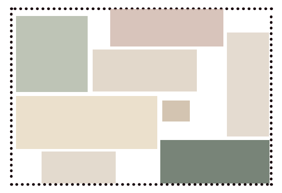
The pieces can be resized to a certain degree in order to avoid space between them, forming what the article calls a water-tight layout.

What the algorithm does not take into consideration is adjacency and circulation constraints. The aim is to fill up a shape with pieces without considering their relative positions. The algorithm has been applied on parcel creation within urban development and other design areas where adjacencies are irrelevant.

04. DENSE PACKING


(a) Predefined rooms and predefined boundary


(b) Space configuration resulting gaps and one room outside

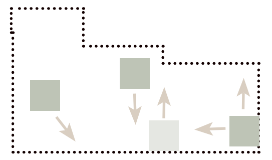

(c) Space configuration resulting in gaps

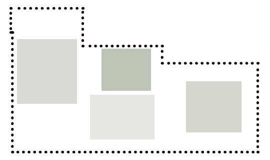
Dense Packing is an optimization problem in mathematics that attempts to pack objects together into a container. The goal is to pack a single container, in this case the boundary of a floor plan, as densely as possible.

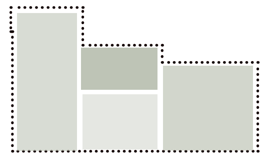
The algorithm will try different placement configurations of predefined shapes, in this case the rooms, and will try to fit as many as possible.

Based on the concept of physical objects representing rooms that are connected with edges Elezkurtaj and Franck (2001, 2002) have shown how dense packing can be applied on floor plans. The problem to be solved was formally described as follows: minimize the sum of all overlapping room areas. The sum is calculated from the sum of overlapping areas of all spaces to be packed.

05. ROOM GROWTH


(a) Axis-aligned irregular boundary and initial positions for the rooms


(b) Spaces once they have reached their preferred areas


(c) Adjustments in order to create a water-tight layout

The Room Growth method generates seeds and grows these seeds iteratively until each individual room has reached its final size and shape.

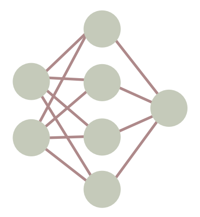
Tutenel et al. (2009) applied a constraint-solving approach to floor plan generation where every type of room is mapped to a class in an external library. In this context, constraints typically define room adjacencies. For each room to be placed, a rectangle of minimum size is positioned at a location where all defined relation constraints hold, and all these rooms expanded until they touch other rooms.

Constraint solving techniques do often create rectangular shapes. Many of them cannot handle irregular shapes, such as L-shaped or U-shaped rooms, which also holds for many of the other techniques discussed in the previous sections. This is according to Lopes et al. (2014) one of the main drawbacks of all these approaches and they therefore presented a constrained growth method for procedural floor plan generation allowing these irregular room shapes. Their method is based on an approach where geometric grids are used as a canvas for generating the spaces and their positions.

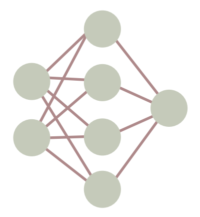
Camozzato (2015) did also present a grow-based procedural method to create floor plans considering user-provided inputs as well as the constraint of a building's exterior walls. First, a grid is created, and then, each room is placed to occupy a single cell in the grid. It is subsequently expanded, occupying adjacent cells to achieve its final size and this growth-based approach can generate different interior models for a wide range of different apartment boundaries.

MACHINE LEARNING

GENERATIVE ADVERSARIAL NEURAL NETWORKS


(a) Generator generating a random image

(b) Generated image (top) vs real images (bottom)


(c) Discriminator evaluating if the image is real or fake.

Stanislas Chaillau presented in 2019 an approach for generating floorplans by using Generative Adversarial Neural Netwroks.

As any machine-learning model, GANs learn statistically phenomena among data presented to it. The GAN structure is made of two key networks, the Generator and the Discriminator, and a feedback loop is created between them in order to refine the ability to generate relevant images.

Once the model is trained, it can distinguish between a real example from the dataset and a "fake" image. The Generator is trained to create images resembling images from the same dataset. As the Generator creates images, the Discriminator provides it with feedback about its quality. In response, the Generator adapts better and more realistic images. Through this feedback loop, a GAN slowly builds up its ability to create relevant artificial images.

Another GAN approach was presented by Chin-Ye Chen 2020. He uses GANs for graph-constrained House Layout Generation and train the networks to create floor plans from graphs and bubble diagrams.

GANs shows promising results but the provided training data is crucial. It can be difficult to get enough training material and the model will only be as good as the data we provide (Chaillau, 2019).

APPROACH COMPARISON

The methods in the related work are compared in table 1. The evaluation criteria are the required inputs for each and every algorithm. Relevant inputs to compare in this care are outline constraints, window constraints, room requirements and adjacency requirements.

Some comparisons have already been made. For instance Reinhard et al. compared dense packing algorithm and subdivision algorithm and concluded that subdivision algorithm can generated valid solutions by ratio 90%, while dense packing algorithm could generate valid solution by ratio 85%. Though subdivision rarely works for non rectangular shapes, and since floor plan boundaries in most cases are non rectangular the arbitrary outline/axis-aligned constrain is important.

The procedural growth method presented by Lopes (2010) meet most of the wanted requirements and shows promising results in previous papers. Therefore this method is chosen as a point of departure for further investigation and as the basis for the code prototype described in the next chapter.

What is excluded from the presented approach is the relation to the facade and the entrance door. This will be discussed and added to the developed algorithm in the next chapters.

Reference	**Type**	**Outline constr.**	**Window constr.**	**Room req.**	**Adjacency req.**	**Training data req.**
Hahn 2006	Subdivision	Rectangle	No	No	No	No
Rinde 2008	Subdivision	Arbitrary	Yes	No	No	No
Tutenel 2009	Room growth	Axis-aligned	No	No	Yes	No
Marson 2010	Subdivision	Rectangle	No	Yes	No	No
Lopes 2010	Room growth	Axis-aligned	No	No	Yes	No
Merell 2010	Inside out	No	No	Yes	Yes	Yes
Peng 2014	Tile plac.	Arbitrary	No	No	No	No
Camozzato 2015	Room Growth	Arbitrary	No	Yes	Yes	No
Chaillau 2019	GAN	No	Yes	No	No	Yes

03 SPACE ANALYSIS

DATA SET

”Real-world architectural programs have significant semantic structure. Such relationships are numerous and are often implicit in architects’ domain expertise. It is not clear how they can be represented with a hand-specified set of rules ”

- Alfredo Andia, Post-Parametric Automation in Design and Construction | 2015

The challenge of generating floor plans is primarily a challenge of generating appropriate locations and areas for the rooms inside a certain boundary. The architectural program needs to be compatible with the shape and other spatial requirements and in order to find compatible programs for a given shape the relationship between these two features is investigated.

References
For the purpose of this work 525 floor plans, from 35 projects of different typologies, were manually encoded* and stored in a database. The plans and their correlated data are collected from 7 different housing development companies in Sweden and represents apartments in multi family housing in different contexts from 2017 to 2020.

Floor plan parameters
Floor plan attributes from these 525 apartments have been recorded on two different levels: space level and inter-space (adjacency) level. All parameters are described in the next section.

Standard deviation
The standard deviation measures the variation in a data set. A low standard deviation indicates that most of the values in the set are close to the mean value, whereas a high standard deviation indicates that the values in the set are spread out over a wider range. The standard deviation is calculated as:

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$$

σ = population standard deviation
N = population size
x_i = each value from the population
μ = the mean value

*See appendix for measurment rules

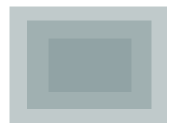
PARAMETERS

Area

The area is the most important and trivial parameter describing a room. The average area, as well as the min and max area, is calculated for each and every room type in the data set.

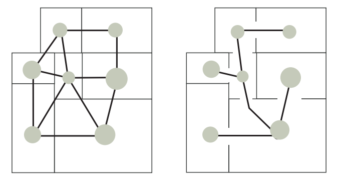
Aspect Ratio

Rooms must be within a proper range of aspect ratio in order to be comfortable and easy to use (Alexander, 1997). The aspect ratio is calculated as the shortest side divided by the longest side. Ratio close to 1 means equally long sides.



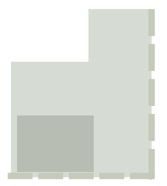
Shape Flexibility

The flexibility is based on the standard deviation of the areas in the data set. Each room type is assigned a weight between 0 and 1 according its flexibility. A higher value means higher probability for the space to adjust its size.



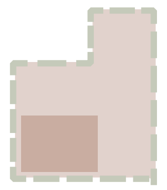
Adjacency and Circulation

Adjacencies (each room connection to other rooms) are studies and extracted from the data set as well as circulation. The connections are presented in a adjacency matrix in the next section.



Facade connection

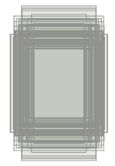
The facade connection tells whether a room is likely to be placed next to the facade or not. Some rooms are always placed close to the facade due to daylight regulations but some are placed more or less likely.



Boundary connection

The boundary connection tells whether a room is likely to be placed next to the boundary or not. Some rooms are always placed close to the boundary but sometimes smaller rooms are placed like islands in bigger apartments.

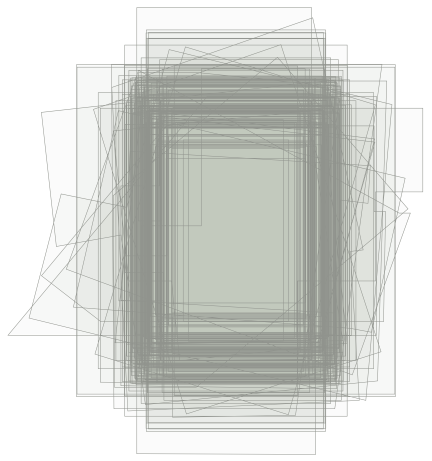
SPACE STATISTICS



HALL

n = 525

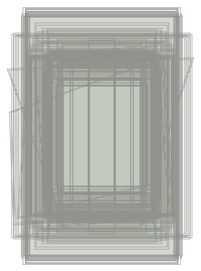
Average Area	5.2 m2
Area SD	1.67 m2
Flexibility	0.13
Aspect ratio	0.86



LIVING ROOM

n = 525

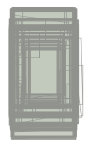
Average Area	23 m2
Area SD	8.91 m2
Flexibility	1
Aspect ratio	0.88



KITCHEN

n = 525

Average Area	11 m2
Area SD	5.67 m2
Flexibility	0.6
Aspect ratio	0.89



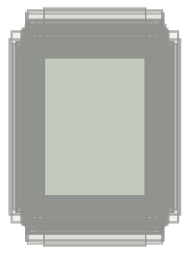
BATHROOMS

n = 713

Average Area	4.67 m2
Area SD	1.51 m2
Flexibility	0.11
Aspect ratio	0.83

Skala 1:200

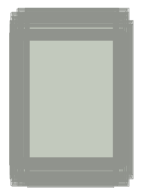
SPACE STATISTICS



DOUBLE BEDROOM

n = 492

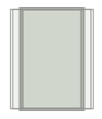
Average Area	13 m2
Area SD	1.5 m2
Flexibility	0.11
Aspect ratio	0.8



SINGLE BEDROOMS

n = 826

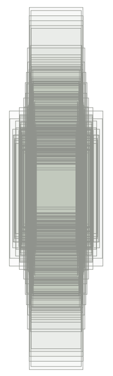
Average Area	8 m2
Area SD	1.1 m2
Flexibility	0.06
Aspect ratio	0.72



STUDIO BED SPACE

n = 18

Average Area	5.1 m2
Area SD	0.6m2
Flexibility	0.0
Aspect ratio	0.75



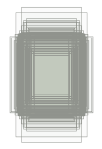
PASSAGE

n = 713

Average Area	4 m2
Area SD	2.83 m2
Flexibility	0.3
Aspect ratio	0.15

Skala 1:200

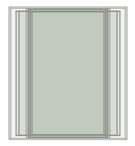
SPACE STATISTICS



CLOSET

n = 289

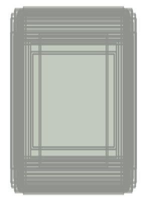
Average Area	3.2 m2
Area SD	1.53 m2
Flexibility	0.11
Aspect ratio	0.93



STUDY

n = 23

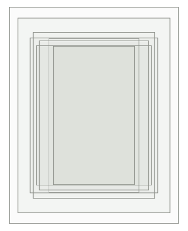
Average Area	8 m2
Area SD	1.1 m2
Flexibility	0.06
Aspect ratio	0.88



DINING

n = 273

Average Area	11 m2
Area SD	2 m2
Flexibility	0.16
Aspect ratio	0.83



LIBRARY

n = 12

Avg Area	15.1 m2
Area SD	5.9 m2
Flexibility	0.63
Avg aspect ratio	0.72

ADJACENCY & CIRCULATION

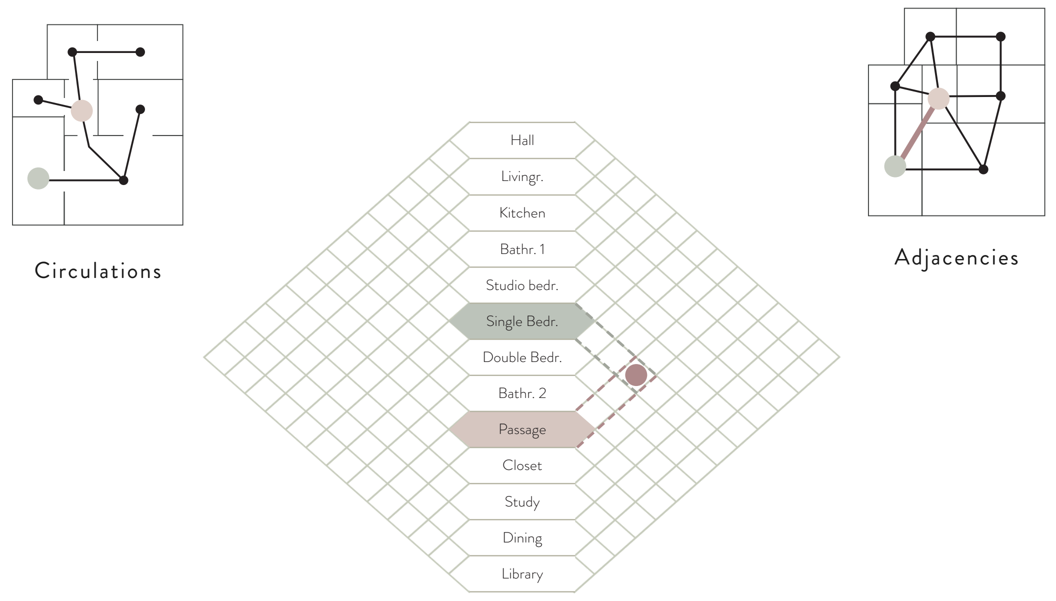
Spaces of the same type have similar shapes, areas and ratios and these parameters can easily be decoded. However, the size of a room can be impacted by the size of other rooms. A larger kitchen can for example result in a smaller living room, and in an open floor plan it is also not trivial where the border between kitchen-living room should be drawn. Such relationships are difficult to cover and the variations within the same room type can also result in less precise input parameters in terms of area and ratio.

The data set represents general trends and should be seen as a proof of concept for the procedural algorithm's input. However, what can be seen from the rooms presented on the three previous sides is that the spaces within the same type have more or less fixed sizes and dimensions. For a generative algorithm some parameters need to vary in order for the code to iterate over possible solutions.

The proximity of rooms to one another is a key dimension of a floor plan (Chaillous, 2019). Therefore the relationships between spaces (on an inter-space level) is presented in the following section. Connections in terms of adjacencies and circulations are presented in a matrix(see fig 4) with the circulation connections to the left and the adjacencies to the right.

This is done for different apartment sizes and for all the apartments together. The amount of appeared connections among a certain apartment type is divided by the total amount of apartments in the set. This gives a percentage of how often the connection between two rooms occur.

Example & Reading Instructions



STUDIO APARTMENTS

30-46 m2, n: 28

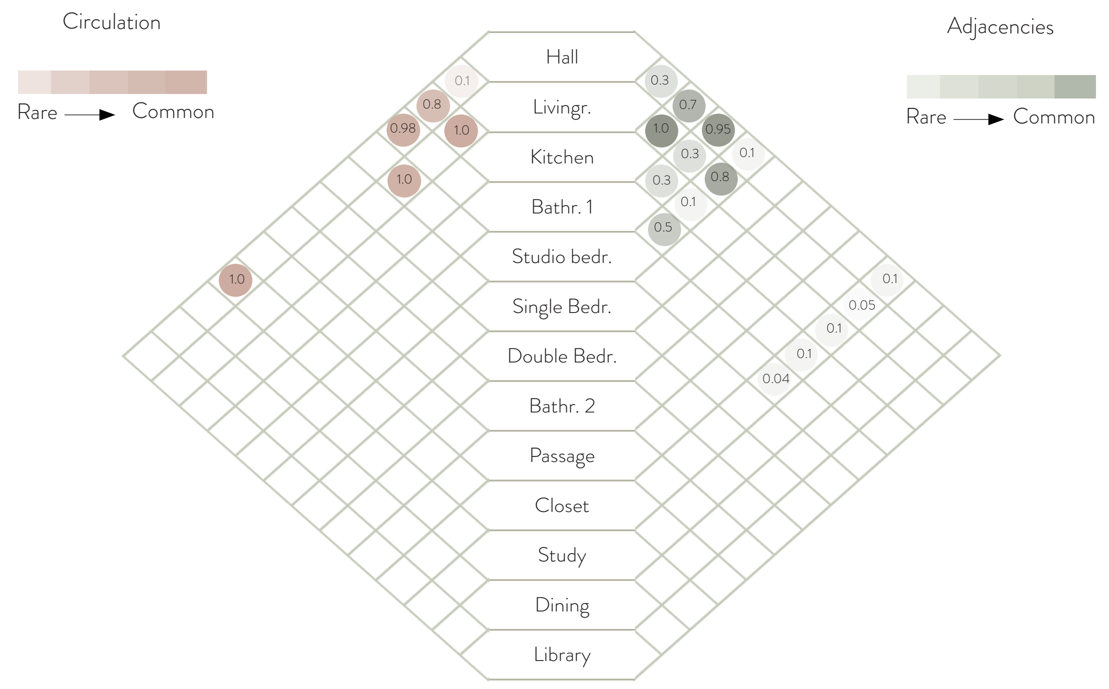
SPACE STATISTICS

	Area	Ratio	Boundary	Daylight	Facad	Type*	Presence
Hall	5	0.84	1.0	N	0.0	C	1.0
Living-room	17	0.93	1.0	Y	1.0	S	1.0
Kitchen	7	0.75	1.0	Y	0.57	S	1.0
Bathroom 1	4.2	0.63	1.0	N	0.0	P	1.0
Studio bed space	5.1	0.75	1.0	Y	0.46	P	1.0
Single Bedroom							
Double Bedroom							
Bathroom 2							
Passage							
Closet	1.3	0.9	1.0	N	0.0	P	0.14
Study room							
Dining							
Library							

*Types

C = Communication
S = Social
P = Private

INTER-SPACE STATISTICS



1 BEDROOM APARTMENTS

35-72 m2, n: 172

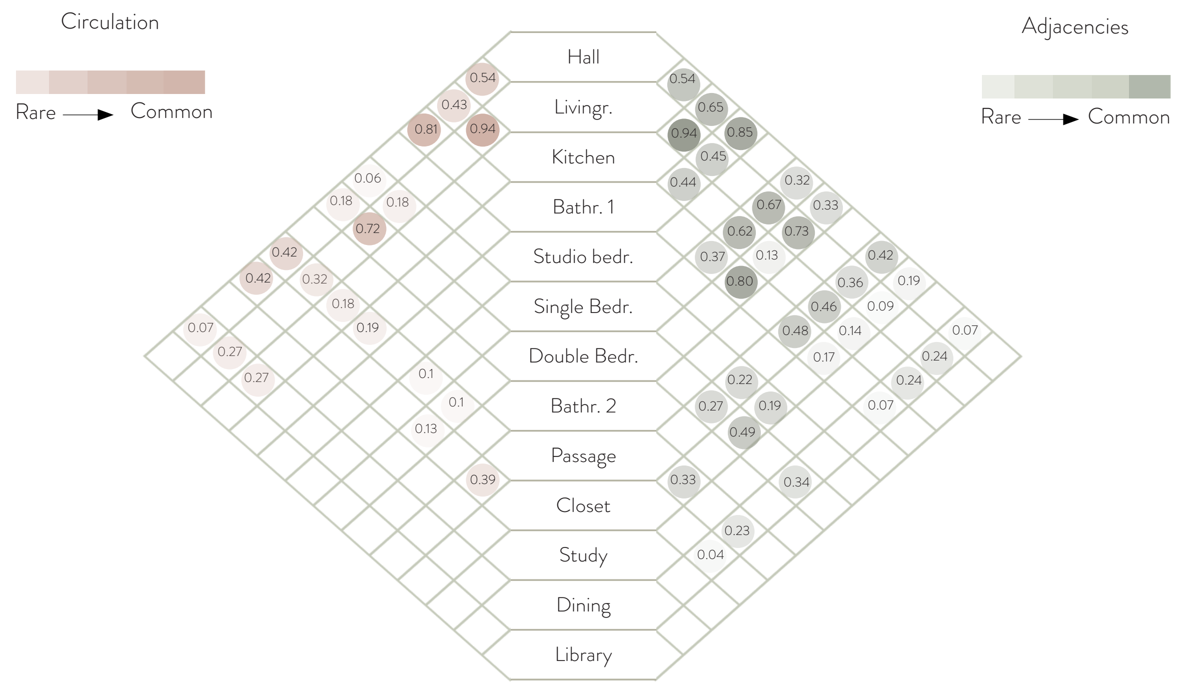
SPACE STATISTICS

	Area	Ratio	Boundary	Daylight	Facad	Type*	Presence
Hall	5	0.82	1.0	N	0.0	C	1.0
Living-room	21	0.87	1.0	Y	1.0	S	1.0
Kitchen	11	0.86	1.0	Y	0.57	S	1.0
Bathroom 1	4.2	0.63	1.0	N	0.0	P	1.0
Studio bed space							
Single Bedroom	8	0.72	1.0	Y	1.0	P	0.1
Double Bedroom	12	0.8	1.0	Y	1.0	P	0.9
Bathroom 2							
Passage	3	0.42	0.017	N	0.0	C	0.23
Closet	2.1	0.9	1.0	N	0.0	P	0.25
Study room							
Dining	8	0.83	1.0	Y	1.0	S	0.44
Library							

*Types

C = Communication
S = Social
P = Private

INTER-SPACE STATISTICS



2-3 BEDROOM APARTMENTS

61-147 m2, n: 303

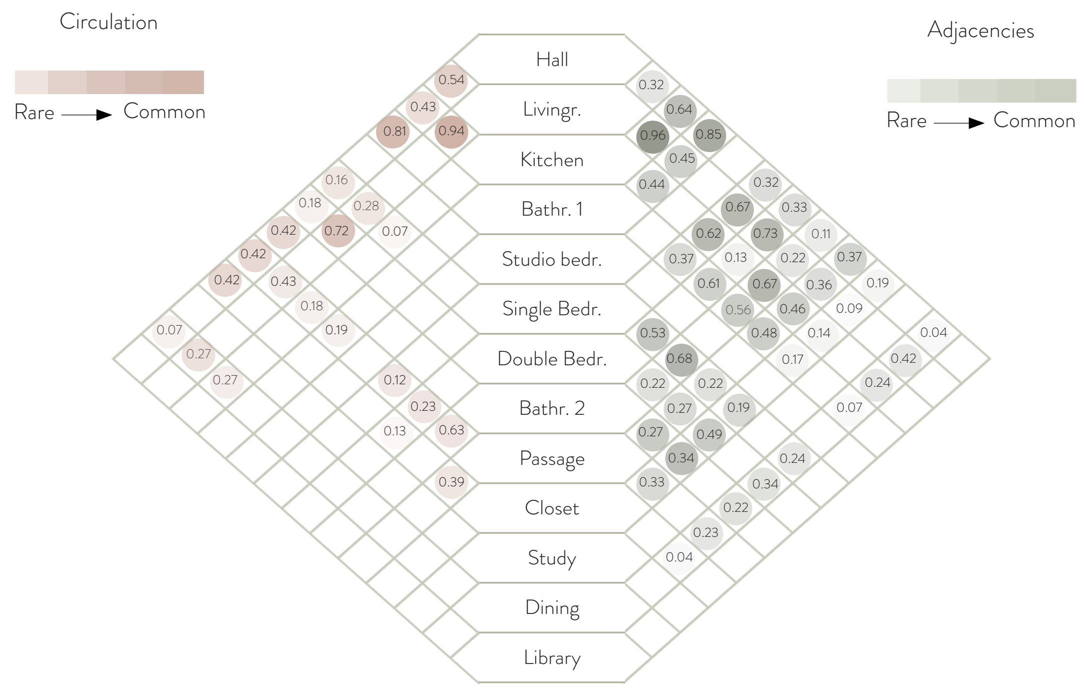
SPACE STATISTICS

	Area	Ratio	Boundary	Daylight	Facad	Type	Presence
Hall	5	0.86	1.0	N	0.0	C	1.0
Living-room	21	0.88	1.0	Y	1.0	S	1.0
Kitchen	11	0.89	1.0	Y	0.49	S	1.0
Bathroom 1	4.2	0.63	1.0	N	0.03	P	1.0
Studio bed space							
Single Bedroom	8	0.74	1.0	Y	1.0	P	1.0
Double Bedroom	13	0.81	1.0	Y	1.0	P	1.0
Bathroom 2	2	0.89	0.5	N	0.0	P	0.14
Passage	7	0.38	0.017	N	0.02	C	0.43
Closet	2.1	0.88	1.0	N	0.06	P	0.45
Study room							
Dining	8	0.83	1.0	Y	1.0	S	0.44
Library							

*Types

C = Communication
S = Social
P = Private

INTER-SPACE STATISTICS



4 BEDROOM APARTMENTS

98-155 m2, n: 18

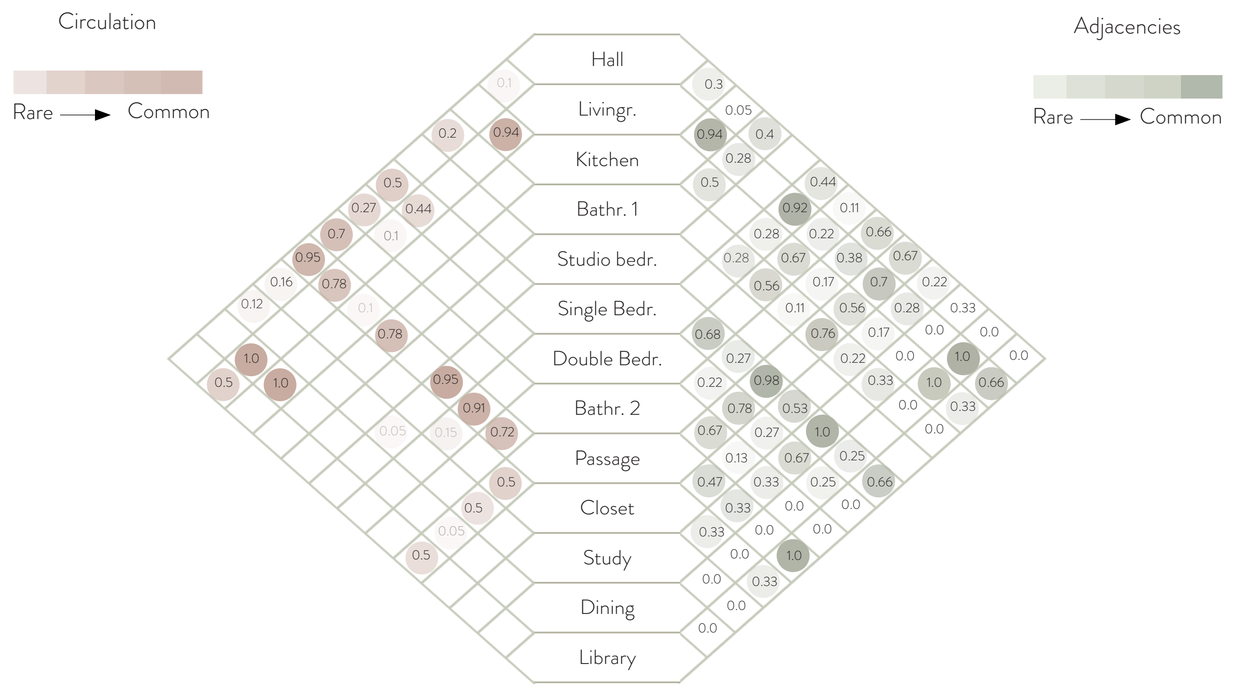
SPACE STATISTICS

	Area	Ratio	Boundary	Daylight	Facad	Type	Presence
Hall	4	0.83	1.0	N	0.0	C	1.0
Living-room	24	0.85	1.0	Y	1.0	S	1.0
Kitchen	14	0.85	1.0	Y	0.83	S	1.0
Bathroom 1	5.8	0.83	0.78	N	0.16	P	1.0
Studio bed space							
Single Bedroom	8	0.71	1.0	Y	1.0	P	2.66
Double Bedroom	13.5	0.8	1.0	Y	1.0	P	1.05
Bathroom 2	2	0.9	0.67	N	0.056	P	1.0
Passage	5	0.15	0.11	N	0.0	C	1.5
Closet	2	0.93	0.84	N	0.2	P	0.83
Study room	5	0.88	1.0	N	1.0	P	0.17
Dining	11	0.83	1.0	Y	1.0	S	0.42
Library	15.1	0.72	1.0	Y	1.0	S	0.17

*Types

C = Communication
S = Social
P = Private

INTER-SPACE STATISTICS



ALL APARTMENTS

30-155 m2, n: 525

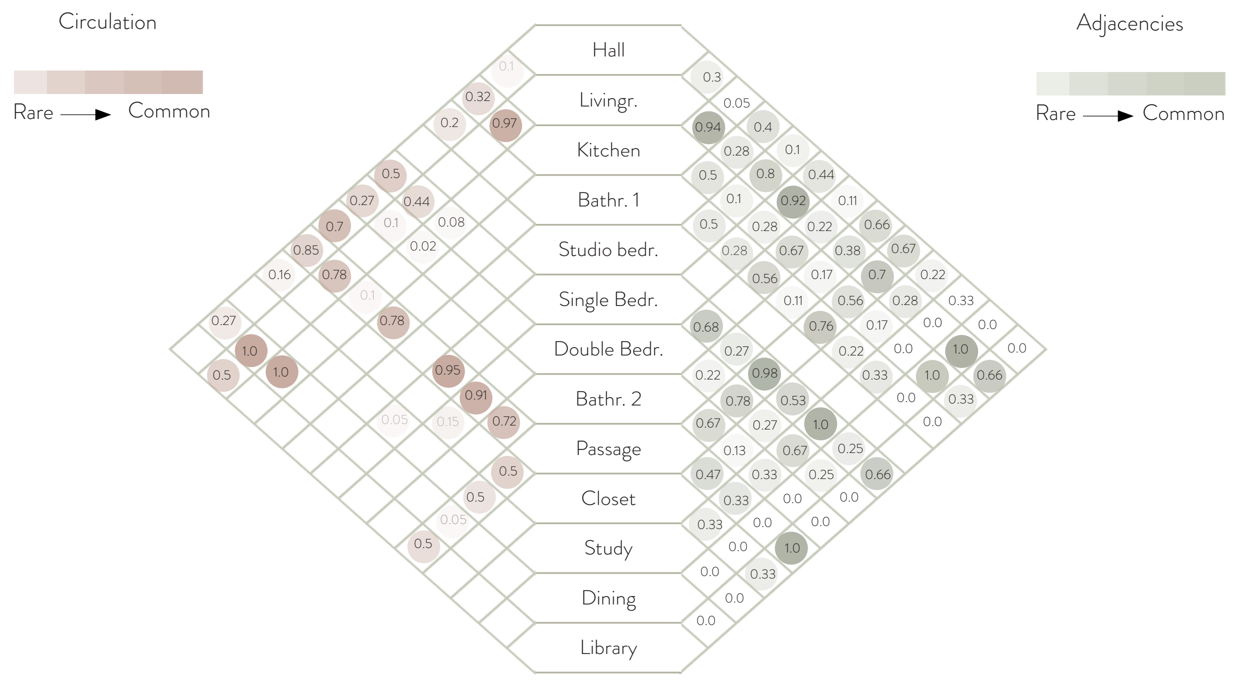
SPACE STATISTICS

	Area	Ratio	Boundary	Daylight	Facad	Type	Presence
Hall	5.2	0.86	1.0	N	0.0	C	1.0
Living-room	23	0.88	1.0	Y	1.0	S	1.0
Kitchen	11	0.89	1.0	Y	0.83	S	1.0
Bathroom 1	4.67	0.83	0.78	N	0.16	P	1.0
Studio bed space	5.1	0.75	1.0	Y	0.46	P	0.03
Single Bedroom	8	0.72	1.0	Y	1.0	P	1.57
Double Bedroom	13	0.8	1.0	Y	1.0	P	0.93
Bathroom 2	2	0.9	0.67	N	0.056	P	0.34
Passage	4	0.15	0.21	N	0.0	C	1.36
Closet	3.2	0.93	0.84	N	0.2	P	0.55
Study room	8	0.88	1.0	N	1.0	P	0.04
Dining	11	0.83	1.0	Y	1.0	S	0.52
Library	15.1	0.72	1.0	Y	1.0	S	0.023

*Types

C = Communication
S = Social
P = Private

INTER-SPACE STATISTICS



CONCLUSIONS

1

Boundary connections

In most of the cases the rooms inside an apartment is adjacent to the boundary. Only a few bathrooms, closest and passages in big apartments are not connected to the boundary.

2

Area and Ratio

Rooms of the same type have similar areas and ratios. The variation is small among most of the room types. However, the room sizes and ratios can vary within a given range. The ratios varies between 0.8 and 0.9.

3

Flexibility

Based on the area standard deviation value the rooms are assigned a value between 0 and 1. This value indicates how probable it is for the room to be adjusted in order to create a water tight layout. It is for example more likely that the living room is adjusted than the bathroom.

4

Adjacencies

All adjacency connections between two spaces are allowed. However, some are more common than others, like living room - kitchen and living room - dining. For an algorithm with a fixed boundary as input a few adjacency preferences will be used as inputs, but the rest will be treated as outputs from the algorithm.

5

Circulation

Clear trends are seen when it comes to circulation. For small apartments most of the connections are via the hall and the living room. For big apartments most of the rooms are connected via passages/inner halls.

6

Passages

The passage stands out from the rest of the spaces both in term of flexibility and amount of connected rooms. The passage occurs as consequent of the other space placements. In the following chapter the passage will therefore not be treated as a room, but a corridor connecting rooms.

04 PROTOTYPE

APPROACH

The challenge of designing floor plans is, as mentioned in previous chapters, primarily a challenge of designing appropriate locations and areas for the rooms.

Based on the literature study regarding related work one can conclude that almost all of the references are from the gaming industry. There is a lacking amount of references from the architectural discipline and the results from the algorithms used in the gaming industry differ a bit from real architectural floor plans.

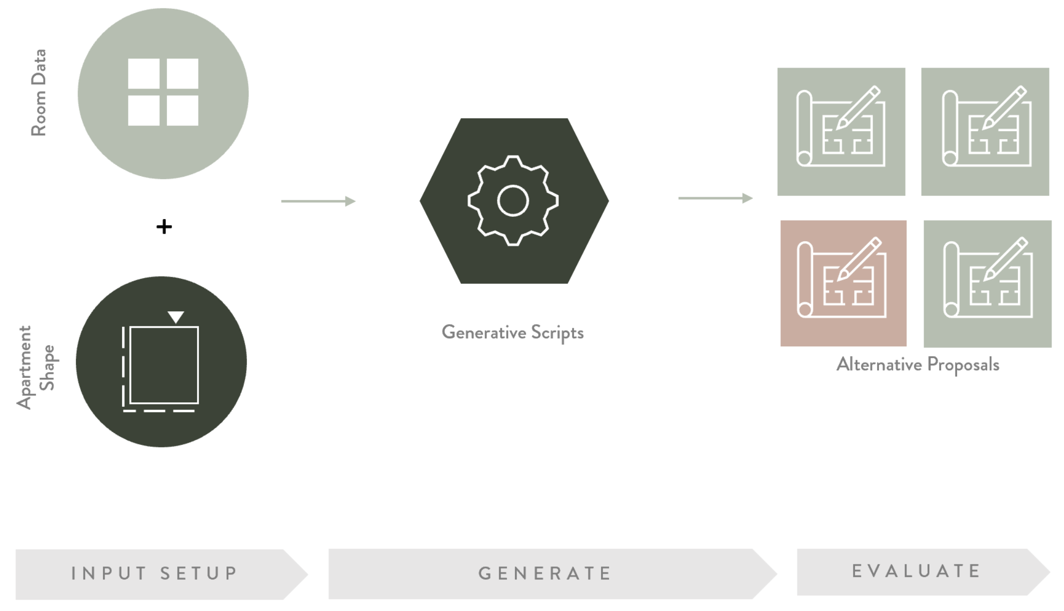
What most of the algorithms do not take into consideration is internal room requirements such as daylight etc. Most of the related algorithms are also applied on single family housing which means the facade is equal to the boundary of the whole floor plan.

In order to create a valid floor plan for an apartment in a multi family house a facade based approach, written in C#, is introduced in the following sections of this thesis. The intention is to emulate real life architectural floor plans, independent of style or type of buildings.

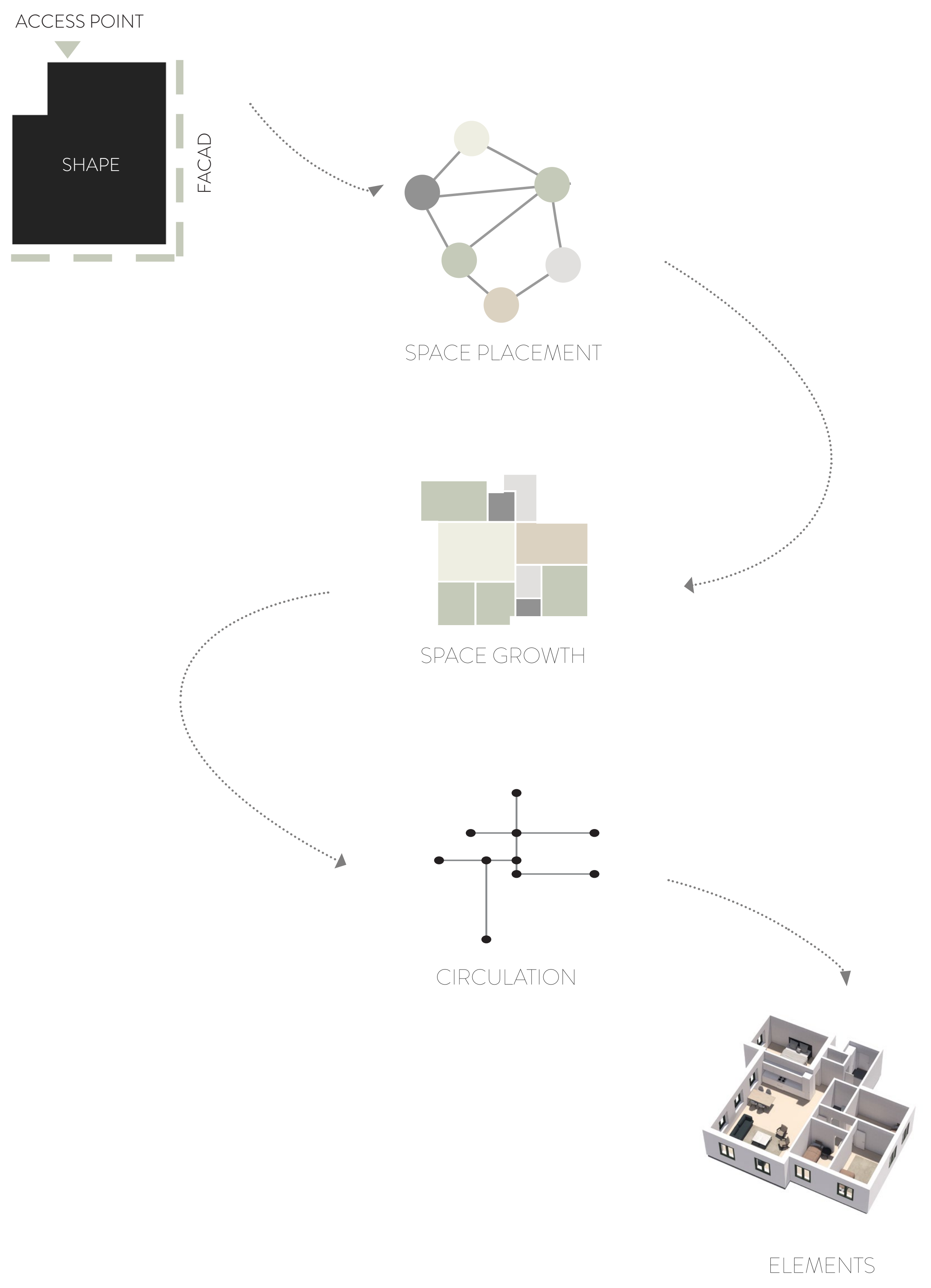
Inputs
The user input for the algorithm is the boundary of the apartment, the facade, an access point and a list of required rooms. The boundary is represented by a poly line in the xy plane and the access point is a point on the boundary represented by an x and y coordinate. The room list represents a list of objects with certain properties presented in the previous chapter.

Outputs
The output is N amount of alternative floor plan layouts (with different adjacency graphs/ room placements). The algorithm and the outputs should not be seen as the final result of a floor plan but rather as an investigation and inspiration for further design development.

An overview of the algorithm is presented in figure 2.1 and the implementation steps are described in the following sections.



ALGORITHM OVERVIEW



SPACE PLACEMENT

The first phase in the algorithm is the space placement. An underlying grid is created within the user defined boundary of the apartment and works as the basis for room placement (1).

Based on adjacency preferences, room areas and required facade connections, the initial positions of the rooms are determined.

As presented in the previous chapter most of the rooms in an apartment are placed next to the boundary. The cells within a given distance away from the boundary get higher weights and the cells closer to the middle get lower weights. It is more likely that a cell with a higher weight is selected for the initial position for a space and therefore the spaces are more likely to be placed next to the boundary.

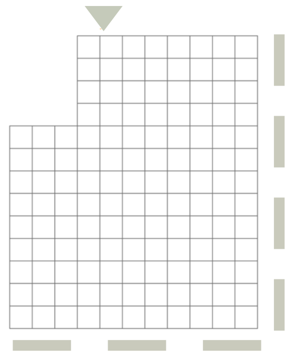
The placement does always start from the access point and the first room to be places is the hall. The weight of the cells close to the access point therefore get altered and the possibility to select one of these cells as the initial position for the entrance increases (2).

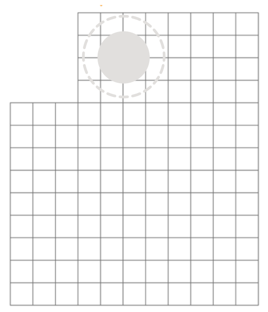
Which space to be placed after the hall depends on adjacency preferences and facade positions. (3)

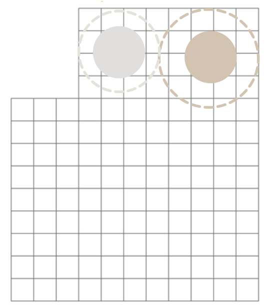
As shown in the previous chapter most of the adjacency connections between rooms are possible. The order for the rooms to be placed is therefore dependent on a random seed, but also user specific adjacency preferences, such as kitchen - living-room.

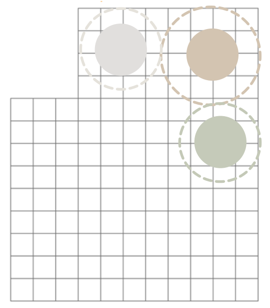
The algorithm start so place rooms along the facade (4 to 6) and then it then continues with the remaining part for spaces witouth daylight requirements (7).

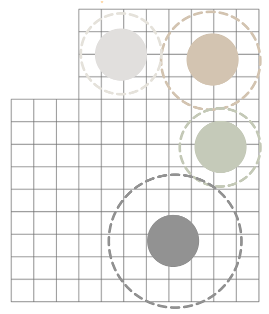
The algorithm continues until all the rooms are placed and no possible cells for initial positions are left (8). Then the grid is removed and the room centre points can be represented in an adjacency graph (9).

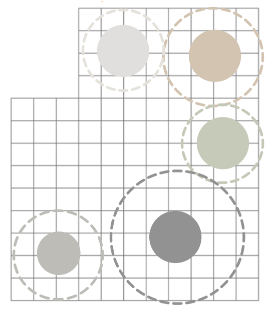

(1) Inputs and Grid creation

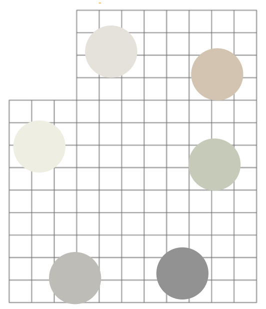

(2) Hall placement next to the access point.

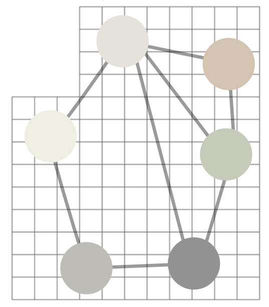

(3) Placement of first adjacent room

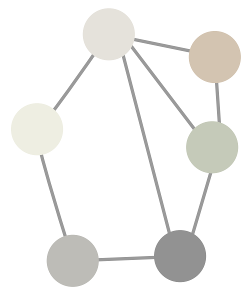

(4) Area occupation and room placement


(5) Area occupation and room placement


(6) Area occupation and room placement


(7) Area occupation and room placement


(8) Final room positions


(9) Remove grid

SPACE GROWTH

Rectangular spatial growth
The growth method starts with a cell containing the initial position of each room and the algorithm (1) then procedurally makes the rooms grow. One room is picked at the time and the algorithm makes the room grow by appending adjacent cells to it (2 to 5). This continues until the rooms have reached their required areas or until no more rectangular expansions are possible (6).

The room selected to grow in an iteration is the room with its current area far away from the required area. In order to achieve plausible room shapes the algorithm aims for as rectangular rooms as possible.

Of course this phase of the algorithm does not ensure that all of the cells and available space gets assigned to a room. Therefore the algorithm continues with the next step where the expanded rooms are considered for further expansion but according to a set of new rules.

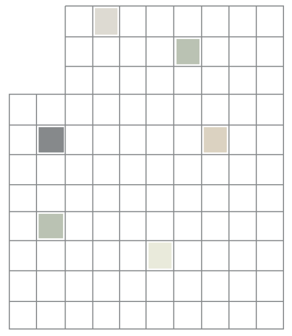
L-shape spatial growth
In this phase of the process the rooms are allowed to grow in another way. Similar to the approach Lopes presented 2010 the algorithm now considers expansion that results in bigger rooms and L-shaped expansions (7). The area is no longer taken into consideration but the flexibility constant indicates which rooms that should be selected and adjusted.

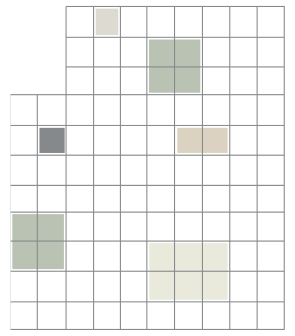
As mentioned in previous chapters different rooms have different probabilities to be adjusted. A living room is not as fixed in shape as a bathroom for example. Therefore the room property “size flexibility” tells how probable it is for a room to be adjusted in this phase. The living room for example has a higher value of this constant then the bathroom and it is therefore more likely for the living room to grow as an L-shape and fill up remaining space in the layout than the bathroom (9).

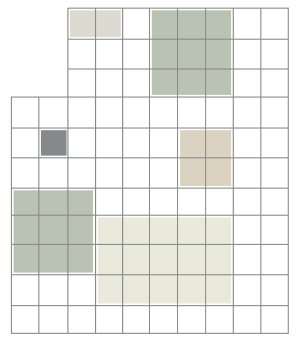
In order to achieve efficient room shapes and avoid narrow striped or U-shaped rooms the longest side in each room is considered. If the living room with let’s say dimensions of 5 x 4 meter is considered to grow the longest side is selected at first and a percentage of the side length is set as a tolerance. If the tolerance is 50% the algorithm only allows the room to grow if more than 2.5 m of the longest side of 5 meters has non occupied cells next to it.

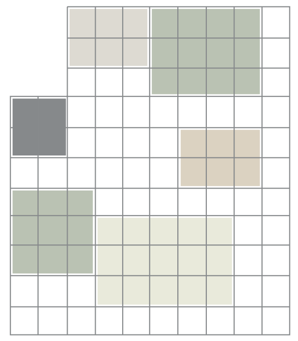
The final step in the algorithm is to fill gaps. The algorithm searches for small empty cells in the underlying grid and appends them to an adjacent room. Once all of the cells in the defined grid is occupied by a room the algorithm has reached it goal and a floor plan layout is created. Yet doors and thickness of inner walls are not assigned and generated.

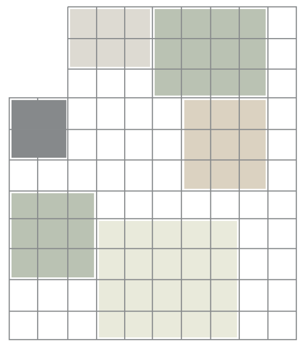
The grid
A grid with small cell dimensions allows better control for the room expansion process, but decreases the chance of snapping to lines extended from exterior features (discontinuity points representing corners on the boundary). The grid size is therefore a variable which must be adjusted to balance the control of the expansion and the snapping. In this prototype the grid size was empirically determined to 0.6 m as a good balance between these two aspects, but the input value can be changed if necessary.

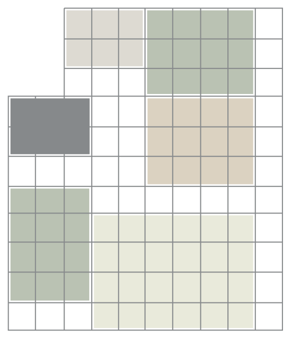

(1) Grid creation and initial space position (from space placement)

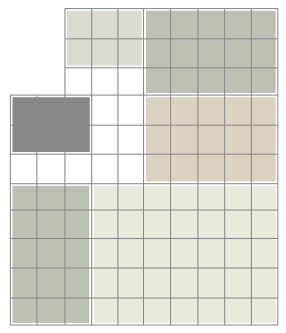

(2) Space expansion. Living-room and bedrooms

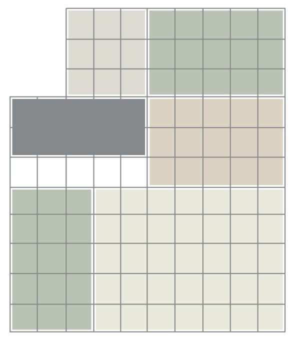

(3) Space expansion. Living-room, bedrooms and kitchen

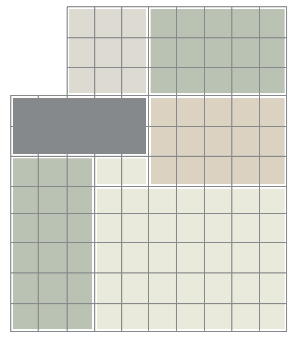

(4) Space expansion


(5) Space expansion


(6) The spaces have reached their prefered areas


(7) Expand flexible rooms


(8) Space shape adjustments. Expand flexible rooms


(9) L-shape expansion (for the living-room)

CIRCULATION

The circulation, or walk-able interconnections between rooms, are considered as a post processing part of the layout creation. It is no longer appropriate to use the underlying grid with cells. What is instead used as a starting point is the inner walls, defined circulation constraints from chapter 3 and the previously defined layout (1).

Musse and Marson presented a method for generating corridors based on the “A*”-algorithm for the shortest path. In order to apply this algorithm all inner paths are extracted (2) and represented as edges/ segments in a graph connected through nodes (3). As a demarcation the outer walls are excluded for the path creation, but there is a possibility to place corridors next to the boundary in real floor plans. This is discussed more in the ”Future Work” section in the last chapter.

The rooms without access from public rooms, as described in chaprter 3, are then selected. Possible edges shared by non connected private rooms are used for creating passages, or inner halls (4 and 5).

The selected edges/segments are then extruded in 2D in order to create a polygon representing the spatial boundary of the passages (6). The passage is then added to the initial floor plan layout (7).

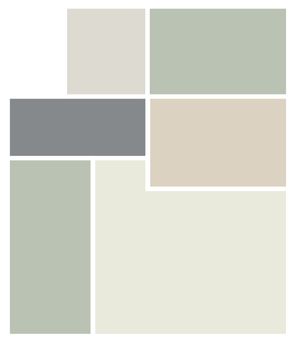
Wall creation
Once the spaces are placed and the layout is created the wall creation can be considered. Private rooms always has walls and in most cases it is enough to create walls along their room boundaries. When two social rooms, e.g. the living room and the kitchen, is located adjacent to each other there might be cases when walls are needed and cases when walls shouldn’t be added. The same cases can occur for communication spaces adjacent to public spaces, e.g. a passage and a living room. When and how the walls are created in this case depends on the circulation constraints defined in previous chapters.

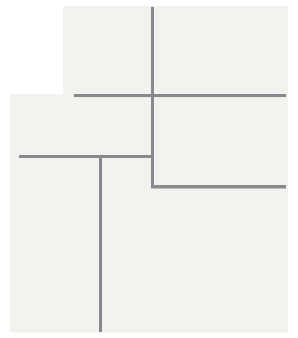
Window placement
Once the walls are created the window placements is the next step. All rooms with daylight requirements will get windows placed on the facade with a random distance (within a given domain) between the windows.

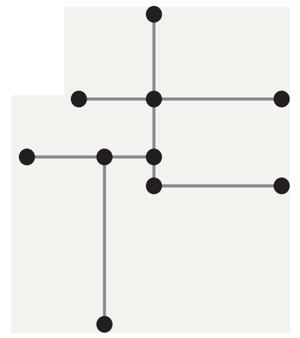
Door placement
The door placement is also dependent on the circulation constraints presented in previous chapters.

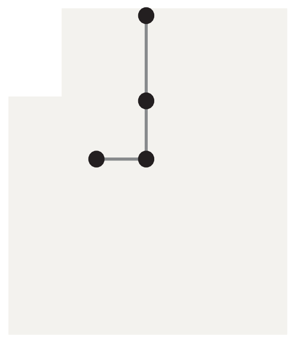
The placement follows two main principles:
(1) No placements between private rooms. The doors will always connect a private room to a social space or a connection space.
(2) Door positions will be located a certain distance from the space corners. As shown in the data set, the door placements are located close to the corners in each room. It is very unlikely for a door to be placed in the middle of a bedroom wall for example.

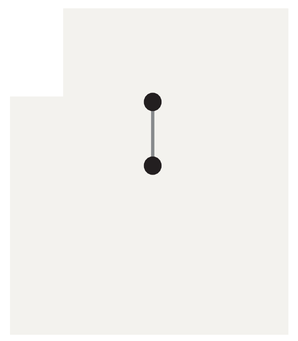
The door placement will only be represented as openings. The algorithm will only suggest position, not place real doors.

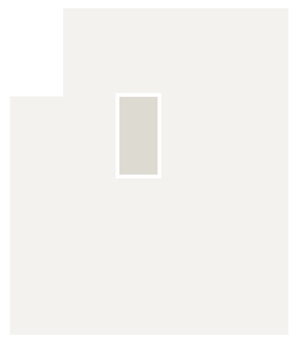

(1) Spaces

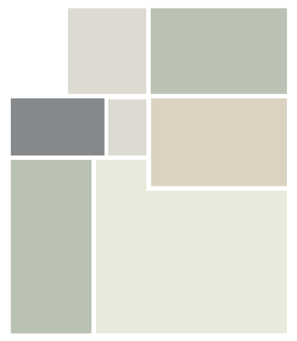

(2) Inner walls


(3) Connections and end nodes


(4) Shortest Path for connecting all the spaces


(5) Edges connected to public rooms are removed.


(6) Passage creation


(7) Final space cinfiguration

05 RESULTS

COMPARISON PROCESS

Results
The procedural algorithm is fast and multiple alternatives for a floor plan can be generated in milliseconds. Either a random selected layout can be used and further developed by an architect or all of the generated solutions can be evaluated and compared. Based on defined evaluation criteria the optimal solution for a given boundary can be selected and then further developed.

Though the algorithm wont provide a perfect final solution it would still provide early ideas of plausible floor plans. This approach would save a lot of time in the early design phases and by comparing different results of floor plans it would also result in a more data driven and evidence based decision making process.

The result shows that the algorithm can generate floor plan layouts for a wide range of different apartment shapes, independent of building style and typology. It works for apartments in lamella structures, point houses and also for complex quarter structures.

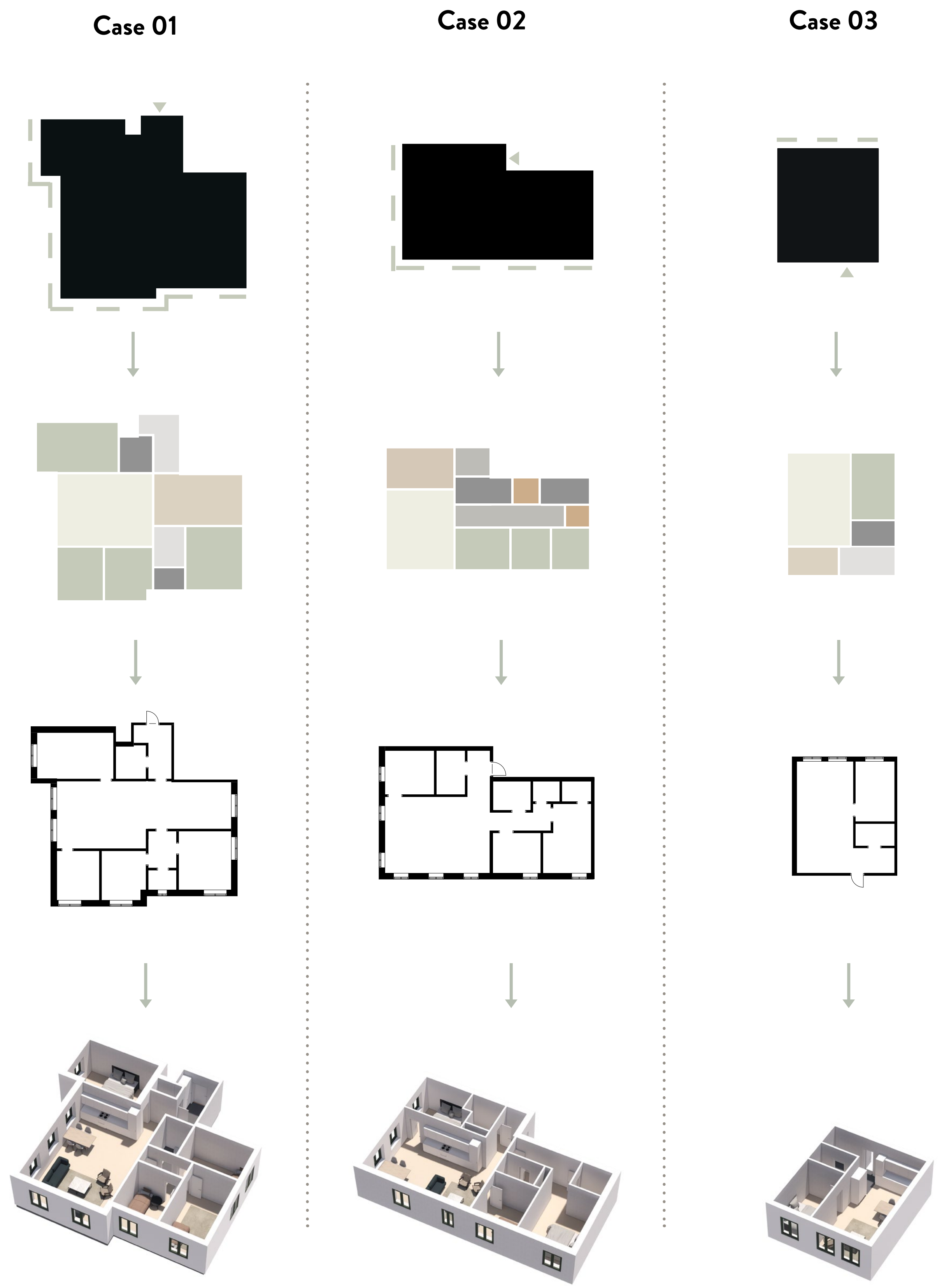
Apartment with different sizes, shapes and room topologies have been selected in order to proof the spread of usability.

Validation
The generated alternatives have been visually validated. In this chapter a couple of existing floor plans are presented together with the computationally generated ones in order to compare the results.

Comparison process
Three floor plans is selected from the reference data set. Their boundary, facade and access point (entrance door position) is recreated as well as the list of rooms they contain.

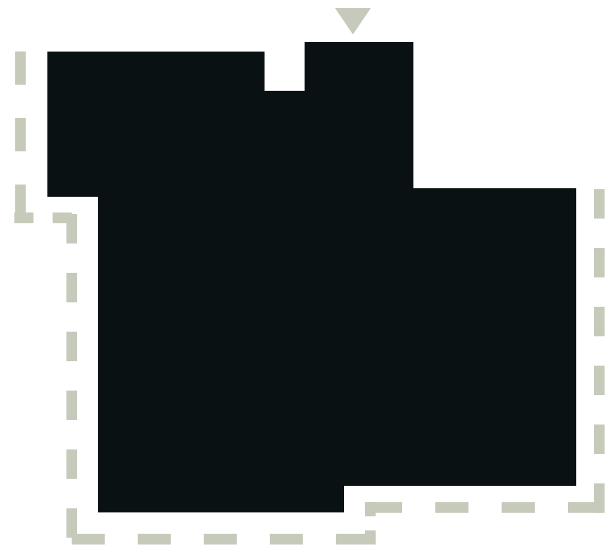
The algorithm is then applied on these cases and several alternatives for each shape is generated. Visually the outcome is evaluated and the most similar solution for each case is selected and presented next to the real architectural plan.

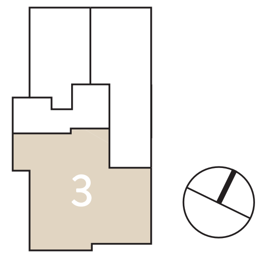
Reference Cases
The selected references are from thee different developers (Bonava, Veidekke and Skanska). Two of the selected apartments are 3 bed room apartments or bigger since it makes more sense to validate the algorithm on more complex plans in terms of size, amount of rooms and adjacency preferences. The last case is though a one bedroom apartment and the reason is not to show the variety of layouts there but rather to show that the algorithm can solve layouts for smaller apartments as well.



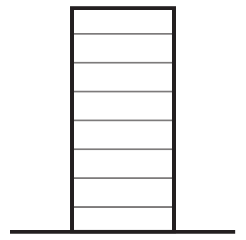
CASE 01

Inputs

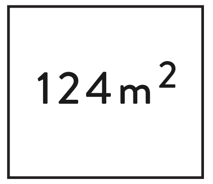




Apartment/Staircase



Typology



Apartment size

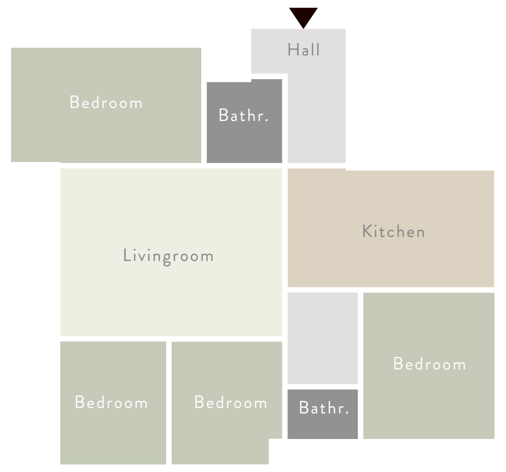
The first reference case used in the validation process is an 105 square meter big apartment in a point house typology. The reason why this shape is used is because of its irregular boundary. The niches will effect the room placement as well as the room sized and in order to proof the algorithms suitability for complex shapes this apartment is selected.

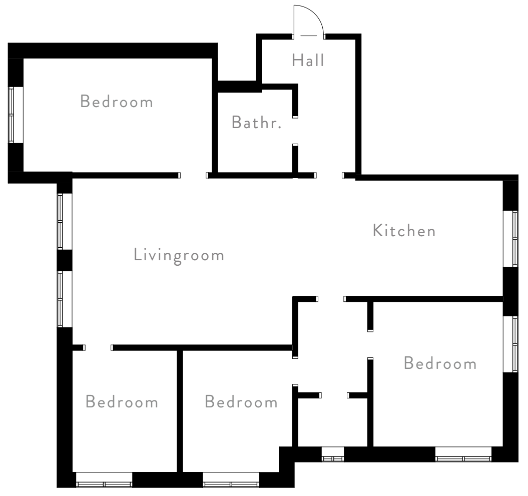
The apartment does also have a long facade which means a wide range of room configurations is possible.

On the next two pages the spatial configuration is presented. The real architectural floor plan is also presented together with the most similar generated alternative.

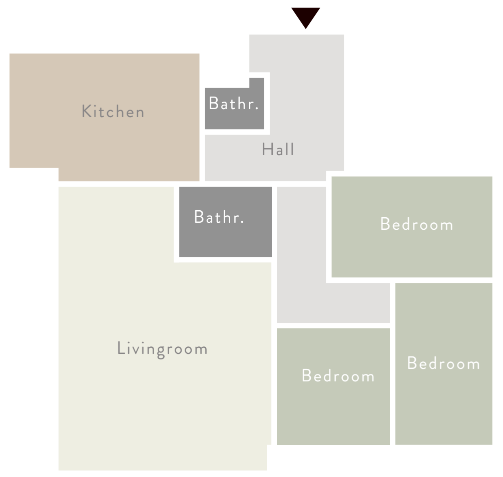
Generated layouts
Scale 1:200

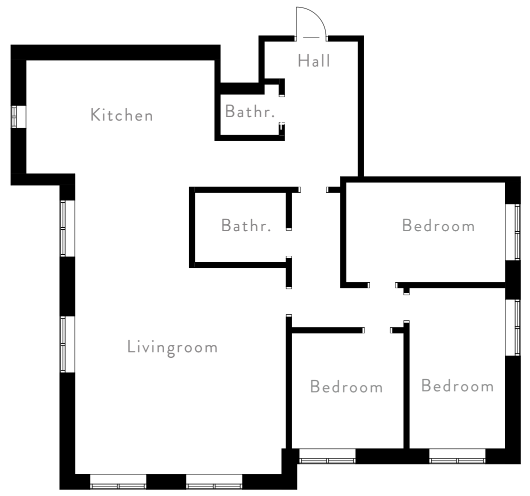
#1



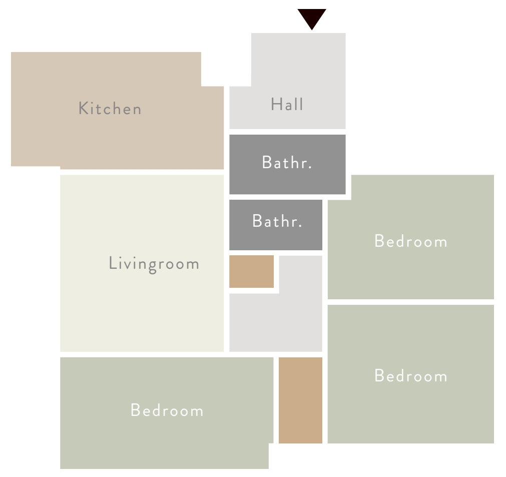


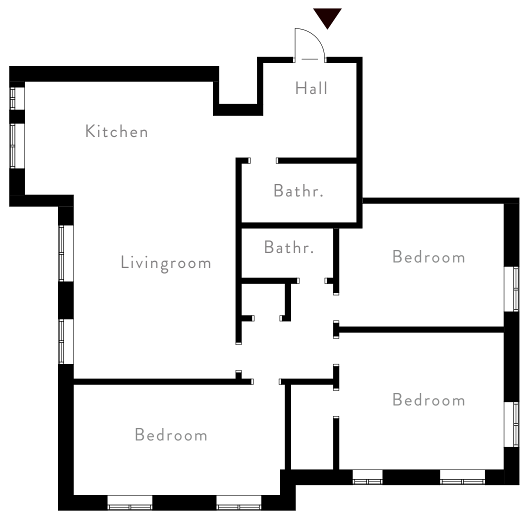
#2



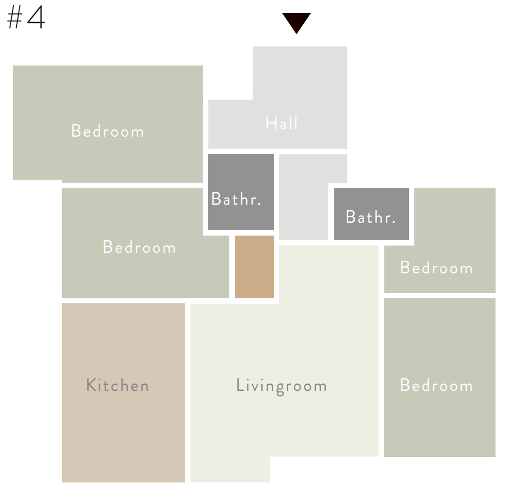


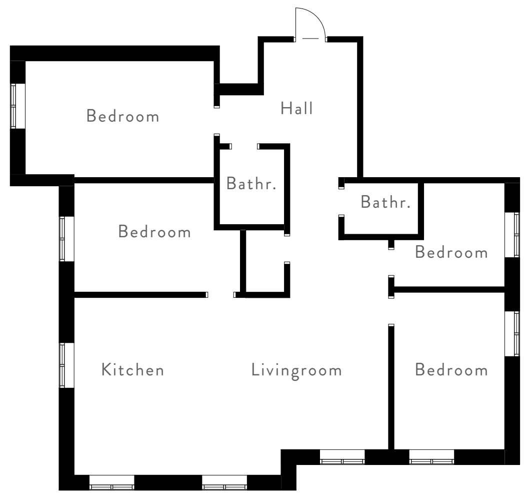
#3



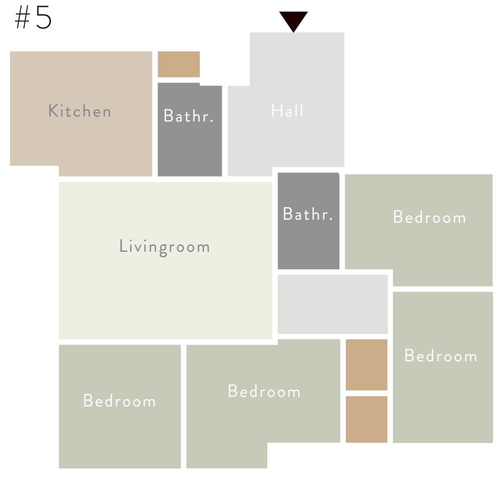


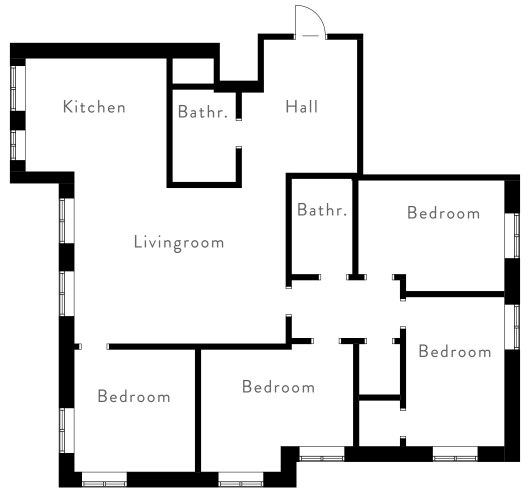
#4


Hall
Bedroom
Bathr.
Bathr.
Bedroom
Bedroom
Kitchen
Livingroom
Bedroom

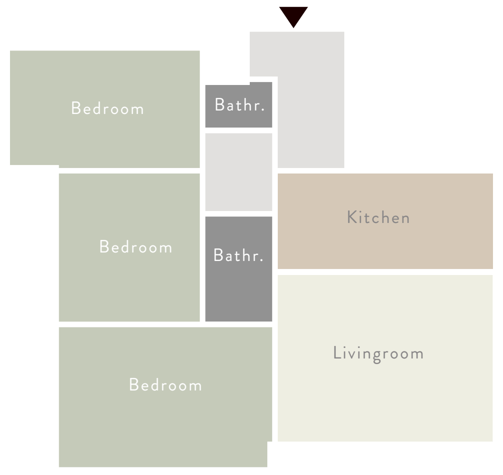

Bedroom
Hall
Bathr.
Bathr.
Bedroom
Bedroom
Kitchen
Livingroom
Bedroom

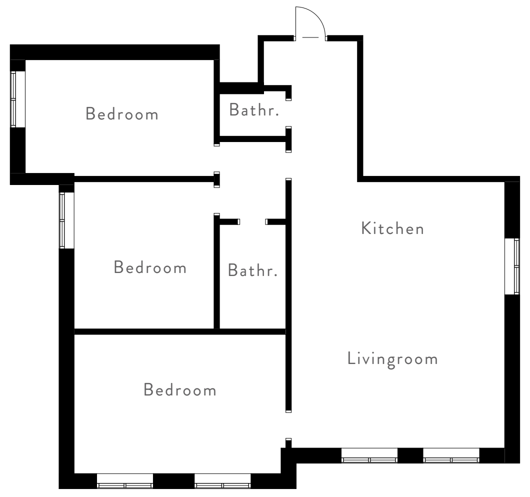
#5


Kitchen
Bathr.
Hall
Bathr.
Bedroom
Livingroom
Bedroom
Bedroom
Bedroom

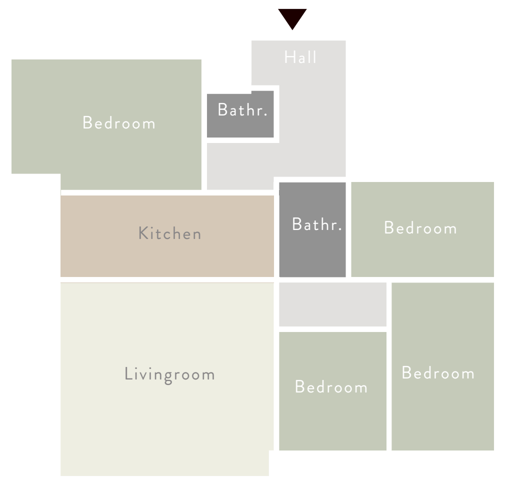

Kitchen
Bathr.
Hall
Livingroom
Bathr.
Bedroom
Bedroom
Bedroom
Bedroom

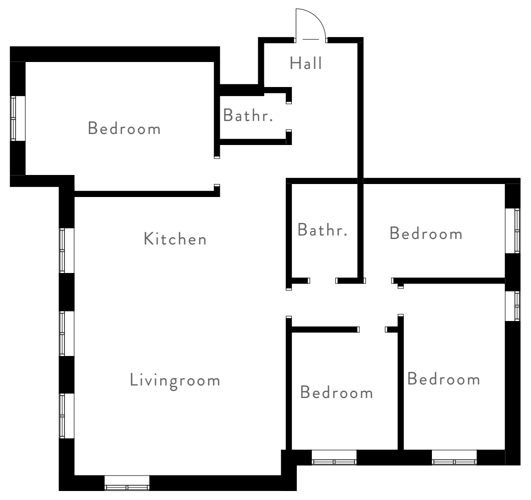
#6


Bedroom
Bathr.
Bedroom
Bathr.
Kitchen
Livingroom
Bedroom

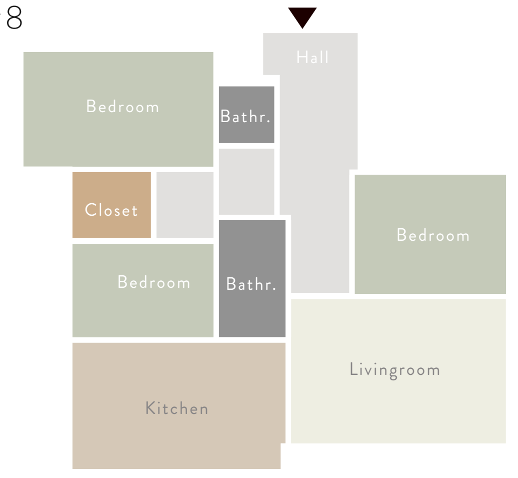

Bedroom
Bathr.
Bedroom
Bathr.
Kitchen
Livingroom
Bedroom

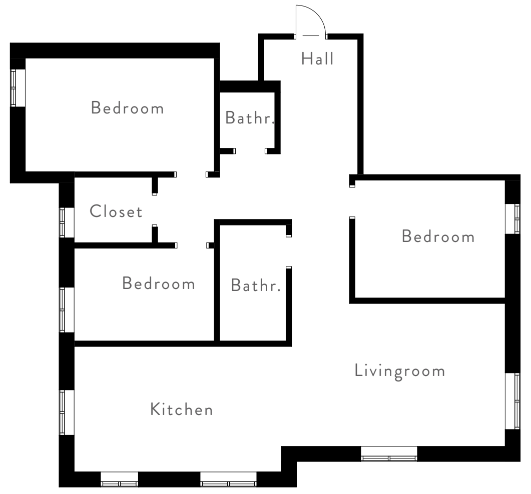
#7


Hall
Bedroom
Bathr.
Kitchen
Bathr.
Bedroom
Livingroom
Bedroom
Bedroom

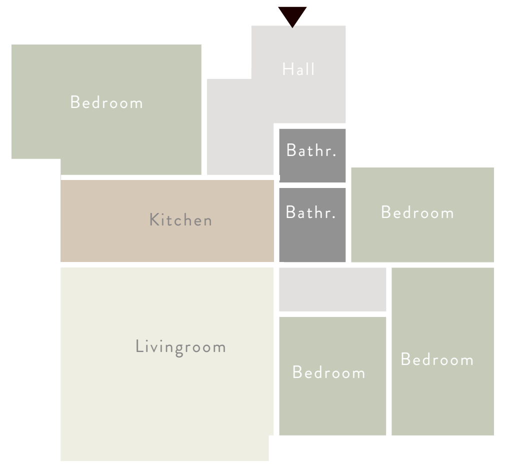

Hall
Bedroom
Bathr.
Kitchen
Bathr.
Bedroom
Livingroom
Bedroom
Bedroom

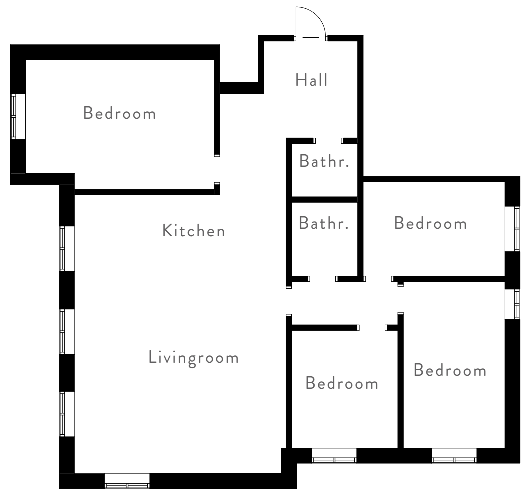
#8


Hall
Bedroom
Bathr.
Closet
Bedroom
Bedroom
Bathr.
Livingroom
Kitchen

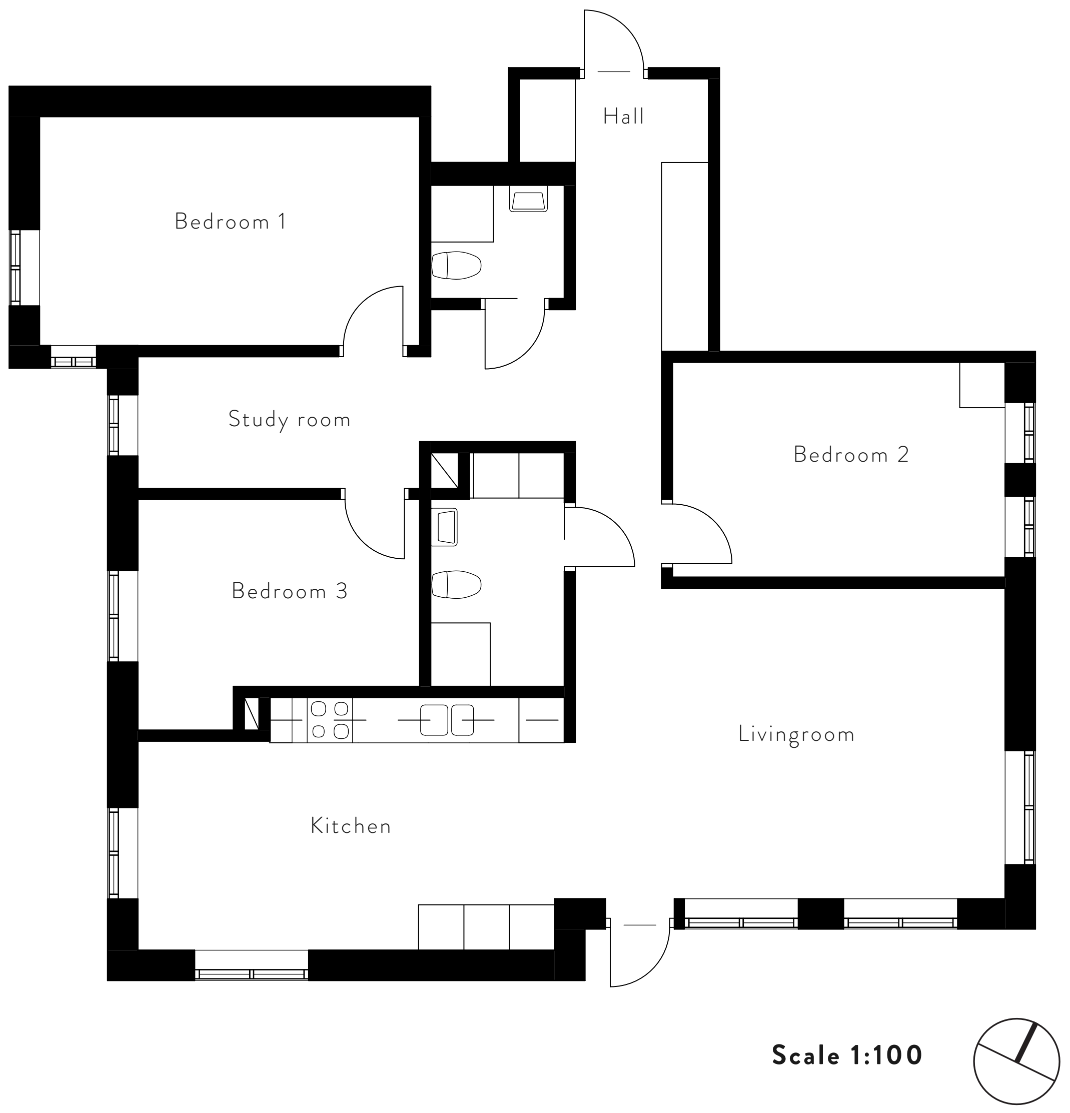

Hall
Bedroom
Bathr.
Closet
Bedroom
Bedroom
Bathr.
Livingroom
Kitchen

#9


Hall
Bedroom
Bathr.
Kitchen
Bathr.
Bedroom
Livingroom
Bedroom
Bedroom


Hall
Bedroom
Bathr.
Kitchen
Bathr.
Bedroom
Livingroom
Bedroom
Bedroom

ARCHITECTURAL FLOOR PLAN

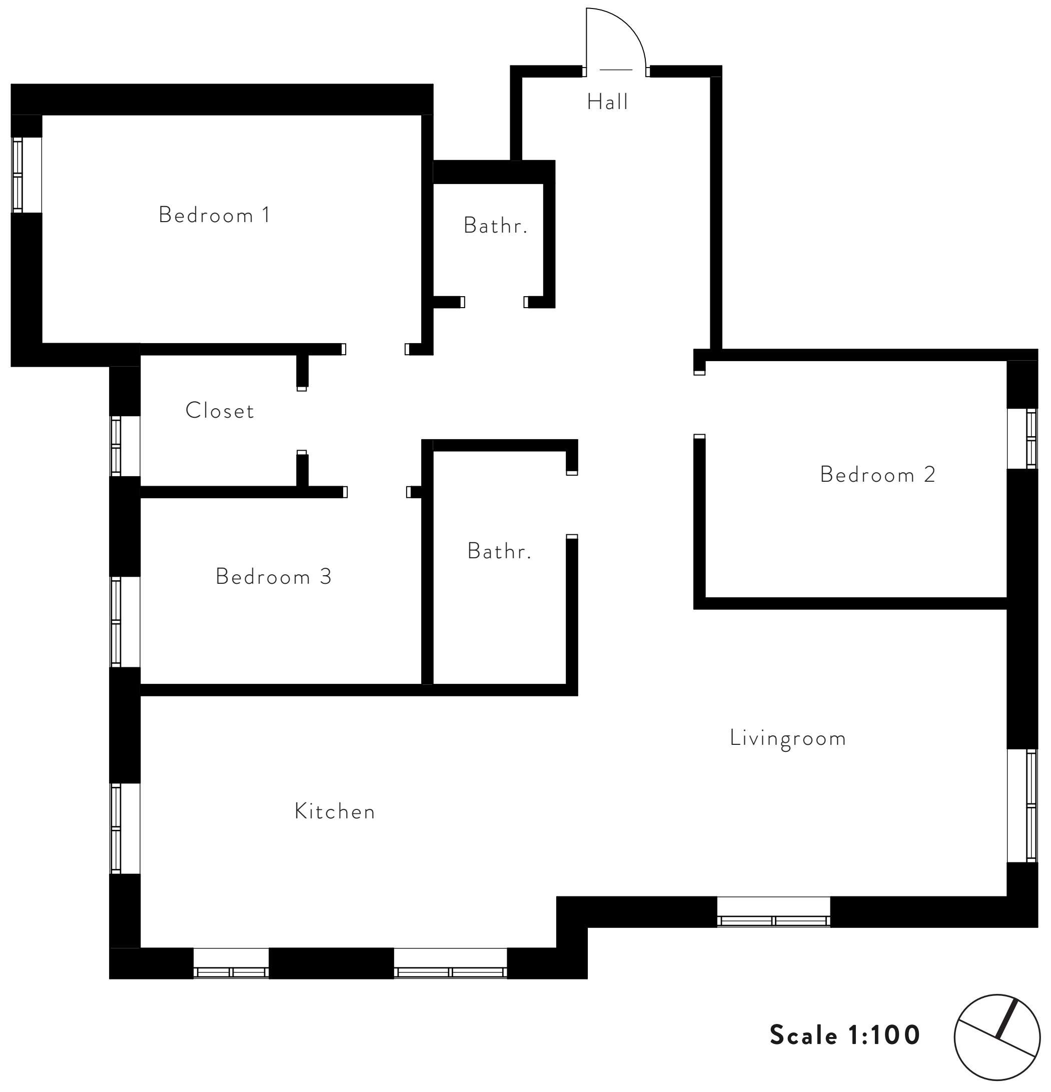


Architect: Ettelva Arkitketer
Developer: Bonava Sverige
Project: Tollare Marina

The architectural floor plan is presented above. The floor plan consists of three bedrooms, a living room, kitchen, hall, two bathrooms and a study room.

Note that the floor plan is redrawn and that deviations may occur in the drawing.

GENERATED FLOOR PLAN

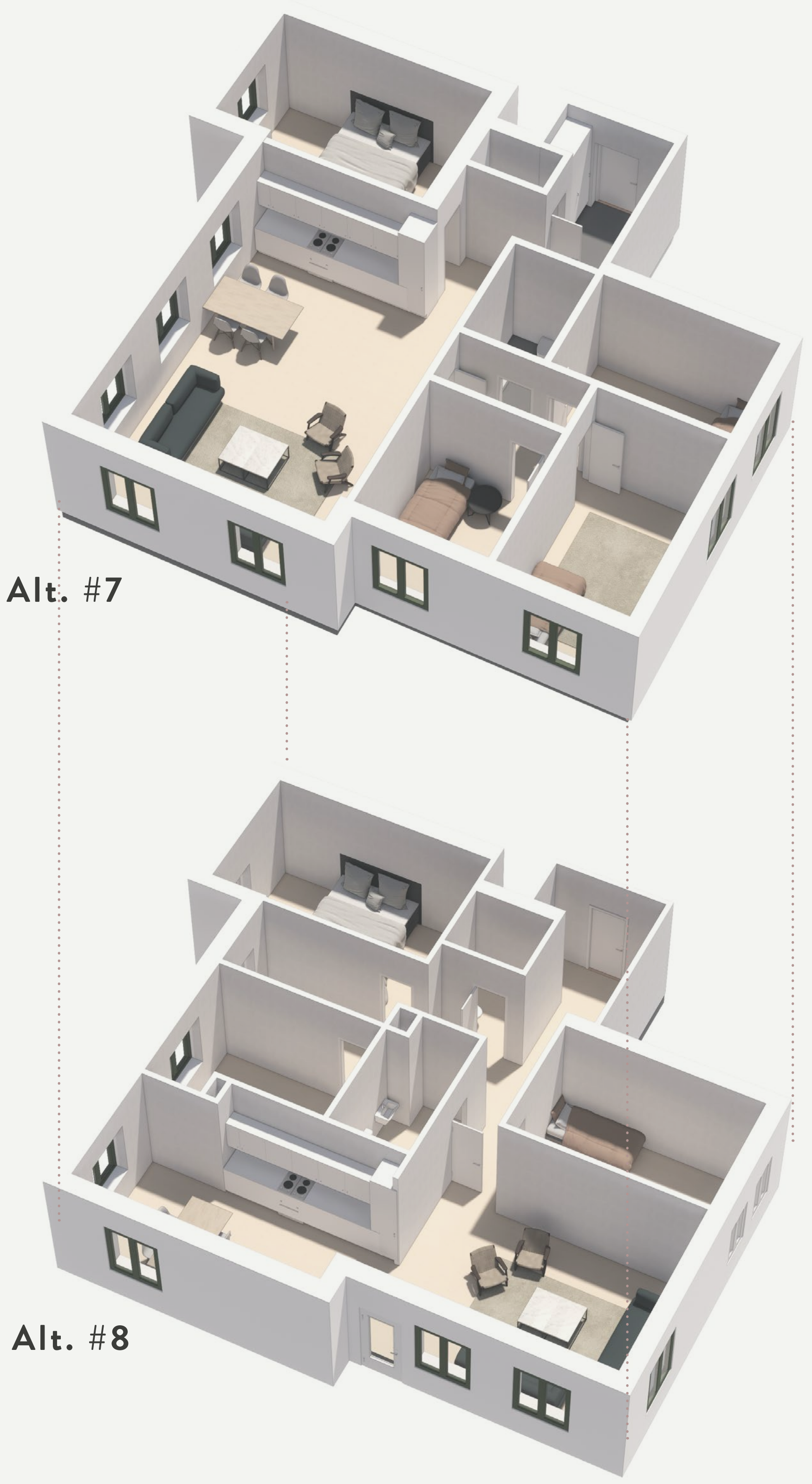


Alternative: #8

The generated alternative most similar to the real architectural floor plan is alternative number 8. The proposal contains almost the same room types and almost the same dimension. Thus, the study room is rarely used as a room type in the algorithm. It is more likely to use an inner hall for connecting to bedrooms and therefore the study room is excluded in the generated alternative. Instead an inner hall and closes is used.

The other generated alternatives differ in quality. Some of them are realistic, such as alternative 3 and 4, and some are unrealistic due to complex corridors, problematic daylight situations and no furnishing possibilities.

3D EXTRUSIONS



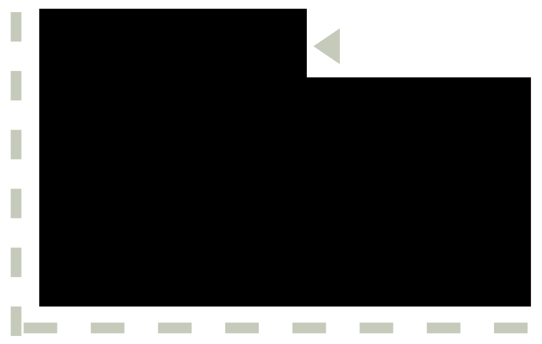
All of the generated layouts can be extruded and represented in 3D. In this case this has been done in order to visually show the generated apartment in 3D with manually added suggested furnishing. Two of the alternatives are presented to the left and the rest can be explored by scanning the QR code below.

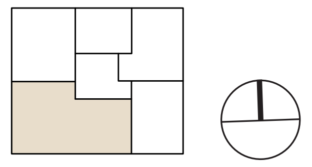
SCAN THE QR CODE FOR MORE FLOOR PLAN RESULTS

FLOORPLANNER.TECH

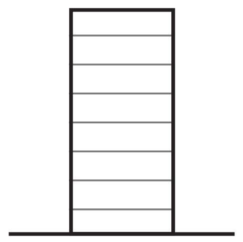
CASE 02

Input

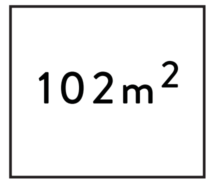




Apartment/Staircase



Typology



Apartment size

The second reference case used in the validation process is an 102 square meter big corner apartment. The shape is one of the most common ones in the data set so the reason why this shape is used is because of its common existence.

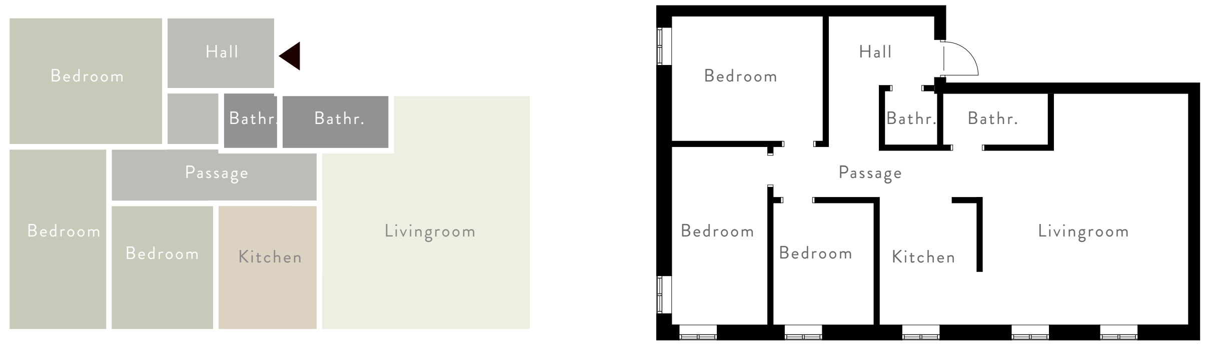
Another reason why the shape is used is because of its simplicity and regular boundary. It is used in order to proof the algorithms suitability for simple and traditional shapes.

No niches means more flexibility in how the spaces are placed and also how the "room growth" part of the algorithm will proceed.

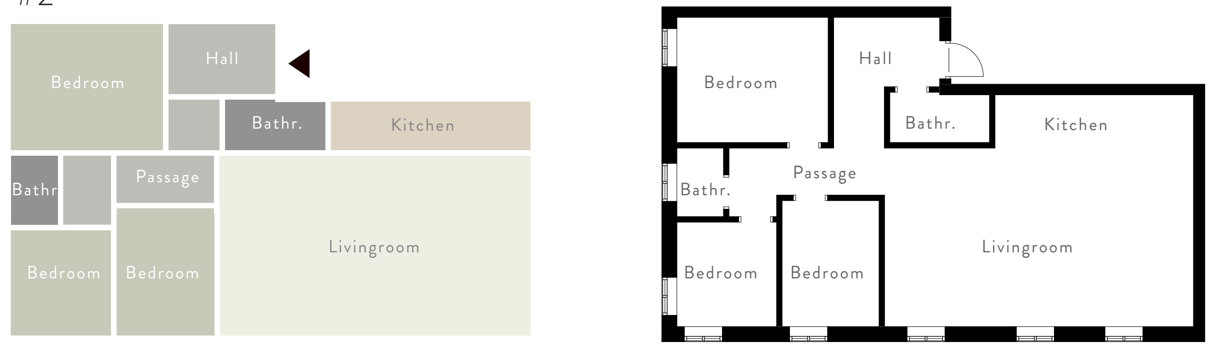
On the next two pages the layout configurations are presented. The real architectural floor plan is also presented together with the most similar generated one.

Generated layouts
Scale 1:200

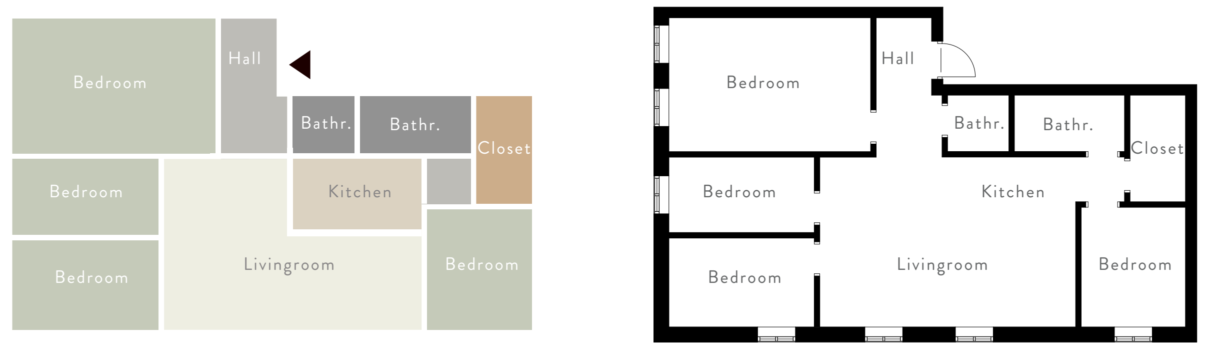
#1



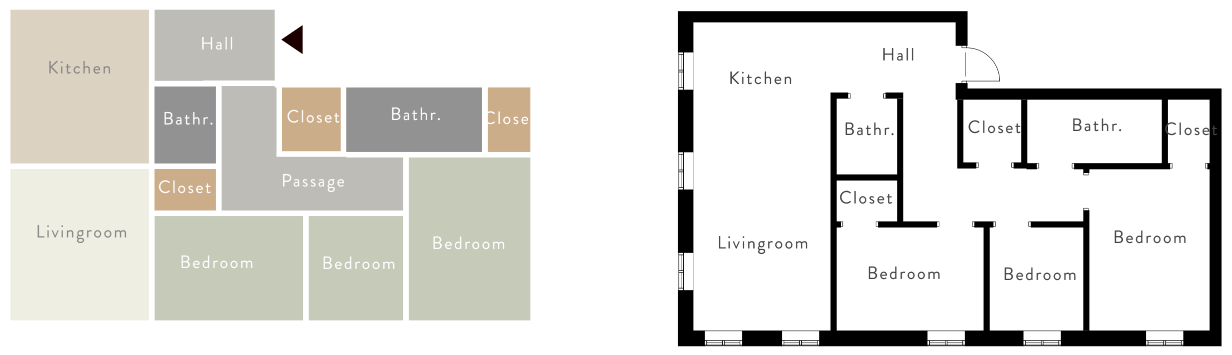
#2



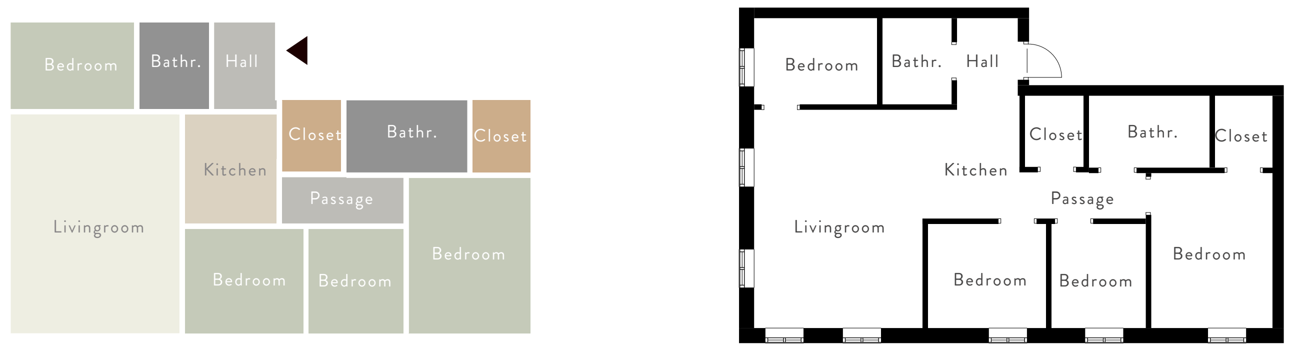
#3



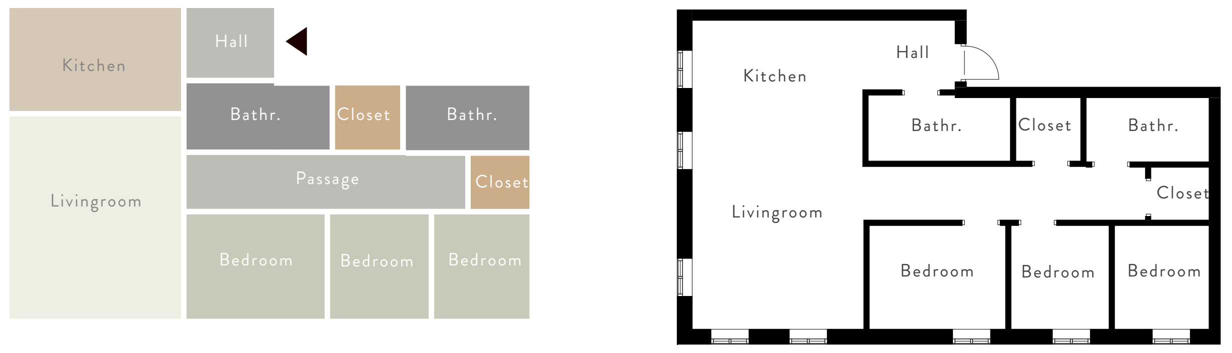
#4


Kitchen
Hall
Bathr.
Closet
Bathr.
Close
Closet
Passage
Livingroom
Bedroom
Bedroom
Bedroom
Kitchen
Hall
Bathr.
Closet
Bathr.
Close
Closet
Livingroom
Bedroom
Bedroom
Bedroom

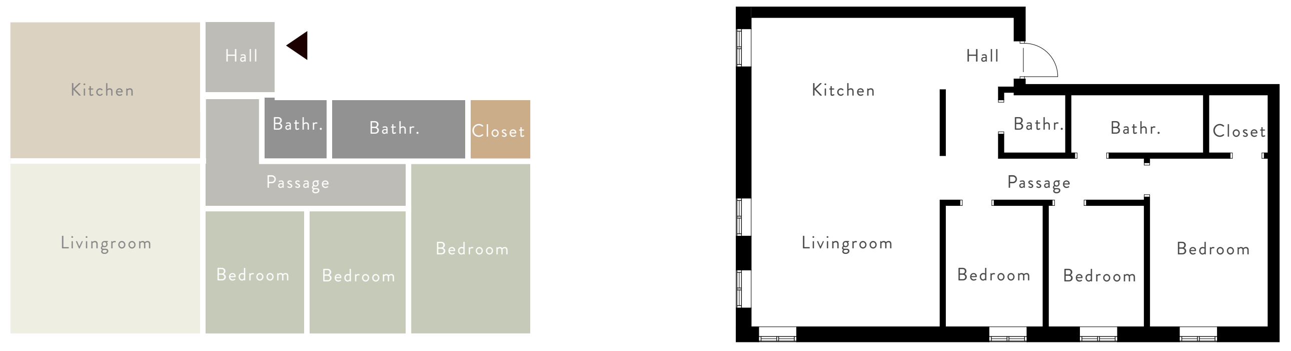
#7


Bedroom
Bathr.
Hall
Closet
Bathr.
Closet
Kitchen
Passage
Livingroom
Bedroom
Bedroom
Bedroom
Bedroom
Bathr.
Hall
Closet
Bathr.
Closet
Kitchen
Passage
Livingroom
Bedroom
Bedroom
Bedroom

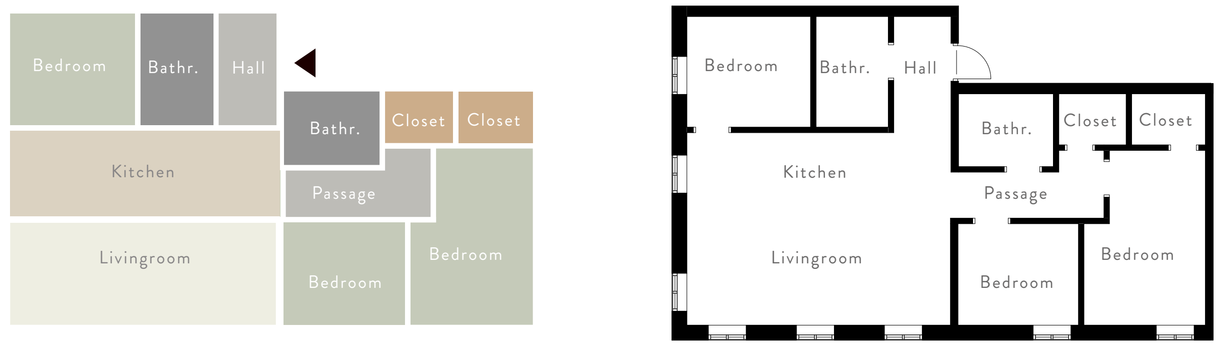
#5


Kitchen
Hall
Bathr.
Closet
Bathr.
Passage
Closet
Livingroom
Bedroom
Bedroom
Bedroom
Kitchen
Hall
Bathr.
Closet
Bathr.
Closet
Livingroom
Bedroom
Bedroom
Bedroom

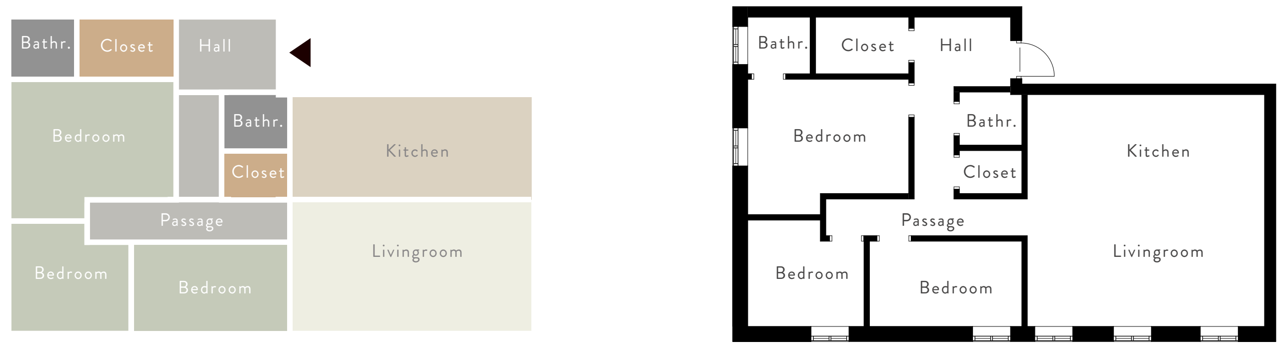
#8


Kitchen
Hall
Bathr.
Bathr.
Closet
Passage
Livingroom
Bedroom
Bedroom
Bedroom
Hall
Kitchen
Bathr.
Bathr.
Closet
Passage
Livingroom
Bedroom
Bedroom
Bedroom

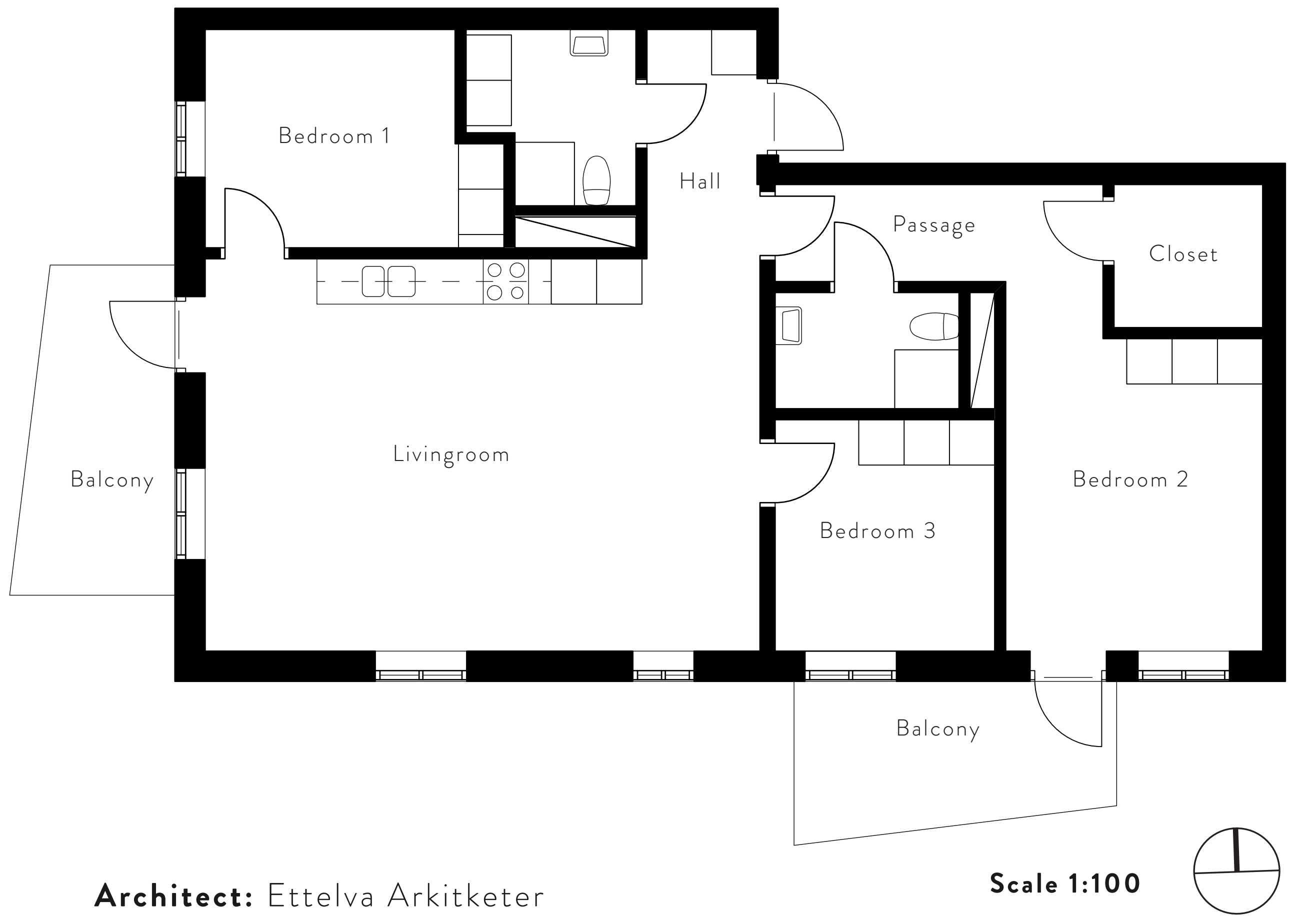
#6


Bedroom
Bathr.
Hall
Bathr.
Closet
Closet
Kitchen
Passage
Livingroom
Bedroom
Bedroom
Bedroom
Bathr.
Hall
Bathr.
Closet
Closet
Kitchen
Passage
Livingroom
Bedroom
Bedroom

#9


Bathr.
Closet
Hall
Bedroom
Bathr.
Closet
Kitchen
Passage
Livingroom
Bedroom
Bedroom
Bathr.
Closet
Hall
Bedroom
Bathr.
Closet
Kitchen
Passage
Livingroom
Bedroom
Bedroom

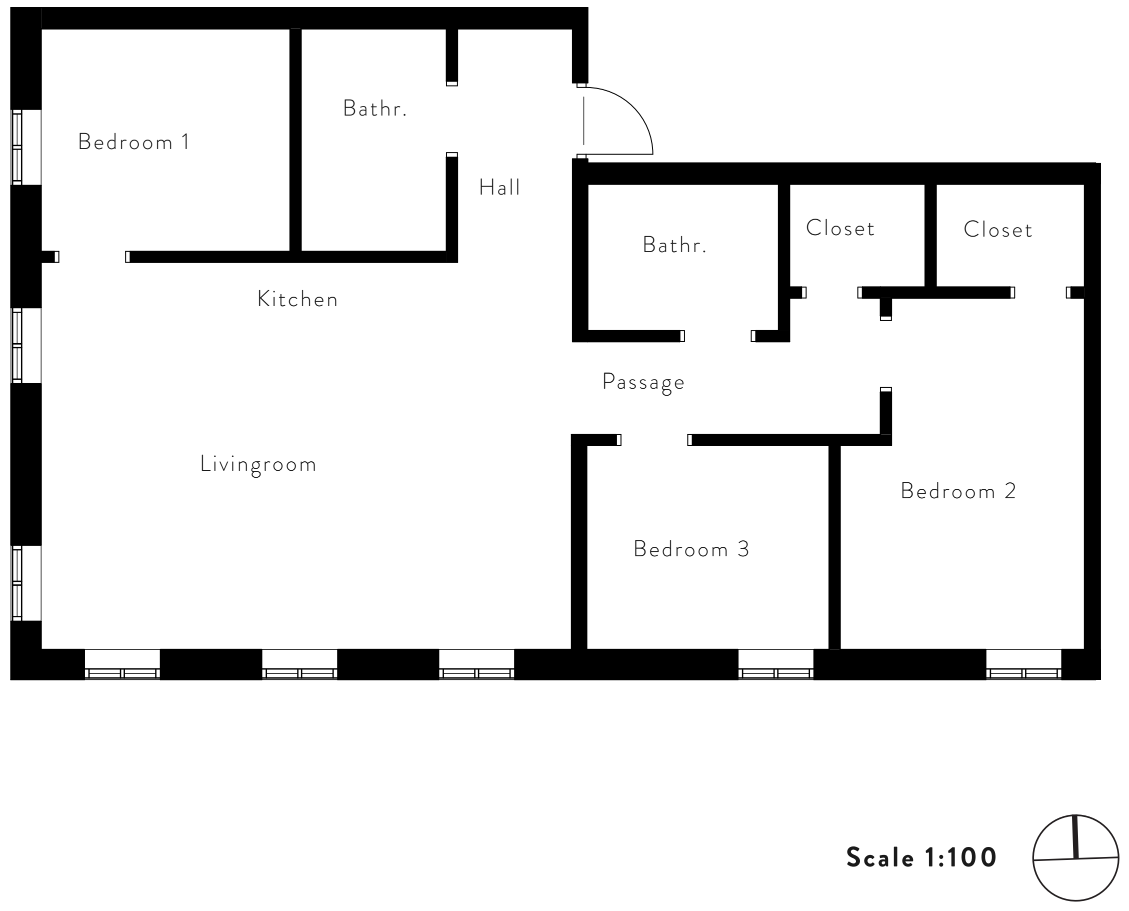
ARCHITECTURAL FLOOR PLAN



Architect: Ettelva Arkitketer
Developer: Veidekke
Project: Änggårdsblicken

The architectural floor plan is presented above. The floor plan consists of three bedrooms, a living room, kitchen, hall, two bathrooms and a study closet. Note that the floor plan is redrawn and that deviations may occur in the drawing.

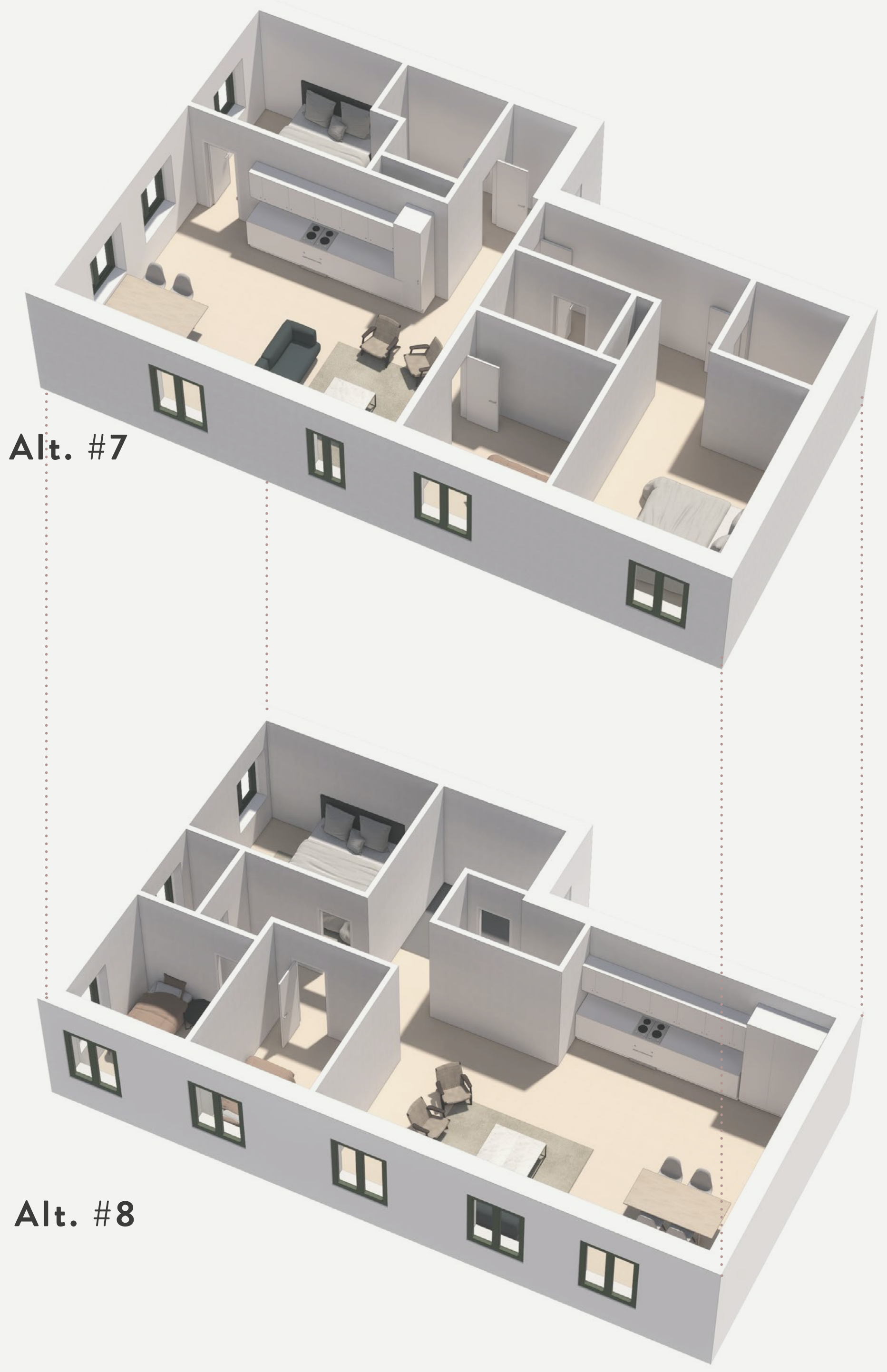
GENERATED FLOOR PLAN



Alternative: #6

The generated alternative similar to the real architectural floor plan is alternative number 11 in this case. The proposal contains almost the same space types and almost the same space positions. Thus, the passage is situated further away from the apartment boundary since the algorithm never allows passages to be situated next to the boundary. That means that the architectural proposal presented here wont be created what so ever. The most similar alternative is therefore an alternative with the passage closer to the middle.

3D EXTRUSIONS



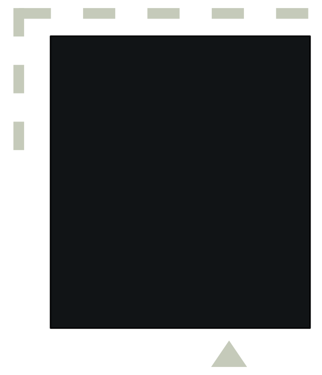
All of the generated layouts can be extruded and represented in 3D. In this case this has been done in order to visually show the generated apartment in 3D with manually added suggested furnishing. Two of the alternatives are presented to the left.

SCAN THE QR CODE FOR MORE FLOOR PLAN RESULTS

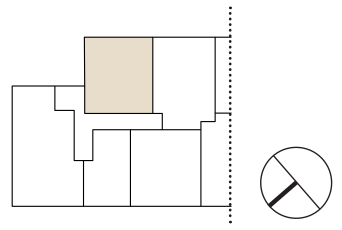
FLOORPLANNER.TECH

CASE 03

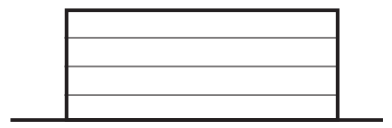
Input



scale 1:200



Apartment/Staircase



Typology

$49\ m^2$

Apartment size

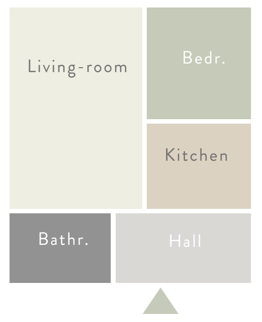
The third case used in the validation process is an 49 square meter one sided apartment. The shape is one of the most common ones in the data set so the reason why this shape is used is because of its commonness and simplicity.

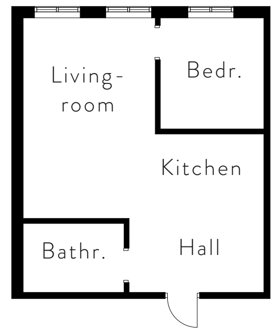
The apartment is much smaller than the previous presented shapes and the facade is shorter. This means fewer layout possibilities, but the point is not to proof variation here but rather validation.

On the next page the layouts are presented. The real architectural floor plan is also presented together with the most similar ones among the generated proposals.

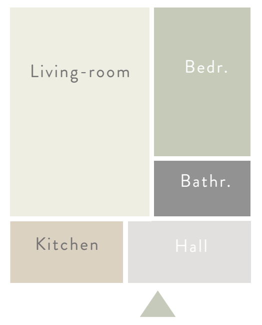
Generated layouts
Scale 1:200

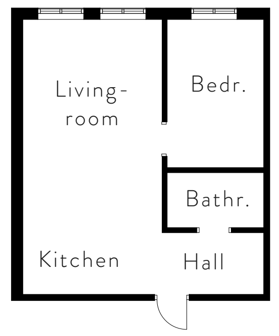
#1



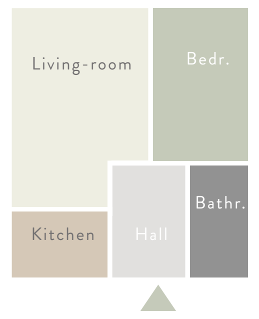


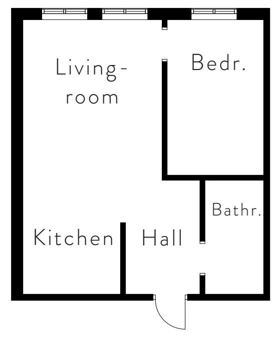
#2



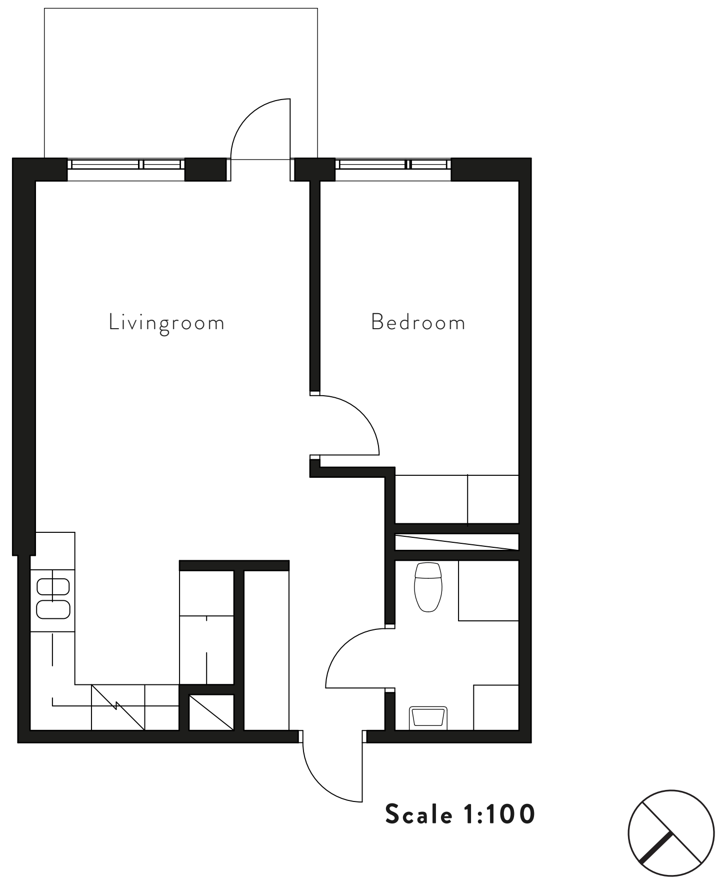


#3





ARCHITECTURAL FLOOR PLAN

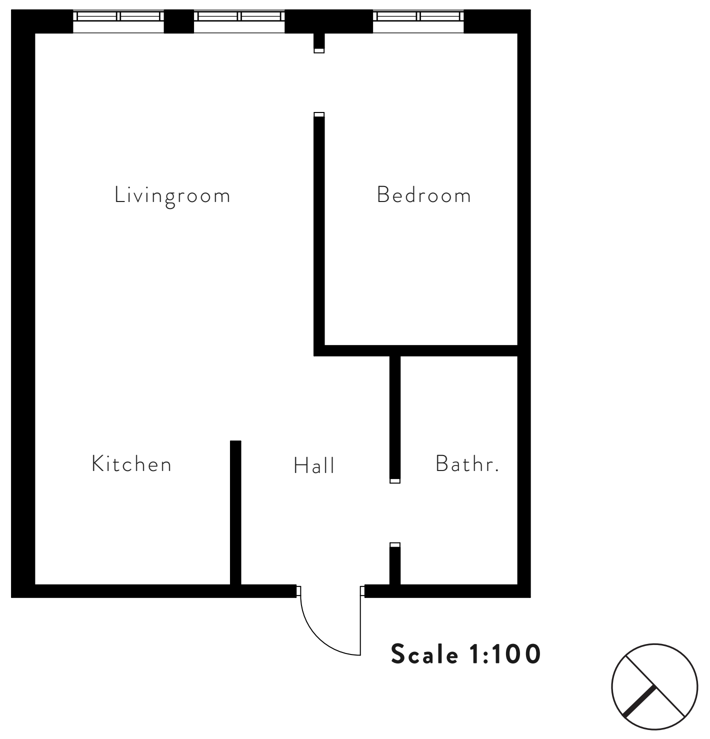


Architect: Sweco Architects
Developer: JM
Project: Frekvensen

The architectural floor plan is drawn by Sweco Architects in collaboration with JM. It consists of one bedroom, a living room, kitchen and a bathroom.

Note that the floor plan is redrawn and that deviations may occur in the drawing.

GENERATED FLOOR PLAN

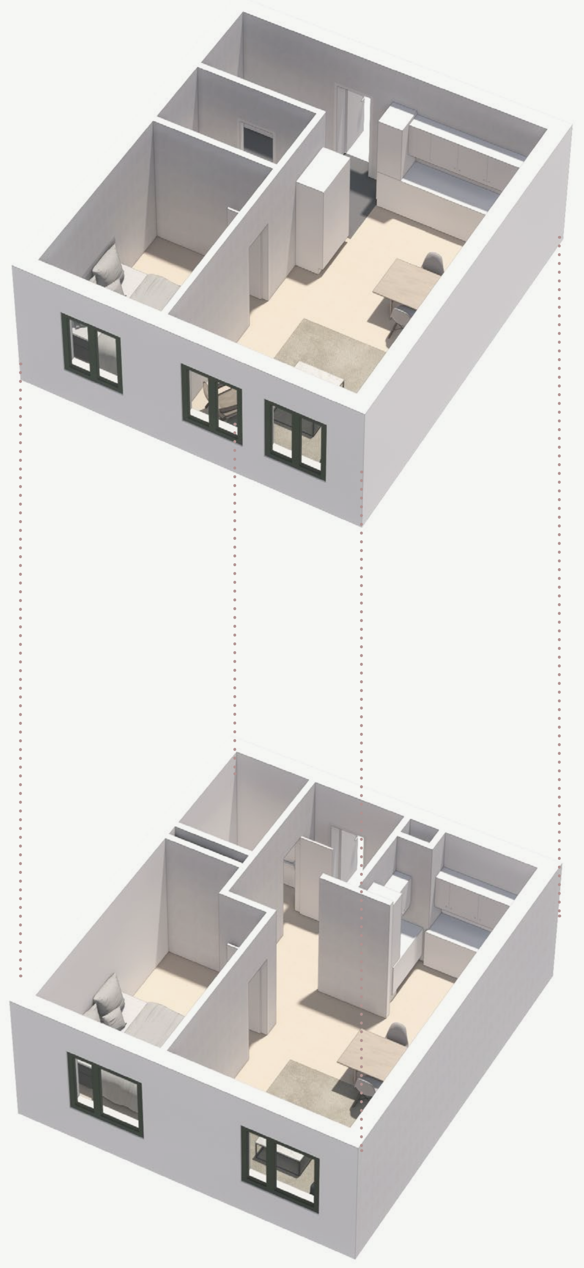


Alternative: #3

The most similar proposal among the generated alternatives is number 3. The proposal contains the same room types as the architectural proposal and the rooms have almost the same dimensions.

The openings generated in the post processing part of the algorithm differ a bit from the original proposal though but apart from that the floor plan draft is similar to what the architect draws.

3D EXTRUSIONS



All of the generated layouts can be extruded and represented in 3D. In this case this has been done in order to visually show the generated apartment in 3D with manually added suggested furnishing. Two of the alternatives are presented to the left.

SCAN THE QR CODE
FOR MORE FLOOR
PLAN RESULTS

FLOORPLANNER.TECH

06 DISCUSSION & CONCLUSIONS

DISCUSSION

Data set
When it comes to residential floor plan design relationships can be found by using statistics of existing drawings. There are logical connections, but there is neither an accurate model nor a rule of thumb for how to create a floor plan. The room sizes and areas can easily be decoded but do in some cases depend on other room sizes. A larger kitchen can result in a smaller living room and such relationships are difficult to cover. Different variations within the same room type can also result in less precise input parameters in terms of area and ratio.

The data set represent general trends and should be seen as a proof of concept for the procedural algorithm's input.

Algorithms in the design process
After comparing various algorithms the ”grow based” PCG method from the gaming industry applied on architectural floor plans provides fast and plausible results, which made it stand out from the other presented methods presented in chapter 02. After improvements, such as facade inclusion and a fixed access point, the implementation shows promising results.

The method can be a useful way of discovering risks at an early stage. Just like simulations in other scientific fields, such as daylight simulations etc., generative design scripts can be important tools in the early design phase.

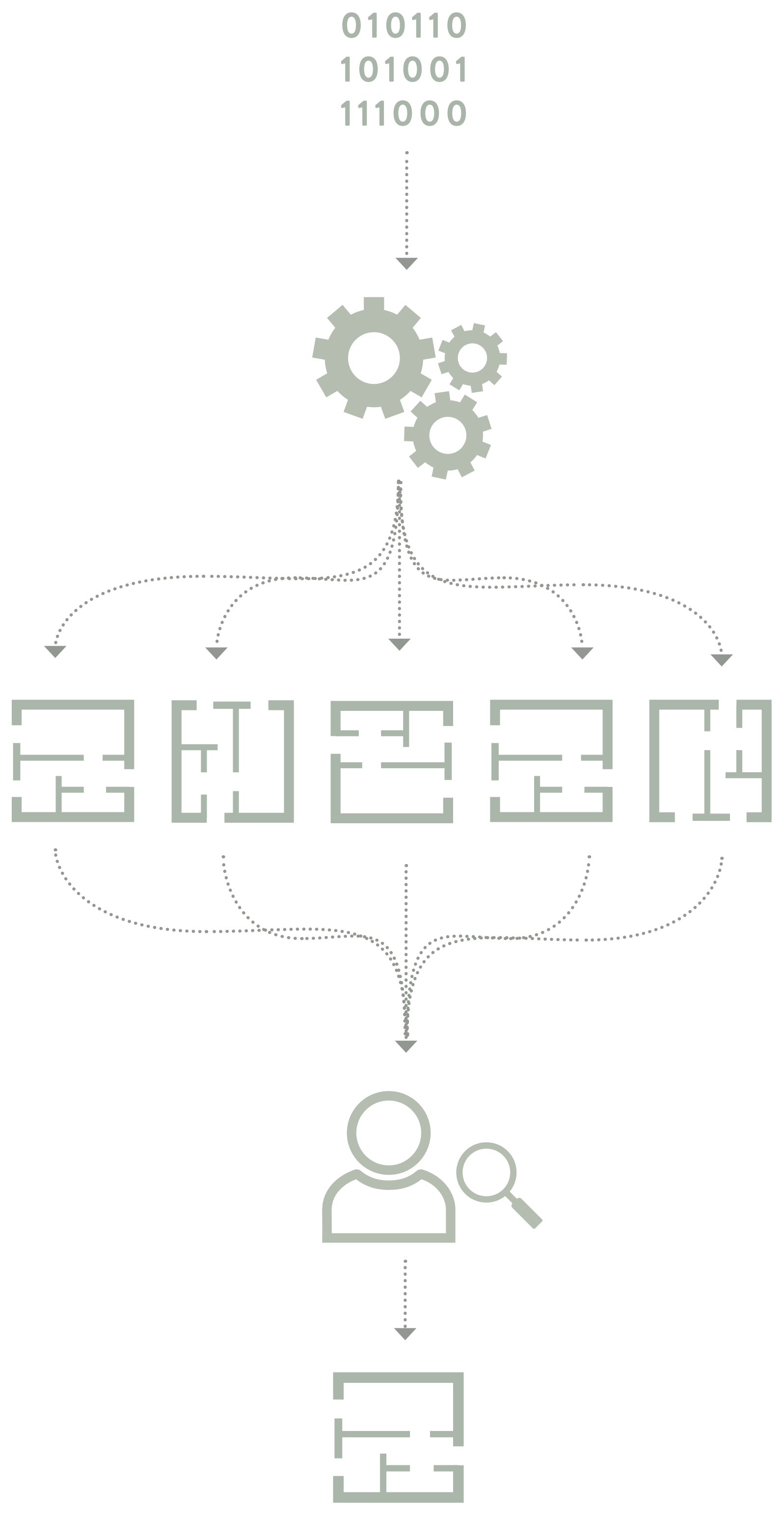
The implementation of the PCG method for generating floor plans shows that the method perform well, however there are some limitations. It is difficult to cover all of the regulations and quality requirements related to floor plans and because of the complexity of floor plan design. The algorithm cannot provide a final layout without human involvement. However, there is no reason to believe that the method could not deliver final layouts in the future.

Algorithm improvements
Certain improvement possibilities have been identified in the algorithm. Firstly, the method for constraining the dimensions of a specific room needs to be improved. The aspect ration (width-to-length) is a variable included in the model, but there are cases when the algorithm generates rooms that gets unrealistic dimensions. Putting additional constraints on the width-to-length ratio of a particular room could improve the method's ability of handling such cases.

Secondly, the ”Room Placement” function needs to be improved in order to avoid unrealistic placements of the rooms. In most of the cases it gives plausible results but there is a small chance that the algorithm will create room placements that results in long and irregular corridors in the post processing part.

In order to solve the latter issue evolutionary algorithms could be added and the corridor length could be the fitness value in an optimization problem. By allowing the corridors along the outer walls, (the boundary of the apartment) would also create more possibilities for the shortest path and in the end create more realistic plans.

Floor plan design is an iterative process where room adjacencies and circulations need to influence, and be influenced by, the topology of the building. A procedural algorithm approach



has difficulties in handling this itterative process and for some cases it will be hard to maintain all requirements on both a room level and an inter-room level.

Qualities aspects
There will always be different preferences and importance of different aspects and sometimes there will be contradicitive goals. As an example good daylight is very often something to strive for, but good daylight does often mean high solar heat loads, which is something that should be avoided. Therefore quality aspects could result in conflicts and the algorithm will not be able to create a ”perfect” plan.

Aspects such as sight lines, axiality, privacy, paths and circulation are not included in the generative algorithm. There are more aspect related to architectural qualities that would be of interest to include in the algorithm. The most beneficial way of handling this would be to only consider geometrical and adjacency constrains for the generative part and let the quality aspects be part of an evaluation phase. This would mean that the algorithm would generate proposals without taking these additional quality aspects into consideration, and then low quality proposals would be removed in a post processing step.

The quality aspect can also be discussed in relation to the purpose of increasing efficiency. This is something that will probably be of great financial importance, but also a way to compare financial values in relation to quality. By analysing the different generated alternatives the architect can use the generated alternatives as a way to communicate, compare and compromise between different ideas and quality aspects in relation to other aspects.

The role of the architect
The intention of the algorithm is not to exclude the architect all together. This method should not be considered as an attempt to make the profession of the architect obsolete in the age of computers and algorithms. The new tools and digital possibilities should be seen as a way for architects to investigate drafts of layouts faster and to increase the creativity. The algorithm can be seen as a a platform of ideas that will give the architects insights and multiple starting points.

The generative design approach can be considered as a process that could help in individualising floor plans. It will produce a lot of alternative outputs and enables architects to find solutions that they maybe did not think of from the beginning. This means that this approach will help in creating less structured and monotonous plans (especially when it comes to bigger apartments).

Risks and concerns
The results presented in the previous chapter indicate promising implementation possibilities. However, there might be risks with pushing the algorithmic approach to far. There is a risk that we blindly trust the algorithm and proceed too far in the design process without validating the results from a human perspective.

As mentioned the algorithm iterates over geometrical constraints, not quality aspects. Therefore it is of great importance to validate the suggested plans from a critical perspective and validate human factors, even if a post processing algorithm evaluating quality aspects could be involved.

Another risk is that we miss some possible layout configurations. The algorithm is good at general requirements but is not good at handling exceptions and rare edge cases. There might be a risk that the user will not be aware of this and trusts the algorithms capability of investigating all different combinations. However, these configurations will always be greater in numbers than what a traditional architectural approach will come up with, but since the algorithm is based on strict input parameters it might not be as limited in generating uncommon proposals. There might be additional possible configurations than what the algorithm comes up with.

Future work
As mentioned, the algorithm iterates over geometrical constraints, and not quality aspects since they can be contradicitve. As a part of a future work it would be of great interest to investigate how to ensure quality in a digital post processing step. If interior living qualities could be measured digitally it would be possible to compare the generated results in a consistent way.

Aspects such as axiality, sight lines, circulation etc. could be taken into consideration and the generated floor plans could get scores for each and every aspect, see figure 3. Another aspect that needs to be included in the future development is the furnishing possibility. Among the generated floor plans certain room dimensions and door positions result in pour furnishing possibilities. The kitchen line-up does also need to be taken into consideration for future work since it does not fulfil requirements in the current version of the algorithm.

Yet another improvement area refer to adjacent apartment layouts. The demarcation for this thesis is set to the unit scale (one apartment) but spaces, such as spaces with plumbing requirements, within one unit is often adapted to adjacent apartments in real-world solutions. The structural system is also something to take into consideration for further investigations.

Overall, the algorithm presented in “Floor Plan Parametrics” provides opportunities to create a homogeneous dataset useful for future research. Even though all generated results are not valid when it comes to tolerances and furnishing possibilities they can be used together with a validating algorithm. The generative algorithm together with validating algorithms can create feedback loops and work as a foundation for further investigations.

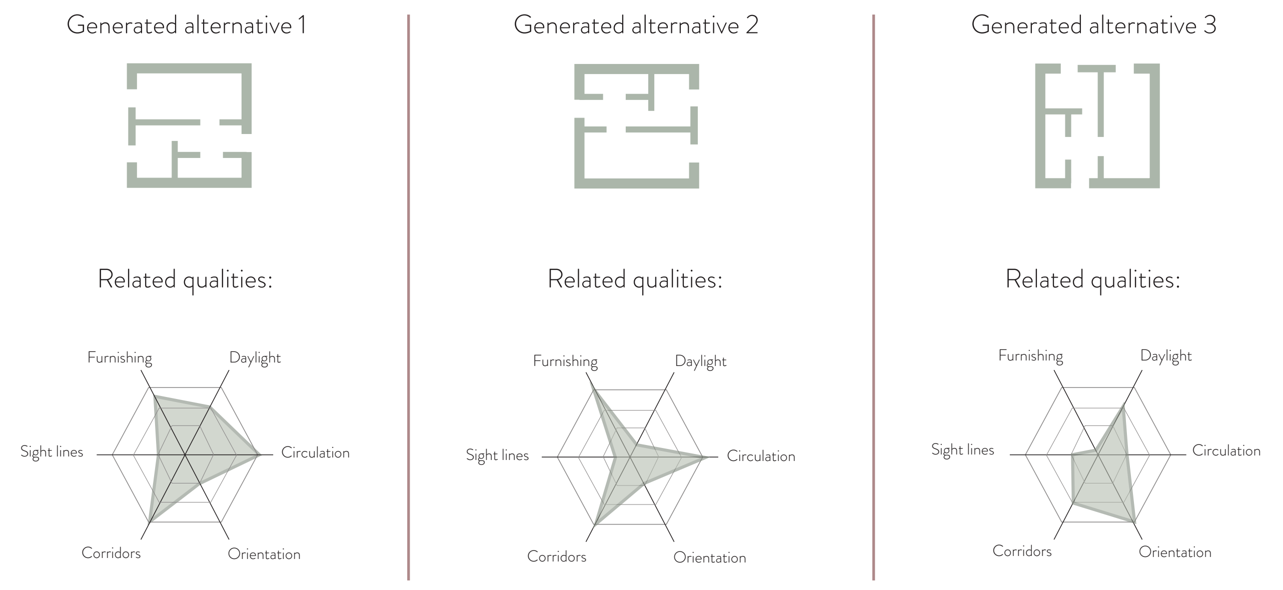


Figure 3.0
Floor plans created by the algorithm and exaples of related qualities that can be defined and compared.

CONCLUSIONS

Working with research on and for design has led to conclusions for how algorithms can be included in the work of residential floor plan design. The outcome shows that procedural algorithms results in plausible solutions when good input values are provided.

How can the logic behind residential floor plans be translated into quantifiable variables by using statistics of existing floor plans?

Architectural floor plans have significant and defined logic. However, **it is not clear how they can be represented with a specific and detailed set of rules** as shown in chapter 03.

Space attributes extracted from the data set proofed the **similarities between spaces of the same type**. There are clear patterns on a room level, but there are less clear statistical patterns on an inter-room level when it comes to adjacencies. There is no rule of thumb for how to place the rooms but adjacencies and circulations need to influence, and be influenced by, the topology of the apartment.

For an approach with a boundary as input **adjacencies seem to be allowed to occur as a consequence of other underlying factors,** such as facade position and boundary shape. Adjacencies seems to be a randomized parameter with the exception of the combination of kitchen- living room - dining and hall - bathroom where a strong statistical correlations can be seen.

When it comes to circulation, there is a clear statistical pattern showing how circulation works in the majority of the space configurations. Graph representations can be generated and by using graphs, floor plans can be compared based on similarity of connections between rooms.
The logic of circulation and adjacencies can be decoded using statistical tools.

How well can an algorithm generate residential floor plans similar to those created by traditional architectural methods?

By comparing the generated floor plans with real architectural floor plans it can be concluded that the "grow-based" procedural algorithm efficiently generates plausible drafts of floor plan layouts. Among the generated layouts **several realistic proposals** can be identified. Proposals **similar to what architects have designed has been seen.**

However, the algorithm iterates over geometrical constraints - not quality aspects. The reason is the existence of contradictive goals. It would be more beneficial letting a **post processing algorithm, together with architects and expertise from other disciplines, define and evaluate qualities** as a result of what subjective quality aspects that is important in every unique project.

The algorithm is based on strict input parameters and might be limited in generating uncommon layouts. The algorithm will not take all edge cases into consideration and there might be more possible configurations than what the algorithm comes up with. However, **the strengths is the quantitative testing** and the execution time.
It can generate one option in **less than 300 milliseconds**, and will thereby relieve work from **the architect who can focus on quality aspects**.

Overall the algorithm provides plausible results and contributes to improved decision support within residential development.

08 REFERENCES

REFERENCES

Publications

B. Whitehead, and M-Z. Eldars, An approach to the optimum layout of singlestorey buildings. The Architects" Journal (17 June), 1964, 1373-1380.

B. Hillier (2007) Space is the Machine: A Con-gurational Theory of Architecture.

B. Hillier and J. Hanson (1984) The Social Logic of Space

Boverkets byggregler, BFS 1993:57 med ändring-ar t.o.m 2005:17 BBR

Bra bostadsutfromning, Boverket, Regler, Kvalitet, Kostnader och exempel för flerbostadshus ISBN: 91-7147-832-9

C.Cheng (2020) House-GAN: Relational Gene-rative Adversarial Networks for Graph-constrai-ned House Layout Generation

C. Peng, Y.Yang, and P. Wonka (2014). Compu-ting layouts with deformable templates

D. Bengtsson and J. Melin (2016) Constrained procedural floor plan generation for game environments

D. Sharma, N. Gupta, C. Chattopadhyay, S. Mehta (2017) A Deep Architecture for Automatic Analysis and Retrieval of Building Floor Plans

E. Hahn, P. Bose, and A.Whitehead (2006) Persistent realtime building interior generation

E. Merell, Schkufza, Koltun (2010) - Compu-ter-Generated Residential Building Layouts

F. Marson and S. R. Musse (2010) Automatic

H. Zheng, W. Huang (2018) Architectural Drawings Recognition and Generation through Machine Learning

J.Martin(2005) Algorithmic beauty of buildings - methods for procedural building generation. Master's thesis, Trinity University, 2005

L. Rinde and A. Dahl (2008) Procedural Generation of Indoor Environments

L. Heras(2013) Statistical segmentation and structural recognition for floor plan interpretation

Lopes, Tutenel, Smelik, Kraker, Bidarra (2010) - A Constrained Growth Method For Procedural Floor Plan Generation

P. Merrell (2007) Example-based model synthe-sis. In I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games

Petrova, Svidt, Pauwels, Jensen (2109). Towards Data-Driven Sustainable Design: Decision Sup-port based on Knowledge Discovery in Disparate Building Data

Real-Time Generation of Floor Plans Based on Squarified Treemaps Algorithm

S. Chaillou. (2019). Artificiall Intelligence and Architecture | Harvard Graduate School of Design